Oracle Tuxedo Application Runtime for CICS

Reference Guide

11*g* Release 1 (11.1.1.3.0)

December 2011

ORACLE

Oracle Tuxedo Application Runtime for CICS Reference Guide, 11*g* Release 1 (11.1.1.3.0)

Contents

CICS Runtime Configuration Files

Environment Variables

Server Configuration

/Q Configuration for CICS Runtime

Security Configuration

CICS Commands and Parameters Coverage

System Commands and Transactions

CICS Runtime Preprocessor

CICS Runtime Messages

Configuring Oracle Tuxedo XA Connection to DB2 Using DB2 Connect

CHAPTER 1

Introduction

The purpose of this document is to document the Oracle Tuxedo Application Runtime for CICS configuration, describing the configuration files, the environment variables and the server configuration including CLOPT options.

In addition this document includes information about the Oracle Tuxedo Application Runtime for CICS Preprocessor. Although the CICS Runtime Preprocessor is not a Runtime tool, it is used on an ongoing basis on the target platform when compiling COBOL programs for use with CICS Runtime.

The document includes the following chapters:

- CICS Runtime Concepts
- CICS Runtime Servers
- CICS Runtime Configuration Files
- Environment Variables
- Server Configuration
- /Q Configuration for CICS Runtime
- Security Configuration
- CICS Commands and Parameters Coverage
- Other Configurations
- CICS Runtime Preprocessor

- CICS Runtime Messages

CICS Runtime Concepts

Purpose

There are different approaches to migrating CICS applications to a UNIX/Linux environment. The purpose of this section is to give an understanding not only of what CICS Runtime is and what it does, but also what it is not and what it does not do. Particularly the aim is to explain that CICS Runtime is not an emulation of the CICS application environment on a UNIX/Linux system. CICS Runtime keeps the application logic contained in the COBOL programs but is totally compatible with the Tuxedo client/server architecture for the execution of that logic. CICS Runtime provides a middleware between the CICS coding in the programs and the Tuxedo OLTP system, UNIX\Linux OS and Oracle database responsible for executing transactions and providing persistence

CICS Runtime Goals

The first aim of CICS Runtime is to preserve the considerable investment already made in CICS applications by allowing migrated programs to run unchanged (except for a syntactic adaptation) by using an API emulation runtime on top of native Tuxedo features. This means the impact of migration is limited on:

- 3270 screens and BMS management; there is no impact on application end-users.
- EXEC CICS calls; there is no impact on developers.

At the same time, CICS Runtime is run entirely on a robust production environment based on Tuxedo that protects and guarantees application functionality.

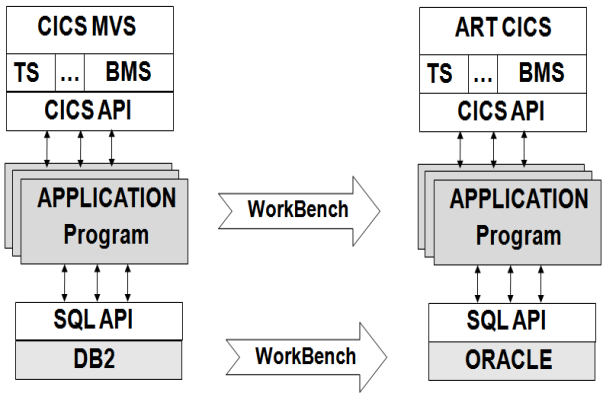
In fact, CICS Runtime gives customers the benefits of Tuxedo distributed architecture without impacting application APIs. It allows the key strengths of Tuxedo to be leveraged and allows routes to the future including SOA to be opened.

CICS Runtime Architecture

Software Development Perspective

The following diagram shows the software bricks used to create the application environment on the migration source and target platforms.

Figure 2-1 Migration Software Environments



Except for the top and bottom bricks, there is little else that changes for the software developer.

Programmatic Interface

CICS Runtime offers a library of CICS API reproducing the functionality of the z/OS CICS API and offering equivalent services to the migrated CICS applications, and in addition it offers BMS capabilities with support for 3270 screens.

In a CICS application on a z/OS platform all interactions with resources are done thru the EXEC CICS API (with the exception of DB2).

The CICS Preprocessor (on Z/OS) transforms these EXEC CICS into calls to the CICS library.

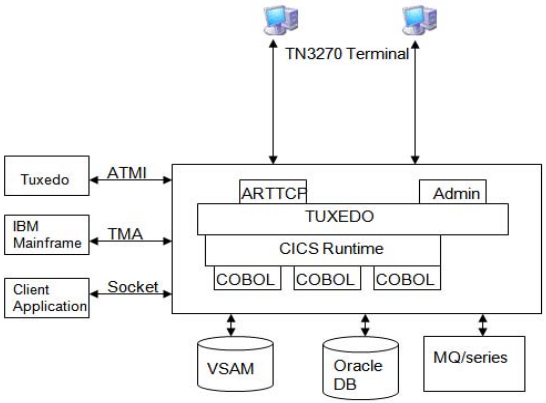
On the target platform, the same philosophy is used; an CICS Runtime Preprocessor (an CICS Runtime compile-time tool) transforms these EXEC CICS into calls to the CICS Runtime library.

For the software developer, there is little that changes. The CICS Runtime preprocessor automatically changes the CICS API that is called. There are some limitations in the command that can be used; these are described in CICS Commands and Parameters Coverage.

System Administration Perspective

For a software administrator, there is little that remains the same. There are the same applications to be executed and end-users still access applications using the same 3270 terminals. Apart from that, everything else is different and relies on a native Tuxedo architecture to manage transactions with the aid of CICS Runtime to provide an API between the COBOL programs and Tuxedo.

Figure 2-2 CICS Runtime Architecture



CICS Runtime provides the run-time support to allow converted CICS applications to run in a robust production ready environment based on TUXEDO /T, while offering applications functionally equivalent behavior.

In term of deployment, the system is distributable on one or multiple machines in a single TUXEDO domain, or several domains communicating through a TUXEDO domain.

In term of administration, the administration is native TUXEDO, with all tuxedo administration tools being normally usable, plus a few administration tables for CICS only concepts, like terminal definitions, and Transaction - First Program table.

z/OS CICS Concepts in a CICS Runtime Environment

Developers and administrators used to working in a z/OS CICS environment naturally want to know how concepts familiar on the source platform are implemented on the target platform.

Table 2-1 gives an overview of how the source platform notions have an equivalent on the target platform.

Table 2-1 Source Platform Notions

z/OS CICS	CICS Runtime
CICS Session	TUXEDO Session
Transaction Transaction First Program	Tuxedo Service Program associated with service
L.U.W. SYNCPOINT [ROLLBACK]	Transaction tpcommit() / tpabort()
COBOL Program	COBOL Program
CALL "SUBPGMX"	CALL "SUBPGMX"
EXEC CICS LINK local	Local call with memory stacking and isolation (same isolation)
DPL (Distributed Program Link)	Tpcall to a tuxedo service
Conversational programs Pseudo-Conversational	tpconnect()/tpsend()/tpreceive() Request/Response tpcall() / tpreply()
COMMAREA (state info/context)	(Tuxedo is stateless, context passed through buffers)

Administrative Tasks

Most of this guide describes how to administer resources for CICS application running with CICS Runtime on a Tuxedo system. CICS Runtime uses Tuxedo natively with the addition of a few extra resource files and servers.

This provides all the robust characteristics of Tuxedo including:

- Load balancing, priority management, Dynamic routing,

- Supervision, automatic restart of servers,
- Transparent distribution on multiple machines,
- Server migration from one machine to another,
- The distribution of the load of a new machine is very simple.
  - Declare the machine in the UBBCONFIG
  - Launch a few servers offering transactions on this machine

CICS Runtime Servers

The CICS Runtime Servers

This section describes the different servers and the role they play in the overall handling of transactions. The configuration of the servers is described in Server Configuration.

3270 Terminals and User Session Management (ARTTCPL/ARTTCPH)

Description

The role of these servers is to accept and manage user connections made thru a 3270 terminal emulator and manage the resulting user session and related 3270 inputs and outputs until the end of the user session.

Functionally, this role resembles the one played by the Terminal Owning Region in a CICS MRO configuration.

These 3270 user session management tasks are managed by a couple of server types: ARTTCPL and ARTTCPH, where the final ‘L’ stands for listener and the final ‘H’ stands for handler.

- ARTTCPL — Performs the role of a listener server. ARTTCPL listens to a public address – the address to which users wanting to connect to this application with a 3270 emulator will connect – then for each incoming connection request, it transmits this connection to one of its handler processes.

- ARTTCPH — Each handler process manages multiple connections including terminal I/O, user authentication, and calling the requested transactions on behalf of the user.

It is the role of the ARTTCPL to launch and manage the requested number of handler processes (ARTTCPH).

Each time a user requests a transaction, ARTTCPH transmits (via tpconnect) this transaction request to the transaction server.

This functionality resembles that provided by T.O.R. in a CICS MRO configuration when it routes a transaction to an A.O.R (Application owning Region).

Connection Server (ARTCNX)

This server offers technical services needed by terminal handlers during user connection and disconnection phases.

The technical services are offered using internal message oriented services like `connect` and `disconnect`:

- `connect` performs various initialization tasks such as attributing the user Session ID and Terminal_ID.
- `disconnect` manages final tasks during disconnection.

The connection server also provides a few classical CICS transactions:

- CESN: the Sign oN transaction
- CESF: the Sign ofF transaction
- CSGM: the Good Morning transaction (default Good Morning transaction)

See “Authentication Transactions” in Other Configurations for more information.

Synchronous Transactions and Program Management

Description

The task of these servers (`ARTSTRN` and `ARTSTR1`) is to offer application transactions and process the corresponding programs.

This server provides a similar functionality to that provided by an Application Owning Region in a CICS MRO configuration. `ARTSTRN/ARTSTR1` servers present application transactions as Tuxedo services and when receiving a transaction request execute the corresponding programs.

Theses servers are conversational in order to be able to manage true conversational CICS transactions.

Processing

1. When starting, a `ARTSTRN/ARTSTR1` server publishes one service per transaction it offers.
2. When a user transmits a transaction request, the ARTTCPH performs a `tpconnect` to the corresponding transaction (service).
3. One `ARTSTRN/ARTSTR1` server offering this service receives the request with the associated commarea and screen, then processes the transaction.
4. Then:
   - In the case of a normal pseudo-conversational CICS transaction; on the RETURN {TRANSID} the server replies to the client, finishing the conversation by a `tpreturn()` returning the new 3270 screen, and the commarea.

Non Concurrent Synchronous Transaction Servers (MAXACTIVE = 1 (ARTSTR1))

- On CICS:
  - Transactions that are defined as belonging to a transaction class are subject to scheduling constraints before they are allowed to execute.
  - If transactions belonging to an active transaction class are already running, any new transactions are queued.
  - The MAXACTIVE attribute is used to specify the maximum number of transactions that you want to run.
  - By putting your transactions into transaction classes, you can control how CICS dispatches tasks.
- In CICS Runtime:
  - The scheduling of transactions and the affectations of resources to group of transactions is performed differently. The number of servers offering transactions determines the scheduling of transactions, and the relative amount of resources affected to a group of transactions.
  - The special case of `MAXACTIVE` 1:

- This case is very specific, because it impacts the functional characteristics of the application.
- It ensures that two transactions of the same class will never execute concurrently. It defines a mutually exclusive behavior that is preserved on the target platform to guarantee correct behavior of the application.

The transactions belonging to a tranclass with a maxactive =1, will not be offered by ARTSTRN servers, because several such servers can automatically be started to manage parallel processing.

Instead, a dedicated type of server—`ARTSTR1`—is allocated to this role. An `ARTSTR1` server publishes the transactions belonging to one TRANCLASS with `MAXACTIVE = 1`, guaranteeing that two transactions of the same tranclass with maxactive =1, will not execute concurrently. In Tuxedo terms, guaranteeing than two such transactions are not published by two different servers.

To summarize the differences:

- `ARTSTR1`: Publishes only once transactions belonging to a `MAXACTIVE 1` tranclass.
- `ARTSTRN`: Publish as many times as needed, transactions with `MAXACTIVE >1`.

Temporary Storage Queue Management (ARTTSQ)

Description

The role of the `ARTTSQ` servers is to centralise the management the TS Queue operations which are requested by applications. These tasks are managed by `ARTTSQ` servers.

Depending on the workload expected on the TS queue, a single server or many `ARTTSQ` servers are configured.

`ARTTSQ` servers publish technical services:

- `TSQUEUE` to service operations on queues not matching any TS Queue Model.
- `{MODEL}_TSQUEUE` to service operations on queues matching a specific model, one such service must be published using one `ARTTSQ` server for each model.

In a simple configuration, a single `ARTTSQ` server will treat all the TS operations, offering the `TSQUEUE` service, and all `{MODEL}_TSQUEUE` services.

In a more complex configuration, one `ARTTSQ` server may offer the `TSQUEUE` and some `{MODEL}_TSQUEUE` services, while other `ARTTSQ` servers will each offer different `{MODEL}_TSQUEUE` services.

Transient Data Queue Management (ARTTDQ)

Description

The role of the ARTTDQ servers is to centralise the management the TD Queue operations which are requested by applications. These tasks are managed by one ARTTDQ server.

A single ARTTDQ server publishes one service per declared queue in the configuration file.and will treat all the CICS TD operations, offering the TD QUEUE service for each queue.

DPL Servers (ARTDPL)

In complex configurations an application may need to make distributed program calls. In this case another kind of server is needed to manage DPL. These tasks are managed by ARTDPL servers.

ARTDPL servers publish programs that are callable by EXEC CICS LINK as services, and manage the execution of these services.

Asynchronous Transaction Servers (ARTATRN/ARTATR1)

An application may request an asynchronous transaction launch using 'EXEC CICS START TRANSID' requests. In this case the request needs to be treated asynchronously by another server. These tasks are managed by ARTATRN/ARTATR1 servers.

These servers publish transactions callable by EXEC CICS START TRANSID as services named ASYNC_{Transaction_Name}, and manage execution of these services.

Conversation Server (ARTCTRN/ARTCTR1)

An application may request an a conversation launch using 'EXEC CICS CONVERSE' requests. In this case the request needs to be treated by another server. These tasks are managed by ARTCTRN/ARTCTR1 servers.

These servers publish transactions callable by EXEC CICS CONVERSE as services named {SysId}_{Transaction_Name}, and manage execution of these services.

Delayed Asynchronous Transaction (/Q Part)

Asynchronous transactions are launched using 'EXEC CICS START TRANSID' requests that may also by launched with a delay set to an interval or to a fixed time.

In this case the transaction request is deposited into a Tuxedo /Q Queue, and when the time is ready, the transaction will be automatically invoked.

For this feature to be available, a few extra components must be activated:

- A Tuxedo /Q Queue Space named `ASYNC_QSPACE` must be created.
- A Tuxedo /Q Queue named `ASYNC_QUEUE` must be created in the queue space.
- A `TMQFORWARD` server must be configured to receive messages from this queue and invoke the application transaction corresponding to the request.

Tip: `TMQFORWARD` will always call the same technical transaction called `ASYNC_QUEUE` (the name of the queue). This transaction will extract the field CX_TRANSID, which will contain the name of the real application transaction to call and will perform a TPACALL(NOREPLY) of this transaction and tpreturn immediately.

Administration Server (ARTADM)

The administration server is responsible for the administration of CICS resources. It provides the following functionalities:

- Takes charge of loading the resource definitions used by other servers.
- Offers the services used by artadmin (ART administration console) for dynamic administration of CICS resources and propagates the dynamic configuration requests from artadmin to all the appropriate servers in the system.
- Propagates the resource definition files to the slave machines from the central configuration repository when configured on each node in a distributed environment.

  The configuration files only need to be configured on the master node, and the administration servers propagate the configuration files to each slave node.

CHAPTER 4

CICS Runtime Configuration Files

Overview

The administration of CICS Runtime is based on Tuxedo native tools with the addition of a limited number of configuration tables for features that are specific to CICS. In CICS configurations, resources are nowadays defined in the CSD when previously they were defined as independent tables. This latter approach is the one used with CICS Runtime.

Each resource configuration table describes a resource of a particular type: transaction, transaction class, program, file, TS Queue model, etc. Each table contains the specific parameters relevant to the resource.

Shared Responsibilities Between Tuxedo and Resource Files

A CICS resource like a transaction with all its characteristics (first program, restartable, …) is described in resource configuration files. The Tuxedo configuration elements, like how many servers of which group on which machine will offer this transaction is described in the Tuxedo configuration file UBBCONFIG.

This way the responsibilities are clearly distributed:

- Configuration of the resources guaranties the functional behavior of a CICS application.
- Configuration of the Tuxedo system guaranties optimal performance and robustness in production.

Resource Definition Directory

All resource configuration files are stored in a common directory indicated by a well known environment variable: `${KIXCONFIG}`.

Each table describing CICS type of information is stored in a file read by servers at start time.

Presentation of Configuration Files

General Content

Each resource configuration table describes a resource type: transaction, transaction class, program, files, TS Queue, …, with all the specific parameters relevant to this resource.

The central file defines the lists of resource groups. When configuring a ART CICS server, the administrator specifies which lists to load. A single list may contain a few tens of resource groups that include hundreds or thousands of individual resources.

Structure

Each resource table contains three columns of parameters:

Field Name	Type	Values	Description
Name of the parameter in the table.	The data type of the field.	When specific values are required they are listed here.	Description of the purpose of the field, and its usage

The rest of this section describes in detail each of these configuration files:

- List of Groups Configuration File
- Transaction Configuration File.
- Tranclasses Configuration File.
- Programs Configuration File
- Files Configuration File
- TS Queue Model Configuration File.

- ENQ-Model Configuration File
- TD Queue Extra Partition Configuration File
- TD Queue Intra Partition Configuration File
- Typeterm Configuration File

List of Groups Configuration File

Table 4-1 defines the lists of resource groups that may be loaded by application servers.

The filename is `list_of_groups.desc`.

Table 4-1 Group List Parameters

Field Name	Type	Values	Description
`LIST`	`X(10)`	`Mandatory`	Name of the list. Referred by - L options of the application servers
`GROUP`	`X(10)`	`Mandatory`	Name of the group to be included in the list. This table contains one line per group to in a list. The same group may present in more than one list.

Transaction Configuration File

Table 4-2 lists the transactions available to application users, with their characteristics.

The filename is `transactions.desc`.

Table 4-2 Transaction Parameters

Field Name	Type	Values	Description
TRANSACTION	X(4)	Mandatory	Name of the transaction.
GROUP	X(10)	Mandatory	The group notion of CICS allowing a group of related resources to be declared and instantiated or not by a CICS system when starting.
DESCRIPTION	X(60)	Optional	A small textual comment zone for description of the resource.
PROGRAM	X(30)	Mandatory	Name of the first program to be called for this transaction.
ALIAS	X(4)	Optional	Used to define an alias for the transaction (usually lower case).
CMDSEC	Bool	N|Y	The ESM to be called for system programming requests. The default value is N.
CONFDATA	Bool	N|Y Optional	Reserved for future use. As in confidential data: specifies whether CICS is to suppress user data from CICS trace entries when the CONFDATA system initialization parameter specifies HIDETC. If the system initialization parameter specifies CONFDATA=SHOW, CONFDATA on the transaction definition is ignored. The default value is N.

Table 4-2 Transaction Parameters

Field Name	Type	Values	Description
PRIORITY	9(3)	1 | n Optional	Reserved for future use. Specifies the transaction priority. This 1-to 3-digit decimal value from 0 to 255 is used in establishing the overall transaction processing priority. (Transaction processing priority is equal to the sum of the terminal priority, transaction priority, and operator priority, not exceeding 255.) The higher the number, the higher the priority. The default value is 1.
RESSEC	Bool	N|Y	Specifies whether resource security checking is to be used for resources accessed by this transaction. The default value is N.
RESTART	Bool	N|Y Optional	Reserved for future use. Specifies whether the transaction restart facility is to be used to restart those tasks that terminate abnormally and are subsequently backed out by the dynamic transaction backout facility. The default value is N.
STATUS	X(10)	ENABLED| DISABLED	Specifies the transaction status. • ENABLED: Allows the transaction to be executed normally. • DISABLED: Prevents the transaction being executed. The default value is ENABLED.
TASKDATAKEY	X(5)	USER |CICS	Reserved for future use. The default value is USER.

Table 4-2 Transaction Parameters

Field Name	Type	Values	Description
TPNAME	X(64)	Optional	Reserved for future use. Specifies the name of the transaction that may be used by an APPC partner, if the 4-character length limitation of the TRANSACTION attribute is too restrictive. This name can be up to 64 characters in length.
TRACE	Bool	Y|N Optional	Reserved for future use. Specifies whether the activity of this transaction is to be traced. The default value is Y.
TRANCLASS	X(8)	Optional	Specifies the name of the transaction class to which the transaction belongs. Transactions belonging to a transaction class are subject to scheduling constraints before they are allowed to execute. See Tranclasses Configuration File for more information on the usage of this parameter on the target platform. A Transaction with no tranclass defined will have no other scheduling constraints than the number of servers offering it.
TWASIZE	Short int	Optional	Specifies the size (in bytes) of the transaction work area to be acquired for this transaction. Specify a 1-to 5-digit decimal value in the range 0 through 32767. The default value is 0.

Each transaction is advertised as an Oracle Tuxedo service by CICS Runtime servers, e.g. ARTSTRN, ARTATRN…. You can divide the transactions into different groups and assign the groups to different servers using option "-l", so that each server just advertise its own services.

Note: It is not recommended to define all transactions to one group, as it causes every service to be advertised by every server and results in enormous consumption of Oracle Tuxedo services.

Tranclasses Configuration File

Table 4-3 lists and defines tranclasses available to regulate parallel transactions activities.

The filename is `tranclasses.desc`.

Table 4-3 Transclass Parameters

Field Name	Type	Values	Description
TRANCLASS	X(8)	Mandatory	Name of the transaction class. A tranclass defines a category of transactions, which should not be running in parallel; probably because they use some resources in a non-serializable way.
GROUP	X(10)	Mandatory	The group notion of CICS allowing a group of related resources to be declared and instantiated or not by a CICS system when starting.
DESCRIPTION	X(60)	Optional	A small textual comment zone for description of the resource.
MAXACTIVE	short	0 - 999	Defines the degree of parallelism of execution. The only value for which we do a special processing is value 1, see below for more information.

Semantic Information

Native Source CICS Definition

Transactions that are defined as belonging to a transaction class are subject to scheduling constraints before they are allowed to execute. If transactions belonging to an active transaction class are already running, any new transactions are queued. Use the `MAXACTIVE` attribute to specify the maximum number of transactions that you want to run. To limit the size of the queue, you can use the `PURGETHRESH` attribute.

By putting your transactions into transaction classes, you can control how CICS dispatches tasks.

Mapping to Target Platform Concepts

On Tuxedo, the scheduling of transactions and the affectation of resources to groups of transactions is performed differently; it is the number of servers offering given transactions which manages the scheduling of transactions, and the relative amount of resources affected to a group of transactions.

The Special Case of MAXACTIVE 1

This case is very specific, because it impacts the functional characteristics of the application.

It ensures that two transactions of this class will never execute concurrently. It defines a mutually exclusive behavior that is preserved on the target platform to guarantee the correct behavior of the application.

A single server `ARTSTR1` will offer the transactions belonging to one TRANCLASS with `MAXACTIVE = 1`.

Programs Configuration File

Table 4-4 lists and defines programs available to be referenced either as first program of a transaction, or being invoked by EXEC CICS LINK and XCTL.

The filename is `programs.desc`.

Table 4-4 Programs Parameters

Field Name	Type	Values	Description
PROGRAM	X(30)	Mandatory	Name of the program.
GROUP	X(10)	Mandatory	The group notion of CICS allowing a group of related resources to be declared and instantiated or not by a CICS system when starting.
DESCRIPTION	X(60)	Optional	A small textual comment zone for description of the resource.
LANGUAGE	X(8)	COBOL | C	The language of the program, required to know how to communicate with it. Current release supports COBOL.

Table 4-4 Programs Parameters

Field Name	Type	Values	Description
EXECKEY	X(4)		Reserved for future use Concerns memory protection of CICS shared structures.
STATUS	X(10)	ENABLED| DISABLED	Specifies the program status. • ENABLED: Allows the program to execute normally. • DISABLED: Prevents the program being executed.
REMOTESYSTEM	X(4)	Optional	Specifies that the program is not offered locally but in a DPL server.
REMOTENAME	X(10)	Optional	Specifies for a DPL program the name of the program on the distant site. Useful only if the remote name is different from the local name.

Files Configuration File

Table 4-5 lists and defines files available to be referenced by the CICS application.

The filename is `files.desc`.

Table 4-5 Files Parameters

Field Name	Type	Values	Description
FILE	X(8)	Mandatory	Name of the file; logical name of the file used in EXEC CICS related to this file.
GROUP	X(10)	Mandatory	The group notion of CICS allowing a group of related resources to be declared and instantiated or not by a CICS system when starting.
DESCRIPTION	X(60)	Optional	A small textual comment zone for description of the resource.

Table 4-5 Files Parameters

Field Name	Type	Values	Description
DISPOSITION	X(5)	SHARE | OLD Optional	Reserved for future use. Specifies the disposition of this file. (shared or exclusive).
DSNAME	X(60)	Mandatory	Specifies the data set name (as known to the operating system) to be used for this file.
JOURNAL	X(20)	NO | journal Optional	Reserved for future use. Specifies whether you want automatic journaling for this file. The journalized data is in the format of the VSAM record and is used for user controlled journaling. The data to be journalized is identified by the JNLADD, JNLREAD, JNLSYNCREAD, JNLSYNCWRITE, and JNLUPDATE attributes. This Journal is for auditing.
KEYLENGTH	Short	Optional	Reserved for future use. Specifies the length in bytes of the logical key of records in remote files, and in coupling facility data tables that are specified with LOAD(NO). In the current release, we support neither remote (extra Tuxedo system) files, nor CFDT; In future we may support Remote file shipping and use this Key Length.
OPENTIME	X(8)	FIRSTREF |STARTUP Mandatory	Reserved for future use. Specifies when the file is opened.

Table 4-5 Files Parameters

Field Name	Type	Values	Description
READINTEG	X(12)	UNCOMMITTED| CONSISTENT| REPEATABLE Mandatory	Reserved for future use. Specifies the level of read integrity required for files defined with RLSACCESS(YES). You can use READINTEG to set a default level of read integrity for a file. This default is used by programs that do not specify one of the API read integrity options. On the target platform the exact semantic of the three levels of integrity may vary from exact CICS/VSAM semantic to another.
RECORDSIZE	Short	1 - 32767 Optional	Reserved for future use. Specifies the maximum length in bytes of records in a remote file or a coupling facility data table. The size specified can be in the range 1 through 32767. In the current release, we support neither remote (extra Tuxedo system) files, nor CFDT; In future we may support Remote file shipping and use this Record Size.
REMOTENAME	X(8)	Optional	Reserved for future use. Specifies the name of the file on the remote system.

Table 4-5 Files Parameters

Field Name	Type	Values	Description
REMOTESYSTEM	X(4)	Optional	Reserved for future use. On source, specifies the name of the connection that links the local system to the remote system where the file resides. On the target platform, will be used only in case of file shipping to another system, either another TUXEDO system or native CICS system.
STATUS	X(10)	ENABLED| DISABLED| UNENABLED Mandatory	Reserved for future use. Specifies the initial status of the file following a CICS initialization. UNENABLED allows for explicit EXEC CICS SET FILE OPEN.

Journaling Attributes

Table 4-6 lists the journaling attributes.

Table 4-6 Journaling Attributes

Field Name	Type	Values	Description
JNLADD	X(6)	NONE| BEFORE| AFTER| ALL Optional	Reserved for future use. Specifies if you want the add operations recorded on the journal nominated by the JOURNAL attribute. On the target platform the semantic is conserved with a simplification: for BEFORE/AFTER/ALL, a single record is logged.
JNLREAD	X(10)	NONE| UPDATEONLY| READONLY| ALL Optional	Reserved for future use. Specifies the read operations you want recorded on the journal nominated by the JOURNAL attribute. Possible values are: • ALL: Journal all read operations. • NONE: Do not journal read operations. • READONLY: Journal only READ ONLY operations (not READ UPDATE operations). • UPDATEONLY: Journal only READ UPDATE operations (not READ ONLY operations).
JNLSYNCREAD	Bool	N|Y Optional	Reserved for future use. Specifies whether you want the automatic journaling records, written for READ operations to the journal, to be synchronous.

Table 4-6 Journaling Attributes

Field Name	Type	Values	Description
JNLSYNCWRITE	Bool	Y|N Optional	Reserved for future use. Specifies whether you want the automatic journaling records, written for WRITE operations to the journal, to be synchronous.
JNLUPDATE	Bool	N|Y Optional	Reserved for future use. Specifies whether you want REWRITE and DELETE operations recorded on the journal.

TS Queue Model Configuration File

Table 4-7 lists and defines theTS Queue models available to be referenced by the CICS application.

The filename is `tsqmodel.desc`

Table 4-7 TS Queue Parameters

Field Name	Type	Values	Description
TSMODEL	X(8)	Mandatory	Name of the TS Queue model.
GROUP	X(10)	Mandatory	The group notion of CICS allowing a group of related resources to be declared and instantiated or not by a CICS system when starting.
DESCRIPTION	X(60)	Optional	A small textual zone for description of the resource.
LOCATION	X(9)	AUXILIARY | MAIN	Specifies the kind of storage to use: file or memory. Only used as a control: MAIN TS cannot be recoverable.

Table 4-7 TS Queue Parameters

Field Name	Type	Values	Description
PREFIX XPREFIX	X(16)	Mandatory	Specifies the character string that is to be used as the prefix for this model. The prefix may be up to 16 characters in length.
RECOVERY	Bool	N|Y	Specifies whether or not queues matching this model are to be recoverable. On the target platform, a default queue is written to file, while a recoverable queue is stored in the RDBMS to provide recovery capabilities.
POOLNAME	X(8)	Optional	Specifies the 8-character name of the shared TS pool definition that you want to use with this TSMODEL definition. Will probably disappear in the future releases since there are other ways on target to arrive to the same result.
REMOTE_SYSTEM	X(4)	Optional	On source platform, specifies the name of the connection that links the local system to the remote system where the temporary storage queue resides. On the target platform, used only in case of TS shipping to another system, either another TUXEDO system or native CICS system.

Table 4-7 TS Queue Parameters

Field Name	Type	Values	Description
REMOTEPREFIX XREMOTEPREFIX	X(16)	Optional	Specifies the character string that is to be used as the prefix on the remote system. The prefix may be up to 16 characters in length. These options are useful (on source and target platforms) only if one wants to translate queue name when shipping TS Queue access from one system to another.
SECURITY	Bool	N|Y	Specifies whether security checking is to be performed for queues matching this model.

ENQ-Model Configuration File

Table 4-8 lists and defines ENQ Models available to be referenced by the CICS application.

The filename is `enqmodel.desc`.

Table 4-8 ENQ Model Parameters

Field Name	Type	Values	Description
ENQMODEL	X(8)	Mandatory	Name of the ENQ model.
GROUP	X(10)	Mandatory	The group notion of CICS allowing a group of related resources to be declared and instantiated or not by a CICS system when starting.
DESCRIPTION	X(60)	Optional	A small textual zone for description of the resource.
ENQNAME	X(255)	Mandatory	Specifies the 1 to 255-character resource name.

Table 4-8 ENQ Model Parameters

Field Name	Type	Values	Description
ENQSCOPE	X(4)	Optional	If omitted or specified as blanks, matching enqueue models will have a local scope, else they will have a global scope
STATUS	bool	E| D	E = Enabled D = Disabled.

TD Queue Extra Partition Configuration File

Table 4-9 lists and defines extra partitions TD queues available to the CICS application.

The filename is `tdqextra.desc`.

Table 4-9 TD Queue Parameters

Field Name	Type	Values	Description
TDQUEUE	X(4)	Mandatory	Specifies the 1- to 4-character name of a transient data queue.
GROUP	X(10)	Mandatory	The group notion of CICS allowing a group of related resources to be declared and instantiated or not by a CICS system when starting.
DESCRIPTION	X(60)	Optional	A small textual zone for description of the resource.
DDNAME	X(8)	Mandatory	Specifies a 1-to 8-character value that may refer to a data set defined in the startup JCL.

Table 4-9 TD Queue Parameters

Field Name	Type	Values	Description
DISPOSITION	X(3)	Optional	Specifies the disposition of the data set. • MOD: ART-CICS first assumes that the data set exists. For an existing sequential data set, MOD causes the read/write mechanism to be positioned after the last record in the data set. The read/write mechanism is positioned after the last record each time the data set is opened for output. If ART-CICS cannot find volume information for the data set: – On the DD statement. A data set allocated dynamically in this way is deleted when the queue is closed, and all records are lost. For a new data set, MOD causes the read/write mechanism to be positioned at the beginning of the data set. • OLD: The data set existed before this job step. • SHR: The data set existed before this job step and can be read by other concurrent jobs.
ERRORPTION	X(1)	I | S	**UNSUPPORTED** Specifies the action to be taken if an I/O error occurs. This can be one of the following: • I = IGNORE: The block that caused the error is accepted. • S = SKIP: The block that caused the error is skipped.

Table 4-9 TD Queue Parameters

Field Name	Type	Values	Description
OPENTIME	X(1)	D | I	**UNSUPPORTED** Specifies the initial status of the data set. The initial status can be one of the following: • D = DEFERRED: The data set remains closed until you indicate that you want to open it by using the CEMT INQUIRE|SET TDQUEUE command. • I = INITIAL: The data set is to be opened at install time. However, if the DSNAME attribute is not specified, and the data set name is not specified in the DD statement in the startup JCL, the transient data queue is allocated to JES during CICS startup.
RECORDFORMAT	X(1)	F | V	Specifies the record format of the data set. • F= FIXED: Fixed records. If you specify RECORDFORMAT FIXED, you must also specify a block format. • V= VARIABLE: Variable records. If you specify RECORDFORMAT VARIABLE you must also specify a block format.
PRINTCONTROL	X(1)	A |	**UNSUPPORTED** Specifies the control characters to be used. There is no default. If you allow RECORDFORMAT to default to blank, you cannot specify anything in the PRINTCONTROL field. The control characters that can be used are: • A= ASA: ASA control characters. • blank: No control characters are to be used.
RECORDSIZE	9(4) COMP	Optional	Specifies the record length in bytes, in the range 0 through 32767.
TYPEFILE	X(6)	Optional	Specifies the type of data set the queue is to be associated with: • INPUT: An input data set. • OUTPUT: An output data set.

Table 4-9 TD Queue Parameters

Field Name	Type	Values	Description
DSNAME	X(80)	Optional	Specifies the name of the file that is to be used to store records written to this extrapartition queue. This file must exist even if empty.
SYSOUTCLASS	X(1)	Optional	**UNSUPPORTED** Instead of allocating an extra partition queue to a physical data set, you can allocate it to a system output data set (referred to as SYSOUT). Use the SYSOUT CLASS attribute to specify the class of the SYSOUT data set. A..Z|0..9 A single alphabetic or numeric character that represents an output class that has been set up on the z/OS system on which the CICS Runtime job is to run.
TRT	X(1)	S | I	New optional CICS Runtime argument, allowing integrators and customers to make their own specific implementation of extra partition queues. No value or S (for Standard) will invoke normal CICS Runtime TDQueue functionalities Setting the value I, will trigger the call to a function `td_extra_actions_int`, which must be provided by the integrator.

TD Queue Intra Partition Configuration File

Table 4-10 lists and defines intra partitions TD queues available to the CICS application.

The filename is `tdqintra.desc`.

Table 4-10 TD Queue Parameters

Field Name	Type	Values	Description
TDQUEUE	X(4)	Mandatory	Specifies the 1- to 4-character name of a transient data queue.
GROUP	X(10)	Mandatory	The group notion of CICS allowing a group of related resources to be declared and instantiated or not by a CICS system when starting.
DESCRIPTION	X(60)	Optional	A small textual zone for description of the resource.
RECOVSTATUS	X(8)	NO | LOGICAL	Specifies if the queue is logically recoverable or not. If a queue is logically recoverable, its elements will be written to tuxedo /Q in the context of the transaction and will be rolled back with the rest of the transaction in case of a rollback. If the queue is non-recoverable, then each enqueue in the /Q queue will be permanent and not rolled back in case of a rollback or abort.
TRANSID	X(4)	Optional	Specifies the name of the transaction that is to be automatically initiated when the trigger level is reached. Transactions are initiated in this way to read records from the queue. If the TRANSID attribute is not specified (or if TRIGGERLEVEL(0) is specified), you must use another method to schedule transactions to read records from transient data queues.

Table 4-10 TD Queue Parameters

Field Name	Type	Values	Description
TRIGGERLEVEL	X(1)	1 | n	Specifies the number of records to be accumulated before a task is automatically initiated to process them. (This number is known as the trigger level.) If you specify the TRANSID attribute, TRIGGERLEVEL defaults to 1. Specify a trigger level of 0 if you want to disable ATI processing. If you do not specify a transaction ID, the trigger level is ignored. For logically recoverable transient data queues, the ATI task is not attached until the task commits forward. This may mean that the trigger level is far exceeded before ATI occurs.
USERID	X(8)	optional	Specifies the userid you want CICS to use for security checking when verifying the trigger-level transaction specified in the TRANSID field.
WAIT	X(1)	YES | NO	**INACTIVE field** accepted only in the resource loading
WAITACTION	X(6)	REJECT | QUEUE	**INACTIVE field** Accepted only in the resource loading.
QSPACENAME	X(15)	mandatory	New mandatory CICS Runtime argument, specifying the name of the tuxedo /Q QSPACE into which this queue is physically stored. Consult your Tuxedo /Q documentation for more information on qspaces and queue administration.
TRT	X(1)	S | I	New optional CICS Runtime argument, allowing integrators and customers to make their own specific implementation of intra-partition queues. No value or S (for Standard) will invoke normal CICS Runtime TSQueue functionalities Setting the value I, will trigger the call to a function `td_intra_actions_int`, which must be provided by the integrator.

Mapset Configuration File

Table 4-11 lists and defines mapsets available to be referenced by the CICS application. For more information, see `tcxmapgen(1)` in System Commands and Transactions.

The filename is `mapsets.desc`.

The format of a `MAPSET` definition is:

```
[mapset]
<field_name_1>=<field_value_1>
<field_name_2>=<field_value_2>
… …
<field_name_n>=<field_value_n>
```

For example,

```
[mapset]
name=ABANNER
filename=abanner.mpdef
```

Table 4-11 Mapset Parameters

Field Name	Type	Values	Description
NAME	X(8)	Mandatory	Name of the mapset.
DESCRIPTION	X(60)	Optional	A small textual comment zone for description of the resource.
FILENAME	X(79)	Mandatory	This specifies the physical (binary) file name of the mapset, which is generated by the tcxmapgen tool. It will be searched in directories defined by the KIX_MAP_PATH environment variable if the absolute path is not specified. If this field is not specified, the default mapset binary file name <MAPSET_name>.mpdef will be used, in which the <MAPSET_name> is the MAPSET name parameter specified in CICS MAP related APIs

Field Name	Type	Values	Description
RESIDENT	Bool	NO | YES	Specifies the residence status of the map set. • NO: The map set is not to be permanently resident. • YES: The map set is to be loaded on first reference and is then to be permanently resident in virtual storage, but is to be pageable by the system.
swastatus	X(10)	ENABLED| DISABLED	Specifies the resource status.•If set to ENABLED, the resource is available.•If set to DISABLED, the resource is unavailable for use by the system.
Usage	X(10)	NORMAL | TRANSIEN T	This attribute specifies the caching scheme to be used once the MAPSET is loaded. NORMAL keeps the MAPSET loaded in a cache. Unload it when the cache overflows and it is the oldest, least used MAPSET in the cache.TRANSIENT unloads the MAPSET if it is not being used.

Typeterm Configuration File

Table 4-12 lists and defines Typeterms supported by ARTTCP.

The filename is `typeterms.desc`

The format of a TYPETERM definition is:

```
[typeterm]
<field_name_1>=<field_value_1>
<field_name_2>=<field_value_2>
… …
<field_name_n>=<field_value_n>
```

For example,

```
[typeterm]
name=IBM-3278-2
userarealen=255
```

Table 4-12 Typeterm Parameters

Field Name	Type	Values	Description
ALTSCREENCOLUMN	Short	{80|132|…}	Specifies the terminal screen size total columns. If the SCRNSIZE=alternate, this parameter is mandatory.
ALTSCREENROW	Short	{24|32|43|27|…}	Specifies the terminal screen size total rows If the SCRNSIZE=alternate, this parameter is mandatory.
DESCRIPTION	X(79)	Optional	A small textual zone for description of the resource.
EXTERCODE	X(10)	{ibm-37 | ibm-1388 |ibm-1380|…}	Specifies which encoding type of outbound data is used. The value of this attribute could be any EBCDIC encoding type used in z/OS platform. The default value is ibm-37.

Table 4-12 Typeterm Parameters

Field Name	Type	Values	Description
INTERCODE	X(10)	{ASCII | UTF-8|Shift-JIS| …}	Specifies which encoding type of inbound data is used. The value of this attribute could be any encoding type used in universal platform. The default value is ASCII.
NAME	X(79)	Mandatory	Name of the typeterm.
PROGSYMBOLS	Bool	NO | YES	Specifies whether the programmed symbol (PS) facility is supported or not. The default value is NO.
SCRNSIZE	Bool	DEFAULT| ALTERNATE	Optional. Specifies whether to send/receive map/text with alternative screen size or not. The default value is DEFAULT which does not send/receive map/text with alternative screen size.
SOSI	Bool	NO | YES	Specifies whether mixed EBCDIC and double-byte character set (DBCS) is supported or not. The default value is NO.
color	Bool	NO | YES	Designates extended color attributes.
defscreencolumn	Short	80	Number of columns of the default screen size.
defscreenrow	Short	24	Number of rows of the default screen size.
hilight	Bool	NO | YES	Indicates whether a terminal supports the highlight feature or not.
logonmsg	Bool	NO | YES	Indicates whether the “Good Morning” (CSGM) transaction is automatically started on the terminal or not. Oracle Tuxedo ART provides a default CSGM transaction. Please refer to section for the configuration of the default “Good Morning” (CSGM) transaction.
outline	Bool	NO | YES	Indicates whether the terminal supports field outlining or not.

Table 4-12 Typeterm Parameters

Field Name	Type	Values	Description
swastatus	X(10)	ENABLED| DISABLED	Specifies the resource status. • If set to ENABLED, the resource is available. • If set to DISABLED, the resource is unavailable for use by the system
uctran	X(10)	NO | YES | TRAN	• YES: translate lowercase alphabetic characters to uppercase. • NO: do not translate lowercase alphabetic characters to uppercase • TRAN: only translate the transaction ID from lowercase to uppercase.
userarealen	Short	0 ~ 255	The terminal control table user area (TCTUA) area size for the terminal.

Environment Variables

CICS Runtime Environment Variables

Two important Tuxedo environment variables are MANDATORY.

- TUXDIR– must be set to indicate the directory in which Tuxedo is installed.
- APPDIR – must be set to indicate the directory where the application server binaries are installed.

Note: For CICS Runtime, APPDIR must be set to the directory containing the CICS Runtime server binaries.

CICS Runtime Specific Environment Variables

KIXDIR

KIXDIR is a mandatory environment variable that indicates the directory where the CICS Runtime product is installed.

Usually, the Tuxedo environment variable APPDIR should be set to ${KIXDIR}/bin

KIXCONFIG

KIXCONFIG is a mandatory environment variable that indicates the directory where resource configuration files are located.

KIX_TS_DIR

KIX_TS_DIR is a mandatory environment variable that indicates the directory where files corresponding to non-recoverable TS are located. It can be differentiated for each tsq server by setting it differently in the server envfile (see the Tuxedo documentation).

KIX_TD_DIR

KIX_TD_DIR is a mandatory environment variable that indicates the directory where files corresponding to the extra partition TDQueues are located.

KIX_TD_QSPACE_DEVICE

KIX_TD_QSPACE_DEVICE is a mandatory environment variable for TD QUEUE(INTRA). It indicates the Tuxedo QSPACE needed by the tdq_srv server.

KIX_TD_QSPACE_NAME

KIX_TD_QSPACE_NAME is a mandatory environment variable for TD QUEUE(INTRA). It indicates the Tuxedo QSPACE name needed by the tdq server.

KIX_TD_QSPACE_IPCKEY

KIX_TD_QSPACE_IPCKEY is a mandatory environment variable for TD QUEUE(INTRA). It indicates the Tuxedo QSPACE ipckey needed by the tdq server.

KIX_TECH_DIR

KIX_TECH_DIR is a mandatory environment variable that indicates the directory where technical files used internally by ART CICS, for exemple to manage named DELAYs and CANCELs (thru the REQID option) or ENQ/DEQ are written. It should be the same for each server until one wants to reproduce the source limitation, where a named DELAY submitted on one CICS region, could not be canceled easily in another region.

KIX_CWA_SIZE

This environment variable is optional.

On the source platform the Common Work Area (CWA) is shared by all the Programs executing inside a single CICS Region. The size of this CICS zone can vary from 0 to 32765 bytes, 0 indicating that no CWA is defined.

On the target platform, the KIX_CWA_SIZE variable also indicates the size of the CWA, ranging from 0 to 32765 bytes. If this environment variable is not set, the value defaults to 0. A value of zero (either explicit or implicit) indicates that no CWA is defined.

KIX_CWA_IPCKEY

The Common Work Area (CWA), when defined (see KIX_CWA_SIZE), is implemented on each machine by a shared memory segment. The KIX_CWA_IPCKEY variable indicates the IPCKEY (the identifier) of the shared memory segment. The value must be defined in the range from 1 to 99 999 999.

Note: This variable is mandatory when KIX_CWA_SIZE is set to a value greater than zero.

KIX_QSPACE_IPCKEY

This mandatory variable is used to create the Tuxedo qspace named ASYNC_QSPACE utilized by ARTATRN for delayed asynchronous transactions.

The value for the IPC key should be picked so as not to conflict with your other requirements for IPC resources. It should be a value greater than 32 768 and less than 262 143.

KIX_TRACE_LEVEL

This optional variable allows the administrator to get traces for the system activities.

It can be set from 0 to 9, 0 represents no trace, 9 represents maximum trace. The default value is 0 when the variable is not defined. The most relevant trace levels are:

- 0 – servers startup and fatal errors information
- 2 – loading resources and advertising services information
- 3 – informations exchanged with the termnal
- 5 – SPOOL (CICS SPOOL) function traces
- 7 – CICS function traces (KIX__)
- 9 – traces for support team

Other levels are reserved. The higher level covers all traces deliverred by lower level.

KIX_MAP_PATH

This optional variable defines the path (the list of directories) in which the physical file of the mapset will be searched, in case the absolute path is not specified in the `FILENAME` field of Mapset in the Typeterm configuration file.

KIX_SPOOL_OUTPUT_DIR

This is a mandatory environment variable used for SPOOL functions. It indicates the directory where the CICS Runtime writes the spool files named
`<spool_token>.<sever_pid>.<time_in_microseconds>.<occurrence_number>`

KIX_SPOOL_JOB_SUBMIT

This is a mandatory environment variable used for SPOOL functions. It indicates the command line for the spool files submission. The command line should contain the first mandatory %s symbol that refers to the spool file name and the second facultative %s symbol that refers to the CLASS.

For example: `KIX_SPOOL_JOB_SUBMIT=/my_path/my_shell_script -f %s -c %s`

Note: This script should be run in the batch execution environment. You can use “nohup” and “&” command to keep running the script in the background after you have logged out.

COB_ENABLE_XA

In release 11.1.1.3.0, this is a mandatory environment variable when using COBOL-IT with ART CICS Runtime. It indicates VSAM file support with COBOL-IT/BDB under XA environment is enabled. It should be set to 1.

Server Configuration

CICS Runtime Servers References

About Generic Tuxedo Server Configuration

All Tuxedo servers configured in the Tuxedo UBBCONFIG configuration file use standard arguments common to all servers. CICS Runtime servers benefit automatically from this flexibility.

The required arguments are `SVRGRP` and `SVRID`.

Other common arguments like `MIN`, `MAX`, `SEQUENCE`, `CONV` etc. are also available.

For precise information about the use of Tuxedo server configuration, consult the Tuxedo documentation, specifically the SERVERS section of UBBCONFIG(5).

One of the most useful of these optional arguments is the `CLOPT` (Command Line OPTions) argument.The `CLOPT` option is a string of command-line options that is passed to Tuxedo servers when they are booted.

This command line option is divided in two parts:

- A generic part, common to every Tuxedo server, common server options are:

```
[-e stderr_file] [-o stdout_file]
```

directing standard output and errors to specific files.

- A server specific part containing options referenced as uargs in Tuxedo documentation.

For precise information about using CLOPT options see the Tuxedo documentation, more specifically the servopts section.

The description of CICS Runtime specific servers systematically includes the two mandatory server arguments SVRGRP & SVRID, plus only the arguments needed specifically by the server type.

Generic CLOPT Options of CICS Runtime Servers

This section describes the options common to all CICS Runtime servers. These options are documented in this section only.

CICS SYSID Argument

This argument defines the name of the CICS system.

Synopsis

```
-s TEST
```

Description

Sets the value returned to programs by EXEC CICS ASSIGN SYSID.

Character, 1-256, A-Za-z0-9[/:-].

The system identifier (CICS SYSID) is limited to four characters.

Exclusion

This option does not apply to ARTTCPL servers and connection servers.

CICS Application ID Argument

This argument defines the APPLID name of the CICS system.

Synopsis

```
-a INVOICE
```

Description

Sets the value returned to programs by EXEC CICS ASSIGN APPLID.

Character, 1-256, A-Za-z0-9[/:-].

The application id (CICS APPLID) is limited to eight characters.

Exclusion

This option does not apply to ARTTCPL servers and connection servers.

Dynamic List of Groups Argument

This argument defines the lists of resource groups to be loaded by this server.

Synopsis

```
-L LIST1:LIST2:…
```

Description

Enables a dynamic change of the groups in a list for a running server. The lists refered by the -L argument should be defined in the `list_of_groups` configuration file. This argument replaces the -l option which is deprecated now.

Lists in the resources configuration files are defined by 10 character strings. A server only loads in memory resources belonging to one of the groups included in one of the lists.

As a facility for tests or generic servers, it is possible to remove the filtering by using -L ‘*’ to allow a server to load all the lists defined in the `list_of_groups` configuration file. A group can be loaded by a server specifying -L ‘*’ only if it is included in at least one list.

Exclusion

This option does not apply to ARTTCPL servers and connection servers.

Static List of Groups Argument

This argument is now deprecated and replaced by -L argument. It is still supported in this release but will be removed in future releases.

This argument lists the resource groups to be considered by the server when loading resources.

The list of groups defined statically in the CLOPTcannot be dynamically modified. For implementing a dynamic change of the list, use -L option instead.

Synopsis

```
-l group1:group2:…:groupn
```

Description

Groups in the resources configuration files are defined by 10 character strings. A server only loads in memory resources belonging to one of these groups.

As a facility for tests or generic servers, it is possible to remove the filtering by using –l '*' to allow a server to load all the resources defined in the configuration file.

Exclusion

This option does not apply to ARTTCPL servers and connection servers.

Configuration Reference of CICS Runtime Servers

ARTTCPL/ARTTCPH Configuration

Server Name

ARTTCPL – Terminal Control Program Listener.

Synopsis

```
ARTTCPL SRVGRP="identifier" SRVID="number" MAXWSCLIENTS="number"
CLOPT="[servopts options] -- -n netaddr -S ssladdr -L pnetaddr [-m minh] [-M
maxh] [-x session-per-handler] [-p profile-name] [-z minencryptbits] [-Z
maxencryptbits] [-D] [+H trace-level]"
```

Description

The terminal control program (ARTTCP) is a group of Tuxedo servers that manage the connections of 3270 terminal emulators to CICS Runtime. When you run programs, the ARTTCP connects terminal emulators to the network ports assigned to ARTTCP. ARTTCP communicates with the emulator using a Telnet protocol.

The ARTTCP server is composed of two types of servers: a single ARTTCP listener (ARTTCPL) process and one or more ARTTCP handler (ARTTCPH) processes. The ARTTCPL process establishes a well-known listening port address to which terminal emulators may connect. The ARTTCPH process listens on this port and accepts incoming connection requests. The ARTTCPH process establishes your user session for the connection and handles all subsequent screen I/O for the terminal emulator. As a performance enhancement, each ARTTCPH process can manage multiple

sessions simultaneously. When you disconnect the emulator from the port, the ARTTCPH terminates the session.

Parameters

The following CLOPT run-time parameters are recognized:

-n netaddr

This address specifies where TN3270 terminal emulators connect to ARTTCPL. The address is a string in standard internet URL format. For example:

```
//computer:4000 designates port 4000 on machine computer.
```

Character, 1-256, A-Za-z0-9[/:-]. Mandatory option if option -S is not specified.

-S ssladdr

This address specifies where TN3270 terminal emulators connect to ARTTCPL via SSL. The address is a string in standard internet URL format. For example:

```
//computer:5000 designates port 5000 on machine computer.
```

Character, 1-256, A-Za-z0-9[/:-]. Mandatory option if option -n is not specified.

ARTTCPL shares the same SSL related configuration with Tuxedo, so the following attributes should be configured in the RESOURCES section of Tuxedo UBBCONFIG configuration file: SEC_PRINCIPAL_NAME, SEC_PRINCIPAL_LOCATION, SEC_PRINCIPAL_PASSVAR. Please refer to the corresponding Tuxedo documentation for details.

-L pnetaddr

This address is used by the system internally between ARTTCPL and ARTTCPH. The address is a string in standard internet URL format. For example:

```
//computer1:4001 designates port 4000 on machine computer.
```

Character, 1-256, A-Za-z0-9[/:-]. Mandatory option.

[-m minh]

The minimum number of handler processes that will be started by ARTTCPL. The actual number of handler processes will always be between the minh and maxh based on system load.

Numeric, 1-4096. Default value is 1.

[-M maxh]

The maximum number of handler processes that will be started by ARTTCPL. The actual number of handler processes will always be between the minh and maxh based on system load.

Numeric, 1-4096. Default value is 4096.

[-x session-per-handler]

The number of sessions a ARTTCPH can maintain concurrently.

Numeric, 1-255. Default value is 32.

[-p profile-name]

The default security profile file name. Please refer to Security configuration for details.

String. The default value is ~/.tuxAppProfile.

[-z minencryptbits]

The minimum level of encryption required when a network link is being established between a TN3270 terminal emulator and ARTTCP. 0 means no encryption, while 40, 56, 128, and 256 specify the length (in bits) of the encryption key. If this minimum level of encryption cannot be met, link establishment fails.

Numeric. Default value is 0. This option will be ignored if -S option is not specified.

[-Z maxencryptbits]

The maximum level of encryption required when a network link is being established between a TN3270 terminal emulator and ARTTCP. 0 means no encryption, while 40, 56, 128, and 256 specify the length (in bits) of the encryption key.

Numeric. Default value is 256. This option will be ignored if -S option is not specified.

[-D]

Enable Debug.

[+H trace-level]

Specify the trace level:

-1: trace off.

0: trace for all ARTTCPH.

n (n>0): trace the first n ARTTCPH.

Examples

```
*SERVERS ARTTCPL SRVGRP="TCPGRP" SRVID=1000 RESTART=Y GRACE=0
MAXWSCLIENTS=20

CLOPT="-- -n //hostname:4000 -L //hostname:4002 -m1 -M10 "
```

ARTSTRN Configuration

Server Name

ARTSTRN – CICS Runtime main server for synchronous terminal oriented transactions with `MAXACTIVE > 1`.

Synopsis

```
ARTSTRN SRVGRP="identifier" SRVID="number" CONV=Y MIN=minn MAX=maxn
RQADDR=queueaddr REPLYQ=Y CLOPT="[servopts] -- -s System_ID -a
Application_ID -L list1:list2"
```

Description

`ARTSTRN` servers present application transactions as Tuxedo services and, when receiving a transaction request, execute the corresponding programs.

These servers are conversational in order to manage true conversational CICS transactions.

1. When starting, an `ARTSTRN` server publishes one service per transaction it offers.
2. When a user transmits a transaction request, the `ARTTCPH` managing the user performs a tpconnect to the corresponding transaction (service).
3. One `ARTSTRN` server offering this service receives the request with the associated commarea and screen and then processes the transaction.
4. After processing the transaction, the ARTSTRN server:

- In the case of a Normal Pseudo-Conversational CICS transaction: On the RETURN `{TRANSID}` a reply is sent to the client, finishing the conversation by a `tpreturn()` returning the new 3270 screen, and the commarea.
- In the case of a Conversational CICS transaction with loop of SEND & RECEIVE:

- On the RECEIVE the ARTSTRN server transmits the prepared 3270 stream via tpsend(), then waits for a tpreceive, for the next user input to complete the RECEIVE
- On the RETURN {TRANSID} the ARTSTRN server replies to the client, finishing the conversation by a tpreturn() returning the new 3270 screen, and the commarea.

Only transactions belonging to no tranclass, or to a tranclass with maxactive >1 are advertised by these servers.

Parameters

CONV

The generic parameter CONV is mandatory for this server type, and must be defined as CONV=Y, because ARTSTRN is non-transactional.

minn and maxn

Specify respectively the initial and maximum number of servers with this configuration to start. For more information see the UBBCONFIG section of the Tuxedo documentation.

CLOPT options

The following CLOPT run-time parameters are recognized:

-s SystemID

Mandatory argument, see CICS SYSID Argument.

-l GroupList

Mandatory option, see Dynamic List of Groups Argument.

-a Application_ID

Optional argument, see CICS Application ID Argument.

-L List_name(s)

Mandatory argument, see Dynamic List of Groups Argument.

Environment Variables Used

- KIXCONFIG
- KIX_CWA_IPCKEY
- KIX_TRACE_LEVEL
- KIX_MAP_PATH

- KIX_TECH_DIR

Examples

```
*SERVERS

ARTSTRN SRVGRP="TCPGRP" SRVID=1000 RESTART=Y GRACE=0

CONV=Y MIN=2 MAX=3 RQADDR=QKIX1000 REPLYQ=Y

CLOPT=" -- -s PROW -a INVOICE -L list1:list2"
```

ARTSTR1 Configuration

Server Name

ARTSTR1 – CICS Runtime main server for synchronous terminal oriented transactions with `MAXACTIVE = 1`.

Synopsis

```
ARTSTR1 SRVGRP="identifier" SRVID="number" CONV=Y MIN=1 MAX=1
CLOPT="[servopts] -- -s System_ID -a Application_ID-L list1:list2,…"
```

Description

Theses servers are a specialized version of `ARTSTRN` servers presenting only transactions with `MAXACTIVE = 1`; While `ARTSTRN` servers present only transactions with `MAXACTIVE > 1`.

It is critical and verified by STR1 servers at boot time that `MIN` and `MAX` number of servers are set to 1. The goal of these servers being to guarantee the parallel processing of only one transaction in a group (with `MAXACTIVE = 1`), to start or let Tuxedo start a few servers offering the same transactions will be self-defeating for STR1 Servers.

Since `MIN` and `MAX` are set to 1 the Tuxedo argument RQADDR, become unnecessary, and should be avoided for simplicity.

The rest of the configuration and behavior of STR1 servers are exactly the same a STRN servers.

Examples

```
*SERVERS

ARTSTR1 SRVGRP="TCPGRP" SRVID=1000 RESTART=Y GRACE=0

CONV=Y MIN=1 MAX=1
```

```
CLOPT=" -- -s PROW -a INVOICE -L list1:list2"
```

ARTTSQ Configuration

Server Name

ARTTSQ – CICS Runtime Temporary Storage Queue Server

Synopsis

```
ARTTSQ SRVGRP="identifier" SRVID="number" MIN=1 MAX=1
        CLOPT"[servopts] -- -L list1:list2"
```

Description

ARTTSQr manages temporary storage queues, it serves the functionalities required by EXEC CICS: WRITEQ TS, READQ TS and DELETEQ TS.

ARTTSQ publishes two main kinds of services:

- TSQUEUE: This service is published only once when the first ARTTSQ starts. TSQUEUE processes TSQ requests for queues matching no TSMODEL.
- {TSMODEL}_TSQUEUE: One of those services is published for each TSMODEL.

  The server publishing this service will accomplish all the operations needed on the queues matching this TSMODEL.

  One server will publish the TSMODELs belonging to the resource groups assigned to this server thru the -l option.

  A group of resources must be assigned to a single tsq serverto avoid trying to publish the same service twice. This is checked at boot time and will generate error messages during the boot phase when not respected, but no action will be taken.

  It is critical, and verified by TSQ servers at boot time, that MIN and MAX number of servers are set to 1.

  It is critical that the same server which created one queue (first write) also serves all other read/write delete requests to this queue. This is the reason why each service, either generic or corresponding to a specific model, must be advertised by a single server.

  This unicity is verified when services are published.

Parameters

ARTTSQ

The following CLOPT run-time parameters are recognized:

-L ListName(s)

Mandatory argument, see Dynamic List of Groups Argument for more information.

DBMS Constraints

SRVGRP must be a Tuxedo group with an Oracle Resource Manager configured with TMSNAME and OPENINFO.

The DBMS user indicated in the OPENINFO of the group containing the server, must have access to the TS_QCONTENT table; either directly (objects created in this schema) or thru a DBLINK.

On this pre-existing table it must have select, insert, update, delete permissions.

The script to create the table for Oracle is listed below:

Listing 6-1 TS_QCONTENT Creation

```
drop table TS_Q_CONTENT purge;
create table TS_Q_CONTENT
( TS_QUEUE   char(16) NOT NULL,
  TS_ITEM    number(8) NOT NULL,
  TS_LENGTH  number(8),
  TS_RAW     LONG RAW,
  primary key (TS_QUEUE, TS_ITEM)
);
```

Environment Variables Used

- KIXCONFIG
- KIX_TS_DIR

- KIX_TRACE_LEVEL
- KIX_MAP_PATH
- KIX_TECH_DIR

Examples

```
*SERVERS

ARTTSQ SRVGRP="GRP02" SRVID=30 RESTART=Y GRACE=0

        MIN=1 MAX=1 CLOPT=" -- -L list1:list2"
```

ARTTDQ Configuration

Server Name

ARTTDQ – CICS Runtime Transient Data Queue Server

Synopsis

```
ARTTDQ SRVGRP="identifier" SRVID="number" MIN=1 MAX=1

        CLOPT"[servopts] -- -L list1:list2:..."
```

Description

`ARTTDQ` manages transient data storage queues, it serves the functionalities required by EXEC CICS: WRITEQ TD, READQ TD and DELETEQ TD.

`ARTTDQ` publishes one service per declared queue as the name of the TDQueue suffixed by "_TDQ":

A group of resources must be assigned to a single `ARTTDQ` server to avoid trying to publish the same service twice. This is checked at boot time and will generate error messages during the boot phase when not respected, but no action will be taken.

It is critical, and verified by TDQ server at boot time, that MIN and MAX number of servers are set to 1.

Parameters

ARTTDQ

The following `CLOPT` run-time parameters are recognized:

-L ListName(s)

Mandatory argument, see Dynamic List of Groups Argument for more information.

Environment Variables Used

- KIXCONFIG
- KIX_TS_DIR
- KIX_TRACE_LEVEL
- KIX_TECH_DIR

Examples

```
*SERVERS
ARTTDQ SRVGRP="GRP02" SRVID=30 RESTART=Y GRACE=0
        MIN=1 MAX=1 CLOPT=" -- -s PROW -l group1:group2"
```

ARTDPL Configuration

Server Name

`ARTDPL` – CICS Runtime server for distributed program link execution.

Synopsis

```
ARTDPL SRVGRP="identifier" SRVID="number" MIN=minn MAX=maxn
       CLOPT="[servopts] -- -s System_ID -a Application_ID -L list1:list2"
```

Description

Theses servers present application programs restricted to DPL subsets as tuxedo services and when receiving a DPL service request execute the corresponding program.

Theses servers do not need to (cannot) address the principal facility (the user terminal) and so do not need to be conversational. They are pure RPC mode servers.

When starting, a `ARTDPL` publishes one service per program it offers.

When a program requests a LINK, if the requested program is configured as DPL then the link is not resolved as usual by a call, but by a `tpcall`, which will be served by one of the DPL servers offering this service (this DPL program).

Only programs with the attribute `REMOTESYSTEM(sysid)` positioned to DPL, will be advertised by DPL servers, and only by servers with this sysid as system indicated thru the `-s` option

The service advertised by `ARTDPL` for each of these programs, will be `SYSID_ProgramName`.

Conversely, these programs will not be available directly from `synchronous and asynchronous transcation` servers.

Parameters

minn and maxn

Specify respectively the initial and maximum number of servers to start. For more information see the UBBCONFIG section of the Tuxedo documentation.

CLOPT options

The following `CLOPT` run-time parameters are recognized:

-s SystemID]

Mandatory option, see CICS SYSID Argument.

-a Application_ID

Optional argument, see CICS Application ID Argument.

-L List_name(s)

Mandatory argument, see Dynamic List of Groups Argument.

Environment Variables Used

- KIXCONFIG
- KIX_CWA_SIZE
- KIX_CWA_IPCKEY
- KIX_TRACE_LEVEL
- KIX_TECH_DIR

Examples

```
*SERVERS

ARTDPL SRVGRP="GRP02" SRVID=60 RESTART=Y GRACE=0

MIN=1 MAX=1

CLOPT=" -- -s PROW -a INVOICE -L list1:list2"
```

ARTATRN Configuration

Server Name

ARTATRN – CICS Runtime server for asynchronous oriented transactions with `MAXACTIVE > 1`.

Synopsis

```
ARTATRN SRVGRP="identifier" SRVID="number" CONV=N MIN=minn MAX=maxn
RQADDR=QKIXATR REPLYQ=Y CLOPT="[servopts] -- -s System_ID -a Application_ID
-L list1:list2:..."
```

Description

`ARTATRN` servers present application transactions as Tuxedo services and when receiving a transaction request, execute the corresponding programs.

These programs are screenless programs which cannot interact directly with the terminal user.

In contrast to `ARTSTRN` servers, these servers are transactional in order to manage true CICS transactions. They are only called from other servers (START TRANSID) and never directly from terminals or clients.

When starting, an `ARTATRN` server publishes one service per transaction it offers. These transactions are named "ASYNC_{transaction_name} (.

This server also publishes an internal transaction called `ASYNC_QUEUE`.

1. When a user program calls a transaction, the `KIX__START_TRANSID` function makes a `tpacall` to the corresponding transaction (service).
2. One `ARTATRN` offering this service receives the request with the associated message, then processes the transaction.
3. The transactions ends without returning a message to the caller.

Only transactions belonging to no tranclasses, or to a tranclass with maxactive >1 are advertised by these servers.

Parameters

CONV

The generic parameter CONV is optional for this server type, if you use it, it must be defined as `CONV=N`, because the `ARTATRN` is non conversational.

minn and maxn

Specify the initial and maximum number of servers to be used to start with this configuration. For more information, see the UBBCONFIG section of the Tuxedo documentation.

CLOPT

A string of command-line options that is passed to the `ARTATRN` when it is booted. The following run-time parameters are recognized:

-s SystemID
: Mandatory argument, see CICS SYSID Argument.

-a Application_ID
: Optional argument, see CICS Application ID Argument.

-L List_name(s)
: Mandatory argument, see Dynamic List of Groups Argument.

Environment Variables Used

KIXCONFIG

KIX_CWA_SIZE

KIX_CWA_IPCKEY

KIX_QSPACE_IPCKEY

KIX_TRACE_LEVEL

KIX_TECH_DIR

Examples

```
*SERVERS

ARTATRN SRVGRP="TCPGRP" SRVID=2000 RESTART=Y GRACE=0

  CONV=N MIN=2 MAX=3 RQADDR=QKIXATR REPLYQ=Y

  CLOPT=" -- -s PROW -a INVOICE -L list1:list2"
```

ARTATR1 Configuration

Server Name

ARTATR1 - CICS Runtime main server for asynchronous oriented transactions with MAXACTIVE = 1.

Synopsis

```
ARTATR1 SRVGRP="identifier" SRVID="number" CONV=N MIN=1 MAX=1
CLOPT="[servopts] -- -s System_ID -a Application_ID -L list1:list2:..."
```

Description

ARTATR1 servers are a specialized version of ARTATRN servers presenting only transactions with MAXACTIVE = 1, whereas ARTATRN servers present transactions with MAXACTIVE > 1.

It is critical, and verified by ATR1 servers at boot time, that MIN and MAX number of servers are set to 1. The goal of these servers is to guarantee the parallel processing of only one transaction in a group (with MAXACTIVE =1). To permit Tuxedo to start several servers offering the same transactions would be self-defeating for ATR1 Servers.

Since MIN and MAX are set to 1, the Tuxedo argument RQADDR, becomes unnecessary, and should be avoided for simplicity.

The rest of the configuration and behavior of ATR1 servers are exactly the same as ATRN servers.

Examples

```
*SERVERS

ARTATR1 SRVGRP="TCPGRP" SRVID=2000 RESTART=Y GRACE=0

CONV=N MIN=1 MAX=1

CLOPT=" -- -s PROW -a INVOICE -L list1:list2"
```

ARTCTRN Configuration

Server Name

ARTCTRN – CICS Runtime server for conversation oriented transactions with MAXACTIVE > 1.

Synopsis

```
ARTCTRN SRVGRP="identifier" SRVID="number" CONV=N MIN=minn MAX=maxn
RQADDR=QKIXCTR REPLYQ=Y CLOPT="[servopts] -- -s System_ID -a Application_ID
-L list1:list2:..."
```

Description

ARTCTRN servers present application transactions as Tuxedo services and when receiving a transaction request, execute the corresponding programs.

These programs are screenless programs which cannot interact directly with the terminal user.

In contrast to ARTSTRN servers, these servers are transactional in order to manage true CICS transactions. They are only called from other servers (CNVERSE) and never directly from terminals or clients.

When starting, a ARTCTRN server publishes one service per transaction it offers. These transactions are named {SysId}_{transaction_name}.

The {SysId} is the name of this region defined in the -s parameter.

1. When a user program calls a transaction, the KIX__CONVERSE function makes a tpacall to the corresponding transaction (service).
2. One ARTCTRN offering this service receives the request with the associated message, then processes the transaction.
3. The transactions ends and the server returns a message to the caller.

Only transactions belonging to no tranclasses, or to a tranclass with maxactive >1 are advertised by these servers.

Parameters

CONV

The generic parameter CONV is optional for this server type; if you use it, it must be defined as CONV=N, because the ARTATRN is transactional.

minn and maxn

Specify the initial and maximum number of servers to be used to start with this configuration. For more information, see the UBBCONFIG section of the Tuxedo documentation.

CLOPT

A string of command-line options that is passed to the ARTCTRN when it is booted. The following run-time parameters are recognized:

-s SystemID
: Mandatory argument, see CICS SYSID Argument.

-a Application_ID
: Optional argument, see CICS Application ID Argument.

-L List_name(s)
: Mandatory argument, see Dynamic List of Groups Argument.

Environment Variables Used

KIXCONFIG

KIX_CWA_SIZE

KIX_CWA_IPCKEY

KIX_QSPACE_IPCKEY

KIX_TRACE_LEVEL

KIX_TECH_DIR

Examples

```
*SERVERS

ARTCTRN SRVGRP="TCPGRP" SRVID=2500 RESTART=Y GRACE=0

  CONV=N MIN=2 MAX=3 RQADDR=QKIXATR REPLYQ=Y

  CLOPT=" -- -s PROW -a INVOICE -L list1:list2"
```

ARTCTR1 Configuration

Server Name

`ARTCTR1` – CICS Runtime main server for conversation oriented transactions with `MAXACTIVE=1`.

Synopsis

```
ARTCTR1 SRVGRP="identifier" SRVID="number" CONV=N MIN=1 MAX=1
```

```
CLOPT="[servopts] -- -s System_ID -a Application_ID -L list1:list2:..."
```

Description

ARTCTR1 servers are a specialized version of ARTCTRN servers presenting only transactions with MAXACTIVE = 1, whereas ARTCTRN servers present transactions with MAXACTIVE > 1.

It is critical, and verified by ARTCTR1 servers at boot time, that MIN and MAX number of servers are set to 1. The goal of these servers is to guarantee the parallel processing of only one transaction in a group (with MAXACTIVE =1). To permit Tuxedo to start several servers offering the same transactions would be self-defeating for ARTCTR1 servers.

Since MIN and MAX are set to 1, the Tuxedo argument RQADDR, becomes unnecessary, and should be avoided for simplicity.

The rest of the configuration and behavior of ARTCTR1 servers are exactly the same as ARTCTRN servers.

Examples

```
*SERVERS

ARTCTR1 SRVGRP="TCPGRP" SRVID=2000 RESTART=Y GRACE=0

  CONV=N MIN=1 MAX=1

  CLOPT=" -- -s PROW -a INVOICE -L list1:list2"
```

ARTCNX Configuration

Server Name

ARTCNX — CICS Runtime connection server for user connection management.

Synopsis

```
ARTCNX SRVGRP="identifier" SRVID="number" CONV=Y MIN=1 MAX=1 RQADDR=QKIX110
REPLYQ=Y CLOPT="[servopts]"
```

Description

This server offers internal services needed by terminal handlers during user connection and disconnection phases.

It offers internal message oriented services such as connect and disconnect:

- connect is in charge of various initialization tasks such as attributing the user Session ID and `Terminal_ID`.
- disconnect manages the disconnection final tasks.

It also offers a few classical CICS transactions:

- CESN: the Sign oN transaction
- CESF: the Sign ofF transaction
- CSGM: the Good Morning transaction (default Good Morning transaction)

It also publishes an internal transaction, `authfail` used by the handler in case of authentication error.

Theses servers are conversational in order to manage CICS transactions CESN, CESF.

This server must be unique in a CICS Runtime system.

Parameters

CONV

The generic parameter `CONV` is mandatory for this server type, and must be defined as `CONV=Y`, because `ARTSTRN` is conversational.

minn and maxn

Must be set to 1. This will still be true in the next release, where each server will be allocated a range of terminal identifiers (see `CLOPT` for more details)

CLOPT

A string of command-line options that is passed to the ARTCNX when it is booted. The following run-time parameter is recognized:
[-t x] (x is included in these ranges, "0 to 9", "A to Z" or "a to z").

Optional parameter used for determine the terminals number (TRMID).

If the parameter is not set you can start only one ARTCNX server (this restriction is checked at start), in this case the terminals number is between 0 to 25,411,680 (0000 to zzzz in base 71)

If you use this parameter, you can start up to 62 ARTCNX servers, each server has up to 357,911 terminals numbers, between 0 to 357,910 (000 to zzz in base 71), in this case the TRMID is composed as follow: x000 to xzzz (x is the character in -t parameter).
At the startup the server cannot check if you have set the same character in the -t parameter in many servers. It is your responsibility to not start several servers with the same parameter, or you risk having duplicated terminal numbers.

Environment Variables Used

KIX_TRACE_LEVEL

Examples

```
*SERVERS

ARTCNX SRVGRP="TCPGRP" SRVID=1000 CONV=Y MIN=1 MAX=1
```

ARTADM Configuration

Server Name

`ARTADM` — Administration Server

Synopsis

```
ARTADM SRVGRP="identifier" SRVID="number" SEQUENCE=1
```

Description

This server is responsible for the administration of CICS resource. It provides the following functionalities:

- Takes charge of loading the resource definitions used by other servers.
- Offers the services used by artadmin (ART administration console) for dynamic administration of CICS resources and propagates the dynamic configuration requests from artadmin to all the concerned servers in the system.
- Propagates the dynamic configuration requests submitted by `artadmin` to all the concerned servers in the system.
- Propagates the resource definition files to the slave machines from the central configuration repository when configured on each node in a distributed environment.

The configuration files only need to be configured on the master node, and the administration servers propagate the configuration files to each slave node.

It is now compulsory to configure a ARTADM server on each machine (master or slave) of the system. The ARTADM server must be started up before other ART servers. The ARTADM server on the master machine must be started up before others on the slave machines.To ensure this sequence, it is necessary to make the following configurations using SEQUENCE:

- For a single machine, configure the ARTADM with SEQUENCE=1.
- For multiple machines, configure the ARTADM on each node:
  - On the master machine, set SEQUENCE=1.
  - On the slave machines, set SEQUENCE=2.

WARNING: Do not use SEQUENCE for other servers, or in any case set with greater numbers.

Environment Variables Used

KIXCONFIG

KIX_TRACE_LEVEL

Examples

```
*SERVERS
ARTADM SRVGRP="ADMGRP" SRVID=1000 RESTART=Y SEQUENCE=1
```

ARTCKTI Configuration

Server Name

ARTCKTI — ART CICS Transaction Trigger Monitor

Synopsis

```
ARTCKTI SRVGRP="identifier" SRVID="number" CLOPT="[servopts options] -- [-i
trigger_interval] [-s retry_interval] [-m queue_manager_name] -q
queue_name1,queue_name2,...”
```

Description

The ART CICS Transaction Trigger Monitor (ARTCKTI) behaves the same as the CICS CKTI transaction. It listens on one or multiple WebSphere MQ initiation queues, gets the trigger message when trigger event occurs, and then forward the trigger message to the target transaction for further operations.

ARTCKTI server accepts the following parameters for the ubbconfig file.

`-i trigger_interval`
: Specifies the maximum time (in milliseconds) that the ARTCKTI server waits for a message to arrive on the initiation queue within each MQGET call.

  Numeric, 0-2147483647. Default value is 5000.

`-s retry_interval`
: Specifies the retry interval (in seconds) for ARTCKTI to reconnect to the WebSphere MQ queue manager or to reopen the WebSphere MQ initiation queue upon failure.

  Numeric, 0-2147483647. Default value is 5.

`-m queue_manager_name`
: Specifies the name of the WebSphere MQ queue manager to be monitored. Only one WebSphere MQ queue manager can be specified for one ARTCKTI server. The default queue manager is used when this parameter is not specified.

`-q queue_name1,queue_name2,……`
: Specifies the names of the initiation queue to be monitored. Multiple WebSphere MQ initiation queues in a WebSphere MQ queue manager can be monitored by one ARTCKTI server.

Server Connection Parameters

ARTCKTI server acts as an WebSphere MQ client, so the channel info for MQ client is needed for ARTCKTI to connect to the WebSphere MQ queue manager.

Generally there are two ways to do this. One is to specify it in the client configuration file, and the other one is to specify it with the environment variable `MQSERVER`.

The channel info should contain the location of the WebSphere MQ server and the communication method to be used. It is a string of the format ChannelName/TransportType/ConnectionName.

ConnectionName must be a fully-qualified network name. ChannelName cannot contain the forward slash (/) character because this character is used to separate the channel name, transport type, and connection name.

ARTCKTI server requires WebSphere MQ multi-threaded library.

For details, please refer to Websphere MQ Client document.

Build ARTCKTI Server

Object files are also provided for users who want to build their own ARTCKTI server based on a different version of WebSphere MQ.

To build the `ARTCKTI` server, execute the following command as the Tuxedo administrator with write permission for the `$KIXDIR/bin` directory:

```
buildserver -o $KIXDIR/bin/ARTCKTI -t -f "$KIXDIR/objs/ARTCKTI.o
$KIXDIR/objs/list.o" -l "-L/$MQM/lib64 -lmqic_r"
```

`$MQM` is the path that WebSphere MQ has been installed.

/Q Configuration for CICS Runtime

/Q Configuration for CICS Runtime

Asynchronous transactions launched using 'EXEC CICS START TRANSID' requests may also be launched with a delay set to an interval or to a fixed time.

In this case, the transaction request is deposited into a Oracle Tuxedo /Q Queue, and when the time is ready, the transaction will be automatically invoked.

For this feature to be available, a few extra components must be activated:

- An Oracle Tuxedo /Q Queue Space named `ASYNC_QSPACE` must be created.
- An Oracle Tuxedo /Q Queue named `ASYNC_QUEUE` will be created in the queue space.
- A TMQFORWARD server will be configured to receive messages from this queue and invoke the application transaction corresponding to the request.

/Q Configuration for Delayed Transactions

Before you begin creating the `qspace` you must load the variable `KIX_QSPACE_IPCKEY` and the Oracle Tuxedo `QMCONFIG` variable.

The `QMCONFIG` variable points to an existing device where the Oracle Tuxedo UDL must be in running mode.

For more details see the Oracle Tuxedo documentation "Creating Queue Spaces and Queues".

Creating an Entry in the Oracle Tuxedo UDL: crdl and Queue Space "ASYNC_QSPACE"

Listing 7-1 shows a crdl and Queue Space "ASYNC_QSPACE" example.

Listing 7-1 crdl and Queue Space "ASYNC_QSPACE"

```
#create the qspace
# qspacecreate -n 1000B
# Queue space name: ASYNC_QSPACE
# IPC Key for queue space: ${KIX_QSPACE_IPCKEY}
# Size of queue space in disk pages: 1000
# Number of queues in queue space: 4
# Number of concurrent transactions in queue space: 9
# Number of concurrent processes in queue space: 9
# Number of messages in queue space: 1000
# Error queue name: errque
# Initialize extents (y, n [default=n]): y
# Blocking factor [default=16]: 16
qmadmin ${QMCONFIG} <<!end
crdl ${QMCONFIG} 0 2000
qspacecreate -n 1000
ASYNC_QSPACE
${KIX_QSPACE_IPCKEY}
1000
4
9
9
1000
errque
```

```
y
16
Q
!end
```

Creating a Queue

Listing 7-2 shows a creating queue example.

Listing 7-2 ASYNC_QUEUE" Using Oracle Tuxedo "qcreate" Tool

```
#create the queue
# qcreate
# Queue name: ASYNC_QUEUE
# Queue order (priority, time, expiration, fifo, lifo): fifo
# Out-of-ordering enqueuing (top, msgid, [default=none]): none
# Retries [default=0]: 2
# Retry delay in seconds [default=0]: 30
# High limit for queue capacity warning (b for bytes used, B for blocks used,
#  % for percent used, m for messages [default=100%]): 80%
# Reset (low) limit for queue capacity warning [default=0%]: 0%
# Queue capacity command:
# No default queue capacity command

qmadmin ${QMCONFIG} <<!end
qopen ASYNC_QSPACE
qcreate
ASYNC_QUEUE
fifo
```

```
none
2
30
80%
0%

qcreate
RPLYQ
fifo
none
2
30
80%
0%

qcreate
errque
fifo
none
2
30
80%
0%
q
!end
```

For more information about errque and RPLYQ see the Oracle Tuxedo documentation.

Oracle Tuxedo /Q Server Configuration in the ubbconfig File

In the *GROUPS Section

```
# /Q
GQUEUE      GRPNO=1000
TMSNAME=TMS_QM TMSCOUNT=2
OPENINFO="TUXEDO/QM:/home/kix04/trf/config/tux/kixqspace:ASYNC_QSPACE"
```

In the *SERVERS Section

```
# /Q
TMQUEUE     SRVGRP=GQUEUE
            SRVID=1010
            RESTART=Y GRACE=0 CONV=N MAXGEN=10
            CLOPT="-s ASYNC_QSPACE:TMQUEUE -- "
TMQFORWARD
            SRVGRP=GQUEUE
            SRVID=1020
            GRACE=0 RESTART=Y CONV=N MAXGEN=10
            CLOPT="-- -n -i 2 -q ASYNC_QUEUE"
```

Security Configuration

Security Configuration

Authentication Configuration

CICS provides two system transactions for authentication purposes:

- CESN is the sign on transaction;
- CESF is the sign off transaction;

ARTTCP implements a similar authentication function leveraging Tuxedo's security mechanisms. Two Tuxedo system services CESN and CESF are provided by CICS Runtime to emulate the CESN and CESF transactions in CICS.

When a terminal connects to ARTTCP, ARTTCP creates a 3270 session and the session joins Tuxedo with the default security profile. The user name defined in the default security profile has the similar role as the CICS default user CICSUSER. The authentication process is then as follows:

1. The operator calls the CESN transaction to sign on to Tuxedo CICS Runtime Runtime.
2. CESN sends a sign-on MAP to ask for username and password.
3. The username and password are entered from the terminal.
4. ARTTCP re-joins Tuxedo using the username and password entered from the terminal.
5. If the authentication:

  - succeeds, a success message is returned to the terminal.
  - fails, an error message is returned to the terminal.

6. When completing the operations, the operator calls service CESF to sign off from Tuxedo CICS Runtime Runtime.

Tuxedo Security Mechanisms

ARTTCP supports three types of Tuxedo security mechanisms: application password (APP_PW), user-level authentication (USER_AUTH), and access control list (ACL and MANDATORY_ACL).

The application password security mechanism requires that every client provide an application password as part of the process of joining the Tuxedo ATMI application. The administrator defines a single password for the entire Tuxedo ATMI application and gives the password only to authorized users. For more information on how to configure Tuxedo application password, please refer to Tuxedo documentation.

The user-level authentication security mechanism requires that in addition to the application password, each client must provide a valid username and password to join the Tuxedo ATMI application. The per-user password must match the password associated with the user name stored in a file named `tpusr`. Client name is not used. The checking of per-user password against the password and user name in `tpusr` is carried out by the Tuxedo authentication service AUTHSVC, which is provided by the Tuxedo authentication server AUTHSVR. For more information on how to configure Tuxedo user-level authentication, please refer to Tuxedo documentation.

When Tuxedo security is enabled, a default security profile, which includes the default USER_AUTH username and password and/or the APP_PW password,, is required to allow users to join the Tuxedo domain before calling the CESN service. A security profile generator tool is introduced to generate the default security profile. Please refer to Security Profile Generator for details.

In the case of APP_PW, the Tuxedo application password must be created in Tuxedo configuration.

In the case of USER_AUTH, the Tuxedo application password, a Tuxedo username and password must be created in the Tuxedo configuration.

In both cases, the password (and username for USER_AUTH) must be specified in the default security profile file that is specified in the command line option (-p profile-name) of theTuxedo ARTTCPL server. The password (and username for USER_AUTH) will be used as parameters of `tpinit()` when ARTTCP server joins Tuxedo.

Integration with the External Security Manager

CICS Runtime offers a security framework which allows a customer to choose integration with an external security manager. The Tuxedo application key (appkey) is used as the credential to be passed to an external security manager. The appkey is 32 bits long, Tuxedo user identifier is in the low order 17 bits and the Tuxedo group identifier is in the next 14 bits (the high order bit is reserved for administrative keys). For more information, please refer to Tuxedo documentation.

An authorisation function is available for customization by the integration team. This function is called by CICS Runtime each time a resource authorization should be checked for a given resource.

A default function that always returns an ok status is provided. It can be replaced by a project specific version by the integration team, for a project where CICS resource authorization must be activated in addition to transaction authorization.

Listing 8-1 COBOL CICS Resource Authorization Interface

```
01 ret-code                    usage int.
LINKAGE SECTION.
01 AUTH-USERID              PIC X(30).
01 AUTH-GROUPID             PIC X(256).
01 AUTH-RSRCE-TYPE          PIC X(256).
01 AUTH-RSRCE-NAME          PIC X(512).
01 AUTH-ACCESS-TYPE         PIC X(6).
PROCEDURE DIVISION USING LK-AUTH-USERID LK-AUTH-GROUPID
                         LK-AUTH-RSRCE-TYPE LK-AUTH-RSRCE-NAME
                         LK-AUTH-ACCESS-TYPE.
```

Accepting

AUTH-USERID	Connection name of the user limited to 8 characters
AUTH-GROUPID	Reserved for future extension
AUTH-RSRCE-TYPE	Type of resource being checked (see Codification).
AUTH-RSRCE-NAME	Name of the resource to check authorization on
AUTH-ACCESS-TYPE	Type of access requested on the resource ("READ", "ALTER", "UPDATE")

Returning

0	For authorization approved.
-1	For authorization refused or failed.

Codification

The resources types are codified as in a native CICS/RACF environment: XTST for Temporary Storage resources, XFCT for files, …

See native CICS documentation for more information. The default version of this function provided with CICS Runtime always returns 0.

TDI_TRIGGER command

Synopsis

```
TDI_TRIGGER -t Transaction_Name [-p <profile>];
```

Parameters

Transaction_name

The transaction to trigger this service should empty the queue.

profile

The name of the profile file to use for autthentification; this file must have been created with `genappprofile`. When not provided it defaults to `~/.TDappProfile`.

Security Profile Generator

When Tuxedo security is enabled, a default security profile, which includes the `APP_PW` password and the default `USER_AUTH` username and password, is required to allow the user to join the Tuxedo domain before calling the CESN service.

A security profile generator tool is introduced to generate the default security profile for TCP.

genappprofile (1)

Name

`genappprofile` — Security Profile Generator

Synopsis

```
genappprofile [-f <output_file>]
```

Description

This utility generates the security profile for Tuxedo applications. When the utility is launched, you are prompted to enter the Tuxedo application password, user name and user password. The output is a security profile file which contains the user name and encrypted passwords. The generated security profile file can be used by CICS Runtime ARTTCPL server to login to the Tuxedo domain.

Options

The command option is:

[-f <output_file>]

The location of the generated security profile file. If this option is not specified, the default value is `~/.tuxAppProfile`.

CICS Commands and Parameters Coverage

CICS Commands and Parameters Coverage

Supported CICS Commands

The following table describes the CICS commands and parameters that are supported by Oracle Tuxedo Application Runtime for CICS.

Note: Commands and parameters not listed in the table below are not supported.

CICS Command and Parameter Support Table

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
ABEND	ABEND	ABCODE (name)	Recognized
		CANCEL	Recognized

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
APPC Mapped conversation	ALLOCATE (APPC)	SYSID (systemname)	
		NOQUEUE	
		STATE	
	CONNECT PROCESS	CONVID (name)	
		PROCNAME (data-area)	
		PROCLENGTH (data-value)	
		SYNCLEVEL	
		STATE	
		SYNCLEVEL (data-value)	Partial Support only SYNCLEVEL 0
	CONVERSE (APPC)	CONVID (name)	
		FROM (data-area)	
		FROMLENGTH (data-value)	
		FROMFLENGTH (data-value)	
		INTO (data-area)	
		TOFLENGTH (data-area)	
		TOLENGTH (data-area)	
		SYNCLEVEL	Partial Support only SYNCLEVEL 0

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
BMS	RECEIVE MAP	MAP (name)	
		MAPSET (name)	
		INTO (data-area)	
		SET (ptr-ref)	
		TERMINAL	
		FROM (data-area)	
		LENGTH (data-value)	
BMS	SEND MAP	MAP (name)	
		MAPSET (name)	
		FROM (data-area)	
		LENGTH (data-value)	
		DATAONLY	
		MAPONLY	
		CURSOR (data-value)	
		ERASE	
		DEFAULT	
		FREEKB	
		ALARM	
		FRSET	
		ACCUM	
		TERMINAL	
		NOFLUSH	
	PURGE MESSAGE		

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
BMS	SEND CONTROL	ERASE	
		DEFAULT	
		ERASEAUP	
		CURSOR (data-value)	
		FREEKB	
		ALARM	
		FRSET	
		ACCUM	
		TERMINAL	

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
BMS	SEND PAGE	RELEASE	
		TRANSID (name)	
		RETAIN	
		TRAILER (data-area)	
	SEND TEXT	FROM (data-area)	
		LENGTH (data-value)	
		CURSOR (data-value)	
		ERASE	
		FREEKB	
		ALARM	
		TERMINAL	
		HEADER (data-area)	
		TRAILER (data-area)	
		JUSTIFY (data-value)	
		ACCUM	
		WAIT	

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
Terminal Control	SEND	FROM (data-area)	
		LENGTH (data-value)	
		FLENGTH (data-value)	
		ERASE	
		CTLCHAR (data-value)	
	RECEIVE	LENGTH (data-value)	
		FLENGTH (data-value)	
		INTO (data-area)	
		SET (ptr-ref)	
		MAXLENGTH (data-value)	
		MAXFLENGTH (data-value)	
		BUFFER	
		NOTRUNCATE	
Built-in functions	BIF DEEDIT	FIELD (data-area)	
		LENGTH (data-value)	

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
Environmental services	ADDRESS	CWA (ptr-ref)	
		TCTUA (ptr-ref)	
		TWA (ptr-ref)	
		EIB (ptr-ref)	
	ADRESS SET	SET	
		USING	
	ASSIGN	ABCODE (data-area)	Recognized
		NETNAME (data-area)	Recognized
		APPLID (data-area)	
		OPID (data-area)	Recognized
		CWALENG (data-area)	
		PROGRAM (data-area)	
		STARTCODE (data-area)	
		SYSID (data-area)	
		TCTUALENG (data-area)	
		TERMCODE (data-area)	Recognized
		TWALENG (data-area)	
		USERID (data-area)	
		USERNAME (data-area)	Recognized

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
FILES	RESETBR	FILE	
		DATASET	
		RIDFLD	
		KEYLENGTH	
		GTEQ	
		EQUAL	
	UNLOCK	FILE	
		DATASET	
		TOKEN	
File control	DELETE	FILE (filename)	
		DATASET(filename)	
		RIDFLD (data-area)	
		KEYLENGTH (data-value)	
		GENERIC	Recognized
		NUMREC (data-area)	Recognized
		SYSID (systemname)	Recognized
	ENDBR	FILE (filename)	
		DATASET(filename)	
		REQID (data-value)	Recognized
		SYSID (systemname)	Recognized

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
File control	READ	FILE (filename)	
		DATASET(filename)	
		UPDATE	Recognized
		INTO (data-area)	
		SYSID (systemname)	Recognized
		LENGTH (data-area)	
		SET (ptr-ref)	
		RIDFLD (data-area)	
		KEYLENGTH (data-value)	
		RBA	Recognized
		RRN	Recognized
		EQUAL	
		GTEQ	
File control	READNEXT	FILE (filename)	
		DATASET(filename)	
		INTO (data-area)	
		SET (ptr-ref)	
		RIDFLD (data-area)	
		KEYLENGTH (data-value)	
		SYSID (systemname)	Recognized
		LENGTH (data-area)	
		RBA	Recognized
		RRN	Recognized

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
File control	READPREV	FILE (filename)	
		DATASET(filename)	
		INTO (data-area)	
		SET (ptr-ref)	
		RIDFLD (data-area)	
		KEYLENGTH (data-value)	
		SYSID (systemname)	Recognized
		LENGTH (data-area)	
		RBA	Recognized
		RRN	Recognized
	REWRITE	FILE (filename)	
		DATASET(filename)	
		FROM (data-area)	
		SYSID (systemname)	Recognized
		LENGTH (data-area)	

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
File control	STARTBR	FILE (filename)	
		DATASET(filename)	
		RIDFLD (data-area)	
		KEYLENGTH(data-value)	
		GENERIC	Recognized
		REQID (data-value)	Recognized
		SYSID (systemname)	Recognized
		RBA	Recognized
		RRN	Recognized
		GTEQ	
		EQUAL	
	WRITE	FILE (filename)	
		DATASET(filename)	
		FROM (data-area)	
		RIDFLD (data-area)	
		KEYLENGTH(data-value)	
		SYSID (systemname)	Recognized
		LENGTH(data-area)	
		RBA	Recognized
		RRN	Recognized
Interval control	ASKTIME	ABSTIME (data-area)	

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
Interval control	FORMATTIME	ABSTIME (data-area)	
		DATE(data-area)	
		FULLDATE(data-area)	
		DATEFORM(data-area)	
		DATESEP (data-value)	
		DAYCOUNT(data-area)	
		DAYOFMONTH (data-area)	
		DAYOFWEEK (data-area)	
		DDMMYY (data-area)	
		DDMMYYYY(data-area)	
		MMDDYY (data-area)	
		MMDDYYYY (data-area)	
		MONTHOFYEAR (data-area)	
		TIME (data-area)	
		TIMESEP (data-value)	
		YEAR (data-area)	
		YYDDD (data-area)	
		YYDDMM (data-area)	
		YYMMDD(data-area)	
		YYYYDDD(data-area)	
		YYYYDDMM (data-area)	
		YYYYMMDD (data-area)	
	CANCEL	REQID (name)	

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
Interval control	DELAY	INTERVAL (hhmmss)	
		SECONDS (data-value)	
		REQID (name)	
		INTERVAL	
		TIME	
		FOR: HOUR MINUTES SECONDS	
	RETRIEVE	INTO (data-area)	
		SET (ptr-ref)	
		LENGTH (data-area)	
		RTRANSID	
		RTERMID	
		QUEUE	
	START	TRANSID (name)	
		INTERVAL (hhmmss)	
		FROM (data-area)	
		LENGTH (data-value)	
		TERMID (name)	Recognized
		QUEUE	
		USERID (data-value) The security stub is called with USERID and TRANSID.	Partial

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
Interval control	START	TIME (hhmmss) AT AFTER HOURS (data-value) MINUTES (data-value) SECONDES (data-value)	
		PROTECT	
		RTERMID	
		RTRANSID	
		SYSID	(In M3_L3).
Program control	LINK	PROGRAM (name)	
		COMMAREA (data-area)	
		SYSID (systemname)	
		LENGTH (data-value)	
	RETURN	TRANSID (name)	
		IMMEDIATE	
		COMMAREA (data-area)	
		LENGTH (data-value)	
Program control	XCTL	PROGRAM (name)	
		COMMAREA (data-area)	
		LENGTH (data-value)	
Syncpoint	SYNCPOINT		
	SYNCPOINT ROLLBACK		

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
System's programmer's function	INQUIRE TRANSACTION	INQUIRE TRANSACTION (datavalue)	
		STATUS (cvda)	
	INQUIRE TRANSCLASS	INQUIRE TRANSACTION (datavalue)	MAXACTIVE: The following parameters are not applicable in CICS Runtime environment: • ACTIVE • PURGETHRESH • QUEUED.
		STATUS (cvda)	
	SPOOLCLOSE	TOKEN(data-area)	
		DELETE	Recognized
	SPOOLOPEN OUTPUT	TOKEN(data-area)	
		USERID(data-value)	Supports only INTRDR
		NODE(data-value)	Recognized
		CLASS(data-value)	
		PUNCH	Recognized
		RECORDLENGTH(data-value)	
	SPOOLWRITE	TOKEN(data-area)	
		FROM(data-area)	
		FLENGTH(data-value)	

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
Task control	DEQ	RESOURCE (data-area)	Mandatory
		LENGTH (data-value)	Mandatory, only the enqueues and dequeues on data values are supported, not the enqueues on address.
		UOW	
		TASK	
	ENQ	RESOURCE (data-area)	Mandatory
		LENGTH (data-value)	Mandatory, only the enqueues and dequeues on data values are supported, not the enqueues on address.
		NOSUSPEND	
		UOW	
		TASK	
Temporary storage	DELETEQ TS	QUEUE (name)	
		QNAME(name)	
		SYSID (systemname)	Recognized

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
Temporary storage	READQ TS	QUEUE (name)	
		QNAME(name)	
		INTO (data-area)	
		SET (ptr-ref)	
		LENGTH (data-area)	
		NUMITEMS (data-area)	
		NEXT	
		ITEM (data-value)	
		SYSID (systemname)	Recognized
	WRITEQ TS	QUEUE (name)	
		QNAME(name)	
		FROM (data-area)	
		SET (ptr-ref)	
		LENGTH (data-value)	
		NUMITEMS (data-area)	
		ITEM (data-value)	
		REWRITE	
		SYSID (systemname)	Recognized
		AUXILIARY	
		MAIN	
		NOSUSPEND	Recognized

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
Transient data	READQ TD	QUEUE (name)	
		INTO (data-area)	
		LENGTH (data-area)	
	WRITEQ TD	QUEUE (name)	
		FROM (data-area)	
		LENGTH (data-value)	
		SYSID (systemname)	
	DELETEQ TD	QUEUE (name)	
		SYSID (systemname)	

Table 9-1 CICS Command

CICS domain	CICS command	Command parameter	Status
Storage Control	GETMAIN		We will not support the SHARED memory allocation. **Note:** NOSUSPEND will be implicit. Not supported in Cobol IT
	FEEEMAIN	DATA	Not supported in Cobol IT
		DATAPOINTER	
Console	WRITE OPERATOR		We provide a stub function. This function can be replaced by the integration team for project needs.

Recognized parameters are processed by the pre-processor and have no effect on the behavior of CICS Runtime.

External Interface for Write Operator

The "`WRITE OPERATOR`" function calls a "stub" named `ExternWriteOperator`.

`ExternWriteOperator` receives all parameters of the `WRITE OPERATOR` and simply returns zero in the return code and nothing else.

It can be replaced by a customer function that respects the interface described below.

The `WRITE OPERATOR` passes the following parameters and expects a return code in signed int format.

Listing 9-1 WRITE OPERATOR Parameters

```
TEXT            pic x(1024).
TEXTLENGTH      PIC S9(9) COMP-5.
ROUTECODES      pic x(1024).
NUMROUTES       PIC S9(9) COMP-5.
ACTION          PIC X(2).
REPLY           pic x(1024).
MAXLENGTH       PIC S9(9) COMP-5.
REPLYLENGTH     PIC S9(9) COMP-5.
TIMEOUT         PIC S9(9) COMP-5.
```

`S9(9) COMP-5` is equivalent to a signed int.

The parameters `REPLY` and `REPLYLENGTH` may be returned to the `WRITE OPERATOR` function if requested, that is to say, if `MAXLENGTH > zero`.

Example COBOl Code for ExternWriteOperator

Listing 9-2 Example ExternWriteOperator.cbl Code

```
IDENTIFICATION DIVISION.
PROGRAM-ID. "ExternWriteOperator".
DATA DIVISION.
WORKING-STORAGE SECTION.
copy "ctypes".
```

```
       01 ret-code            usage int.
       LINKAGE SECTION.
       01 LK-TEXT             pic x(1024).
       01 LK-TEXTLENGTH       PIC S9(9) COMP-5.
       01 LK-ROUTECODES       pic x(1024).
       01 LK-NUMROUTES        PIC S9(9) COMP-5.
       01 LK-ACTION           PIC X(2).
       01 LK-REPLY            pic x(1024).
       01 LK-MAXLENGTH        PIC S9(9) COMP-5.
       01 LK-REPLYLENGTH      PIC S9(9) COMP-5.
       01 LK-TIMEOUT          PIC S9(9) COMP-5.

       PROCEDURE DIVISION USING LK-TEXT LK-TEXTLENGTH LK-ROUTECODES
                                LK-NUMROUTES LK-ACTION LK-REPLY
                                LK-MAXLENGTH LK-REPLYLENGTH LK-TIMEOUT.


*      * display "ExternWriteOperator : LK-TEXT        =<" LK-TEXT ">"
*      * display "ExternWriteOperator : LK-TEXTLENGTH  =<" LK-TEXTLENGTH ">"
*      * display "ExternWriteOperator : LK-ROUTECODES  =<" LK-ROUTECODES ">"
*      * display "ExternWriteOperator : LK-NUMROUTES   =<" LK-NUMROUTES ">"
*      * display "ExternWriteOperator : LK-ACTION      =<" LK-ACTION ">"
*      * display "ExternWriteOperator : LK-REPLY       =<" LK-REPLY ">"
*      * display "ExternWriteOperator : LK-MAXLENGTH   =<" LK-MAXLENGTH ">"
*      * display "ExternWriteOperator : LK-REPLYLENGTH =<" LK-REPLYLENGTH ">"
*      * display "ExternWriteOperator : LK-TIMEOUT     =<" LK-TIMEOUT ">"
```

```
*     *    in case of REPLY
           if LK-MAXLENGTH > zero
*     *       move "..............." to LK-REPLY
*     *       move 15                to LK-REPLYLENGTH
           end-if

           move zero to ret-code
*     *    return code
*     *    0  = OK
*     *    -1 = operation failed (INVREC wil returned to the user program)
*     *    -9 = time out ocurred before the operators's reply was received


           GOBACK returning ret-code.
```

External Interface for Query Security

The "QUERY SECURITY" function calls a “stub” named ExternQuerySecurity.

The delivered ExternQuerySecurity stub receives all parameters of the QUERY SECURITY, it always allows access to the resources and returns zero in the return code. It can be replaced by a customer function that respects the interface described below.

The QUERY SECURITY passes the following parameters:

Listing 9-3 Query Security Extern Inteface

```
restype                   pic x(7).
restype-data-value        pic x(12).
resclass                  pic x(8).
resclass-data-value       pic x(8).
residlength               pic x(11).
```

```
residlength-data-value      pic s9(8) comp-5.
resid                       pic x(5).
resid-data-value            pic x(246).
logmessage                  pic x(10).
logmessage-cvda             pic s9(8) comp-5.
read                        pic x(10).
read-cvda                   pic s9(8) comp-5.
update                      pic x(10).
update-cvda                 pic s9(8) comp-5.
control                     pic x(10).
control-cvda                pic s9(8) comp-5.
alter                       pic x(10).
alter-cvda                  pic s9(8) comp-5.
resp                        pic s9(8) comp-5.
resp2                       pic s9(8) comp-5.
userid                      pic x(8).
```

All parameters are passed to ExternQuerySecurity, only the following parameters are expected in return:

```
read-cvda                   pic s9(8) comp-5.
update-cvda                 pic s9(8) comp-5.
control-cvda                pic s9(8) comp-5.
alter-cvda                  pic s9(8) comp-5.
resp                        pic s9(8) comp-5.
resp2                       pic s9(8) comp-5.
```

If "read" is fulfilled with "READ", read-cvda is expected.

If "update" is fulfilled with "READ", update-cvda is expected.

If "control" is fulfilled with "READ", control-cvda is expected.

If "alter" is fulfilled with "READ", alter-cvda is expected.

"resp" and "resp2" are always expected.

Note:

Each interface field is ended by a binary zero, it easier if you want write the "ExternQuerySecurity" in C.

The cvda values for "read" are:

```
READABLE         35.
NOTREADABLE      36.
```

The cvda values for "update" are:

```
UPDATABLE        37.
NOTUPDATABLE     38.
```

The cvda values for "control" are:

```
CTRLABLE         56.
NOTCTRLABLE      57.
```

The cvda values for "alter" are:

```
ALTERABLE        52.
NOTALTERABLE     53.
```

For more details, see cvda values in IBM documentation.

S9(9) COMP-5 is equivalent to a signed int.

Example COBOl Code for ExternQuerySecurity

Listing 9-4 Example COBOL Code for ExternQuerySecurity

```
IDENTIFICATION DIVISION.
       PROGRAM-ID. "ExternQuerySecurity".
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       copy "ctypes".
       01 ret-code            usage int.
```

```
01 cvda-logmessage        pic s9(8) comp-5.
   88 LOG                 value 54.
   88 NOLOG               value 55.
01 cvda-read              pic s9(8) comp-5.
   88 READABLE            value 35.
   88 NOTREADABLE         value 36.
01 cvda-update            pic s9(8) comp-5.
   88 UPDATABLE           value 37.
   88 NOTUPDATABLE        value 38.
01 cvda-control           pic s9(8) comp-5.
   88 CTRLABLE            value 56.
   88 NOTCTRLABLE         value 57.
01 cvda-alter             pic s9(8) comp-5.
   88 ALTERABLE           value 52.
   88 NOTALTERABLE        value 53.
LINKAGE SECTION.
01 LK-restype                    pic x(7).
01 LK-restype-data-value         pic x(12).
01 LK-resclass                   pic x(8).
01 LK-resclass-data-value        pic x(8).
01 LK-residlength                pic x(11).
01 LK-residlength-data-value     pic s9(8) comp-5.
01 LK-resid                      pic x(5).
01 LK-resid-data-value           pic x(246).
01 LK-logmessage                 pic x(10).
01 LK-logmessage-cvda            pic s9(8) comp-5.
01 LK-read                       pic x(10).
```

```
01 LK-read-cvda                  pic s9(8) comp-5.
01 LK-update                     pic x(10).
01 LK-update-cvda                pic s9(8) comp-5.
01 LK-control                    pic x(10).
01 LK-control-cvda               pic s9(8) comp-5.
01 LK-alter                      pic x(10).
01 LK-alter-cvda                 pic s9(8) comp-5.
01 LK-resp                       pic s9(8) comp-5.
01 LK-resp2                      pic s9(8) comp-5.
01 LK-userid                     pic x(8).
PROCEDURE DIVISION USING LK-restype
                         LK-restype-data-value
                         LK-resclass
                         LK-resclass-data-value
                         LK-residlength
                         LK-residlength-data-value
                         LK-resid
                         LK-resid-data-value
                         LK-logmessage
                         LK-logmessage-cvda
                         LK-read
                         LK-read-cvda
                         LK-update
                         LK-update-cvda
                         LK-control
                         LK-control-cvda
                         LK-alter
```

```
                                LK-alter-cvda
                                LK-resp
                                LK-resp2
                                LK-userid
                                .
*    *   display "ExternQuerySecurity : LK-restype           =" LK-restype
*     *    display "ExternQuerySecurity : LK-restype-data-value    ="
LK-restype-data-value
*     *    display "ExternQuerySecurity : LK-resclass              ="
LK-resclass
*     *    display "ExternQuerySecurity : LK-resclass-data-value   ="
LK-resclass-data-value
*     *    display "ExternQuerySecurity : LK-residlength           ="
LK-residlength
*     *    display "ExternQuerySecurity : LK-residlength-data-value="
LK-residlength-data-value
*     *    display "ExternQuerySecurity : LK-resid              =" LK-resid
*     *    display "ExternQuerySecurity : LK-resid-data-value      ="
LK-resid-data-value
*     *    display "ExternQuerySecurity : LK-logmessage            ="
LK-logmessage
*     *    display "ExternQuerySecurity : LK-logmessage-cvda       ="
LK-logmessage-cvda
*     *    display "ExternQuerySecurity : LK-read               =" LK-read
*     *    display "ExternQuerySecurity : LK-read-cvda             ="
LK-read-cvda
*     *    display "ExternQuerySecurity : LK-update             =" LK-update
*     *    display "ExternQuerySecurity : LK-update-cvda           ="
LK-update-cvda
*     *   display "ExternQuerySecurity : LK-control           =" LK-control
```

```
*     *    display "ExternQuerySecurity : LK-control-cvda          ="
LK-control-cvda
*     *   display "ExternQuerySecurity : LK-alter              =" LK-alter
*     *    display "ExternQuerySecurity : LK-alter-cvda           ="
LK-alter-cvda
*     *    display "ExternQuerySecurity : LK-resp                =" LK-resp
*     *    display "ExternQuerySecurity : LK-resp2               =" LK-resp2


            if address of LK-read not = null
               if LK-read = "READ"
                  set READABLE         to true
                  move cvda-read       to LK-read-cvda
               end-if
            end-if
            if address of LK-update not = null
               if LK-update = "UPDATE"
                  set UPDATABLE        to true
                  move cvda-update     to LK-update-cvda
               end-if
            end-if
            if address of LK-control not = null
               if LK-control = "CONTROL"
                  set CTRLABLE         to true
                  move cvda-control    to LK-control-cvda
               end-if
            end-if
            if address of LK-alter not = null
```

```
               if LK-alter = "ALTER"
                  set ALTERABLE        to true
                  move cvda-alter      to LK-alter-cvda
               end-if
            end-if

            move zero to LK-resp LK-resp2

            move zero to ret-code
*     *     return code
*     *     0  = OK
*     *     -1 = operation failed (INVREC wil returned to the user program)

            GOBACK returning ret-code.
```

Developing the ExternQuerySecurity in C

1. Use this interface:

   int ExternQuerySecurity(char *restype, char *restype_data_value, char *resclass, char *resclass_data_value, char *residlength, int *residlength_data_value, char *resid, char *resid_data_value, char *logmessage, int *logmessage_cvda, char *read, int *read_cvda, char *update, int *update_cvda, char *control, int *control_cvda, char *alter, int *alter_cvda, int *resp, int *resp2, char *userid);

2. Generate an "`ExternQuerySecurity.o`", and link it in ART servers (STRN, STR1, ATRN, ATR1, CTRN, CTR1).

3. In the makefile_intg (Cics_Rt/tools), add the "`ExternQuerySecurity.o`" in object variables for each server (STRN_OBJS, STR1_OBJS, ATRN_OBJS, ATR1_OBJS, CTRN_OBJS, CTR1_OBJS)

4. Execute the makefile to rebuild these servers.

Supported BMS Macros

The following describes the BMS Macros that are supported by Oracle Tuxedo Application Runtime for CICS.

- Mapset DFHMSD
- Map DFHMDI
- Field DFHMDF

Mapset DFHMSD

TYPE=(Choose only one from list)

```
SYSPARM (recognized)
DSECT (recognized)
MAP (recognized)
FINAL
```

Note: `recognized` means function is not achieved, but there are no errors when doing MAPGEN compilation.

MODE=(Choose only one from list)

```
OUT
IN
INOUT
```

LANG=(Choose only one from list)

```
COBOL
PLI
C
```

STORAGE=AUTO

BASE=NAME

CTRL=(Choose any combination from the list, separate with a comma)

```
FREEKB
ALARM
FRSET
```

EXTATT=(Choose only one from list)

```
NO
MAPONLY
YES
```

COLOR=(Choose only one from list)

```
DEFAULT
Color
```

HILIGH=(Choose only one from list)

```
OFF
BLINK
REVERSE
UNDERLINE
```

PS=(Choose only one from list)

```
BASE
Psid
```

Note: PS=8 is only supported for data initialization in GINIT and XINIT.

TERM=3270-2

MAPATTS=(Choose any combination from list, separate with a comma)

```
COLOR
HILIGHT
OUTLINE
PS
SOSI
```

DSATTS=(Choose any combination from list, separate with a comma)

```
COLOR
HILIGHT
OUTLINE
PS
SOSI
```

OUTLINE=(Choose only one from list)

```
BOX
LEFT
LEFT,RIGHT
LEFT,OVER
LEFT,UNDER
LEFT,RIGHT,OVER
LEFT,RIGHT,UNDER
LEFT,RIGHT,OVER,UNDER
RIGHT
RIGHT,OVER
RIGHT,UNDER
RIGHT,OVER,UNDER
OVER
OVER,UNDER
```

UNDER

SOSI=(Choose only one from list)

YES
NO

Map DFHMDI

SIZE=(line,column)

CTRL=(Choose any combination from the list, separate with a comma)

FREEKB
ALARM
FRSET

EXTATT=(Choose only one from list)

NO
MAPONLY
YES

COLOR=(Choose only one from list)

DEFAULT
Color

HILIGHT=(Choose only one from list)

OFF
BLINK
REVERSE
UNDERLINE

PS=(Choose only one from list)

BASE
psid

MAPATTS=(Choose any combination from list, separate with a comma)

COLOR
HILIGHT
OUTLINE
PS
SOSI

DSATTS=(Choose any combination from list, separate with a comma)

COLOR
HILIGHT
OUTLINE
PS

```
SOSI
```

OUTLINE=(Choose only one from list)

```
BOX
LEFT
LEFT,RIGHT
LEFT,OVER
LEFT,UNDER
LEFT,RIGHT,OVER
LEFT,RIGHT,UNDER
LEFT,RIGHT,OVER,UNDER
RIGHT
RIGHT,OVER
RIGHT,UNDER
RIGHT,OVER,UNDER
OVER
OVER,UNDER
UNDER
```

SOSI=(Choose only one from list)

```
YES
NO
```

COLUMN=(Choose only one from list)

```
SAME
Number
NEXT
```

LINE=(Choose only one from list)

```
SAME
Number
NEXT
```

JUSTIFY=(Choose only one from list)

```
LEFT
LEFT,FIRST
LEFT,LAST
RIGHT
RIGHT,FIRST
RIGHT,LAST
BOTTOM
```

HEADER=YES

TRAILER=YES

Field DFHMDF

POS=(Choose only one from list)

```
Number
Line,Column
```

LENGTH=number

JUSTIFY=(Choose only one from list)

```
LEFT
LEFT,BLANK
LEFT,ZERO
RIGHT
RIGHT,BLANK
RIGHT,ZERO
```

INITIAL='char-data'

XINIT='hex-data'

GINIT='DBCS-characters'

ATTRB=(parameter group A, parameter B, parameter group C)

Choose exactly one from parameter group A:

```
ASKIP
PROT
UNPROT
UNPROT,NUM
```

Choose zero or one from parameter group B:

```
BRT
NORM
DRK
```

Choose any combination (0 to all) from parameter group C; this group is allowed only if parameter from gourp B is also used:

```
IC
FSET
```

COLOR=(Choose only one from list)

```
DEFAULT
Color
```

PS=(Choose only one from list)

```
BASE
Psid
```

HILIGHT=(Choose only one from list)

```
OFF
BLINK
REVERSE
UNDERLINE
```

GRPNAME=group-name

OCCURS=number

PICIN='value'

PICOUT='value'

OUTLINE=(Choose only one from list)

```
BOX
LEFT
LEFT,RIGHT
LEFT,OVER
LEFT,UNDER
LEFT,RIGHT,OVER
LEFT,RIGHT,UNDER
LEFT,RIGHT,OVER,UNDER
RIGHT
RIGHT,OVER
RIGHT,UNDER
RIGHT,OVER,UNDER
OVER
OVER,UNDER
UNDER
```

SOSI=(Choose only one from list)

```
YES
NO
```

CHAPTER 10

System Commands and Transactions

System Commands

Table 1 System Commands

Name	Description
cpy2view32(1)	Generates VIEW32 definition from COPYBOOK
tcxcsdcvt (1)	CICS CSD Converter
tcxmapgen(1)	CICS Runtime MAPSET Generator
artadmin (1)	ART CICS Runtime administration

cpy2view32(1)

Name

cpy2view32—Generates Oracle Tuxedo VIEW32 definition file from COBOL copybook file.

Synopsis

```
cpy2view32 [OPTION...] FILE
```

Description

This utility parses the COBOL copybook file and generates the corresponding Oracle Tuxedo VIEW32 definition files.

It supports the following options:

`-n`
: Specifies the source copybook is in "normal" format (i.e., copybook contains sequence number area (columns 1 through 6 is the sequence number area, followed by the indicator area)).

`-e`
: Specifies the source copybook is in "exceptional" format (i.e.,copybook does not contain a sequence number area (the first column is the indicator area)).

`-o`
: Specifies the output file name, followed parameter is the output file name. If this parameter is not specified, the output file name changes the suffix of the input file name to `.v`. For example, `abc.cbl` is converted to `abc.v`.

This utility supports the following annotation in the source copybook:

`* @binary`: by default, copybook data types without the following qualifiers are converted to `string`: `BINARY, COMP, COMP-1, COMP-2, COMP-3, COMP-4, COMP-5, PACKED-DECIMAL`. With the `* @binary=true` annotation, the copybook data types without those qualifiers are converted to `CARRAY`. `* @binary=false` changes the conversion rule back to the default. When this annotation is defined on a group, all subordinates in the group are affected.

Environment Variables

`PATH`
: The `cpy2view32` utility is written in Java. You must install JDK 1.6 or above and add the Java command "java" to the `PATH` environment variable.

Example(s)

1. The following example converts normal copybook file `abc_orig.cbl` to view file `abc_orig.v`:

   ```
   cpy2view32 /home/abc_orig.cbl
   ```

2. The following example converts exceptional copybook file abc.cbl to view file `abc.v`:

   ```
   cpy2view32 -e /home/abc.cbl
   ```

3. The following example converts normal copybook file `abc_orig.cbl`and outputs to view file `xyz.v`:

   ```
   cpy2view32 -o xyz.v /home/abc_orig.cbl
   ```

4. Listing 10-1 through Listing 10-4 provide Copybook and view file output examples:

Listing 10-1 Copybook Example 1

```
#########################
        01 BOOK-INFO.
           05 BOOK-ID     PIC 9(9) COMP-5.
           05 BOOK-NAME   PIC X(100).
           05 PUBLISHER   PIC X(100).
           05 PRICE       USAGE COMP-1.
#########################
```

Listing 10-2 VIEW32 Output Example 1

```
#########################
#type                 cname                 fbname                count
flag    size    null


VIEW book_info
unsignedint              book_id              -               1   -   -   -
string               book_name               -               1   -   100 -
string               publisher               -               1   -   100 -
float                 price                 -               1   -   -   -
END
#########################
```

Listing 10-3 Copybook Example 2

```
##########################
        01 COMPUTER.
           05 COMPUTER-ID     PIC 9(9) COMP-5.
           05 COMPUTER-NAME   PIC X(20).
           05 PRODUCER        PIC X(40).
           05 FILLER          PIC X(4).

           05 SELL-PRICE      USAGE COMP-2.
           05 RENTAL-PRICE    PIC S9999V999 PACKED-DECIMAL.
           05 KEYBOARD-PRICE  PIC S9(4) SIGN IS LEADING SEPARATE.
           05 MOUSE-PRICE  PIC S9(4) SIGN IS LEADING.

           05 FILLER PIC X(4).
      * define other computer components below
           05 CPU.
              10 MODEL     PIC X(20).
              10 PRODUCER  PIC X(40).
              10 PRICE     USAGE COMP-1.
           05 COMPUTER-MEMORY OCCURS 4 TIMES.
              10 MODEL PIC X(20).
              10 PRODUCER  PIC X(40).
              10 PRICE     USAGE COMP-1.
           05 MAINBOARD.
              10 MODEL     PIC X(20).
              10 PRODUCER  PIC X(40).
              10 PRICE     USAGE COMP-1.
```

```
           05 MONITOR.
              10 MODEL     PIC X(20).
              10 PRODUCER  PIC X(40).
              10 PRICE     USAGE COMP-1.
           05 HARDDISK.
              10 MODEL     PIC X(20).
              10 PRODUCER  PIC X(40).
              10 PRICE     USAGE COMP-1.
##########################
```

Listing 10-4 VIEW32 Output Example 2

```
##########################
#type                 cname                fbname                count
flag    size    null

VIEW cpu
string                model                -                 1   -   20  -
string               producer                -               1   -   40  -
float                 price                -                1   -   -   -
END

VIEW computer_memory
string                model                -                 1   -   20  -
string               producer                -               1   -   40  -
float                 price                -                1   -   -   -
END
```

```
VIEW mainboard
string                model                -                1   -   20  -
string               producer               -                1   -   40  -
float                 price                -                1   -   -   -
END

VIEW monitor
string                model                -                1   -   20  -
string               producer               -                1   -   40  -
float                 price                -                1   -   -   -
END

VIEW harddisk
string                model                -                1   -   20  -
string               producer               -                1   -   40  -
float                 price                -                1   -   -   -
END

VIEW computer
unsignedint             computer_id             -              1   -   -   -
string              computer_name               -              1   -   20  -
string               producer                 -              1   -   40  -
string                filler1                -                1   -   4   -
double               sell_price               -              1   -   -   -
carray              rental_price                -              1   -   4   -
string              keyboard_price              -              1   -   5   -
string               mouse_price              -               1   -   4   -
```

```
string                filler2               -                1   -   4   -
struct                 cpu                cpu                1   -   -   -
struct             computer_memory            computer_memory
4   -   -   -
struct               mainboard               mainboard               1
-   -   -
struct               monitor             monitor              1   -   -   -
struct               harddisk                harddisk                1
-   -   -
END
#########################
```

Limitations

The following are general `cpy2view32` limitations:

1. This tool does not parse `REDEFINES` clause, `REDEFINES` clause and their subordinate items are skipped.
2. `POINTER` phrase, `FUNCTION-POINTER` phrase and `PROCEDURE-POINTER` phrase are skipped in the conversion.
3. `VALUE` clause is skipped in the conversion.
4. `SYNCHRONIZED` clause is skipped in the conversion.
5. `JUSTIFIED` clause is skipped in the conversion.
6. `BINARY`, `COMP`, and `COMP-4` are synonyms, they are converted to `CARRAY` in view file.

tcxcsdcvt (1)

Name

`tcxcsdcvt` -- translates RDO file to all z/OS resource configuration files.

Synopsis

```
tcxcsdcvt [-option] [Filename]
```

Description

`tcxcsdcvt` translates RDO files to all z/OS resource configuration files.The generated resource configuration files by default are found in the current directory where this tool is run.

`tcxcsdcvt` supports the following options:

`-h`
: Display help information for this tool.

`-d <director>`
: Specifies the target directory for generated configuration files.

`-D`
: Generate log file in case there is error information during conversion.

Example(s)

To convert the RDO file "lirgao.cicsb.dfhcsd", enter following command:

```
tcxcsdcvt lirgao.cicsb.dfhcsd
```

tcxmapgen(1)

Name

`tcxmapgen` — CICS Runtime MAPSET Generator.

Synopsis

```
tcxmapgen [-options] <file>
```

Description

CICS Runtime provides a mapset generator to compile BMS macro source files, to produce a physical (binary) file and a symbolic (copybook) file. There is also an option to produce a listing file. During execution, the mapset generator validates the syntax and level of support for each BMS macro statement.

The generated physical (binary) file should be used in the MAPSET configuration file. See “Mapset Configuration File” in CICS Runtime Configuration Files.

The generated symbolic (copybook) file should be included when you compile the CICS/COBOL program which uses the MAP in this MAPSET

Options

The command options are:

`[-c]`
: Specifies that no binary mapset file (`.mpdef`) is produced.

`[-l]`
: Specifies a listing output file (`.lst`) is produced.

`[-m]`
: Specifies that no COBOL copybook (`.cpy`) output file is generated.

`[-o file]`
: Specifies the name used for the generated output files. The compiler uses the file name with an appended extension when creating the output file names.

`[-u]`

: Specifies that the output fields are not sorted but kept in the defined order. Without specifying this option, all fields in a map are sorted according to their positions by default.

Example(s)

To compile the BMS source file `file.map`, use the following command:

```
$ tcxmapgen -o file file.map
```

The resulting binary mapset file is `file.mpdef`.

artadmin (1)

Name

`artadmin` — ART CICS Runtime administration.

Synopsis

```
artadmin [-p <profile>]
```

Parameter

`Profile`

The name of the profile file used for authentification. This parameter is useful for secure Oracle Tuxedoconfiguration. The profile file must be created with `genappprofile`. If no file name is provided, it defaults to `~/.ADMINappProfile`.

Description

In some cases it is necessary to modify the configuration while the system is running. Normally, configuration changes which are relative to performances are managed by Oracle Tuxedo or the RDBMS level using commands (for example, `tmadmin`) or Oracle TSAM for Oracle Tuxedo dynamic configuration.

However, if the requirement is more functional (for example, needing to put some transactions online, installing a hot fix for some programs and screens, or changing some resource configurations), you must use `artadmin`.

`artadmin` is useful when making hot configuration changes in the CICS resources of a running ART CICS system. It allows the administrator to:

- request all or partial servers to take in changes to the CICS resources configuration files.
- transmit a New Copy request to do a hot fix of some modified programs or maps.

`artadmin` is launched interactively (similar to `tmadmin`). When the `artdmin` is launched and connects to Oracle Tuxedo successfully, it returns a prompt requiring you to enter the commands.

artadmin commands

Commands can be entered either by full name or abbreviation (as given in the parenthese), followed by any appropriate arguments. Arguments appearing in the square brackets [] are optional, and those in curly braces {} indicate a selection from mutually exclusive options.

To let a set of commands be executed together, the administration comannds entered are kept in the buffer and performed only when the administrator enters a perform commmand.

The commands with their abbreviation and options are described below.

clear (cl)

Resets the commands buffer.

config_update (cu) [on|off] (default is on)

Propagates the configuration changes and requests the application servers to take in the changes in the configuration.

help (h) [command]

Without extra argument: prints a list of all the commands.

With a command argument: prints the synopsis of the command.

list (l)

Displays the commands buffer.

If the buffer is empty, the message " WARNING: No command in buffer." is displayed.

newcopy (n) {p|s} object_name 1 object_name2 ...

Enters a newcopy command for screen or program object types.

perform (p)

Performs the commands submitted to the server and clears the commands buffer.

If the buffer is not empty, the buffer container is displayed and a confirmation is required.

If the submission fails, the message " Perform cancelled." is displayed, and the error is logged into the USERLOG.

quit (q)

Quits this session.

If the buffer is not empty, the buffer container is displayed and a confirmation is required.

sysid (s) {*:SSSS}

By default, the administration commands are transmitted to all servers in the ART CICS system. The configuration is global. For the newcopy, you may want to limit it to some specific servers. The `sysid` command is used to limit the command effect to the servers with a specific SYSID.

Limitations

- The resources which are taken in account on a dynamic reloading operation are limited to transactions, programs, and mapsets.
- The `transactions.des TRANCLASS` parameter cannot be changed dynamically.

System Transactions

Authentication Transactions

CESN

The CESN transaction uses `MAPSET CSIGNON`. The following `MAPSET` definition must be added to the `MAPSET` configuration file `${KIXCONFIG}/mapsets.desc` if CESN transaction is required:

```
[mapset]
name=CSIGNON
filename="<${KIXDIR}>/sysmap/csignon.mpdef"
```

Using this default `MAPSET` definition, CESN supports a maximum eight-character password. If the following MAPSET definition is added to the `MAPSET` configuration file, CESN allows a maximum 32-characters password.

```
[mapset]
name=LSIGNON
filename="<${KIXDIR}>/sysmap/lsignon.mpdef"
```

If two `MAPSET` definitions are both added to the `MAPSET` configuration file, the default `MAPSET` definition `CSIGNON` is used. CESN, in this case, allows a maximum eight-character password.

The CESN transaction ignores the `UCTRAN` setting in the `TYPETERM` configuration file. The username and password entered from terminal are always case-sensitive, no matter which `UCTRAN` value is set.

CESF

No special configuration is required for CESF transaction.

CSGM

Oracle Tuxedo Application Runtime for CICS provides a default "Good Morning" transaction CSGM, which can be added to the Transaction configuration file `${KIXCONFIG}/transactions.desc`.

The default CSGM transaction uses MAPSET ABANNER. So the following MAPSET definition must be added to the MAPSET configuration file if the default CSGM transaction is configured:

```
[mapset]
name=ABANNER
filename="<${KIXDIR}>/sysmap/abanner.mpdef"
```

CHAPTER 11

CICS Runtime Preprocessor

Overview

Definition

The CICS Runtime Preprocessor prepares COBOL programs to execute underOracle Tuxedo Application Runtime for CICS.

Pre-Requisites

The programs handled by the CICS preprocessor must respect the following conditions, or else run the risk of producing errors at compile- or -run-time.

Note: These conditions are ensured by the COBOL translator for programs migrated from the original source platform, but you have to enforce them yourself for maintained or newly-developed programs.

1. CICS Runtime RunTime must be installed. Some technical copy files used by `prepro-cics.pl` are delivered under cpylib CICS Runtime RunTime module.
2. The environment variable `COBCPY`, which indicates to the Micro Focus COBOL Compiler-or Cobol IT compiler where copybooks are stored, must be correctly set to include CICS Runtime RunTime copy files (cpylib) during compilation time.
3. The following copy files must be inserted in the Working-Storage Section or the Local-Storage Section:
   - `KIX--INDICS` and `KIX--ALL-ARGS`, always;

  - KIX--CONDITIONS, always;
  - KIX--DFHRESP, always;
  - KIX--DFHVALUE, if the DFHVALUE pseudo-function is used in the program or one of the copy files it includes;

4. The following copy files must be inserted in the Linkage Section:
   - DFHEIBLK
5. The program must take exactly two parameters, DFHEIBLK (defined by the copy file of the same name) and DFHCOMMAREA, defined as suitable for the application PROCEDURE DIVISION. In other words, the program must look like this:

```
LINKAGE SECTION.
  COPY  DFHEIBLK.
 01  DFHCOMMAREA.
     ....
PROCEDURE DIVISION USING DFHEIBLK DFHCOMMAREA.
```

The case of programs compiled with the NOLINKAGE option of the IBM CICS preprocessor is not (yet) supported by ART and the CICS Runtime preprocessor.

prepro-cics.pl

Name

prepro-cics.pl — A function that reads a Cobol program file from standard input, and outputs it with CICS instructions translated on standard output.

Synopsis

```
prepro-cics.pl [-type_output <output type>] [-notrec <notrec behavior>]
```

Description

prepro-cics.pl takes a COBOL program as input, reads it line by line, and output a file with CICS instructions translated.

prepro-cics.pl performs only one pass and processes lines one by one. That is, it reads a line from the standard input, outputs one or several lines (it may output none depending on the output

type), and then reads the next input line. This behavior enables it to be compatible for use as a preprocessor inside a compiler, but prohibits using the same file as input and output. Note that it will output lines at the end of the input file.

The preprocessor expects the input COBOL program to have a 6-column left-margin. The output is in fixed format, or an error message should appear.

Options

notrec

`notrec` specifies the way instructions that are not fully supported are processed. (Some options of the instruction are not recognized, hence the "notrec"). There are two possibilities:

- Stop – (default) means that if the instruction contains non-supported options, then the whole instruction is considered as not supported and translation will fail.
- Warn – means that the instruction is processed normally, but the generated call sets up a flag to signal some options may not be supported.

In both cases, a message is displayed on the error output.

type_output

`type_output` determines the way that output is printed; recognized values are:

debug
: Prints every line with its status (untouched, modified, deleted, created). Always outputs at least one line for every line read.

orig
: Prints every line, deleted lines are printed as comments. Always outputs at least one line for every line read.

normal (default)
: Prints every line, except deleted ones. Does not always output at least one line for every line read.

Any other value will be considered as "normal".

Restrictions

- The preprocessor expects the input COBOL program to be in fixed format.
- The preprocessor ignores copies. Any CICS inside a copy will not be translated.

- The two words EXEC and CICS must be on the same line for a CICS instruction to be recognized.

Error Messages

Invalid CICS Messages

Error messages printed whenever an Invalid CICS instruction is found use the following format:

- Error summary
- Text of the CICS statement (without margins)
- More detailed explanations, if necessary

Summaries may be:

- Instruction invalid (IGNORE and HANDLE instructions),
- No rules matching the fol-lowing instruction,
- Several rules matching the following instruction.

IGNORE and HANDLE instructions messages are quite straightforward:

IGNORE should be constructed with CONDITION.

When no rules match a CICS instruction, the error message explains, for all commands starting by the same keyword, why this command does not fit.

- <command> expects one of <keyword list>, but none is present.
- <command> expects <keyword>, but couldn't find it.
- <command> does not know about <keyword>.
- In <command>, <keyword> expects either: … <keyword> (one of <keyword list>), … but none of them was found.
- In <command>, <keyword> is present and not <keyword> even if they must be used at the same time.
- In <command>, <keyword> and <keyword> cannot be used at the same time.
- Default value of <keyword> is supposed to be computed with value of <keyword>, but its value (<value>) is not a charstring.

If several commands match, the preprocessor lists them all. This will not actually happen, as the preprocessor checks the commands for ambiguity before translation.

Non Supported Error Messages

If a CICS instruction is unknown, or registered as "non supported", or "obsolete", an error message:

```
Instruction non supported
```

is generated.

The same error message is printed if there are some non-recognized keywords in an instruction, when the option notrec is set with value `stop`.

CICS Runtime Messages

Messages

CICS Runtime Messages provide the following information:

- Description: The meaning and context of the message.
- Action: What steps you can take to correct any problems identified.
- See Also: A pointer to related information (not specified for all messages).

Preprocessor Messages

Error Messages

Invalid CICS Messages

Error messages are printed whenever an invalid CICS instruction is found use the following format:

- Error summary.
- Text of the CICS statement (without margins).
- More detailed explanations, if necessary.

Summaries may contain "Instruction invalid" (in the case of IGNORE and HANDLE instructions), "No rules matching the following instruction", "Several rules matching the following instruction".

IGNORE and HANDLE instructions messages are quite straightforward:

"IGNORE should be constructed with CONDITION"

When no rules match a CICS instruction, the error message lists, for all commands starting by the same keyword, why this command does not fit.

- <command> expects one of <keyword list>, but none is present.
- <command> expects <keyword>, but could not find it.
- <command> does not know about <keyword>.
- In <command>, <keyword> expects either: … <keyword> (one of <keyword list>), … but none of them were found.
- In <command>, <keyword> is present and not <keyword> even if they must be used at the same time.
- In <command>, <keyword> and <keyword> cannot be used at the same time.
- Default value of <keyword> is supposed to be computed with value of <keyword>, but its value (<value>) is not a charstring.

If several commands match, the preprocessor lists them all. This will not actually happen, as the preprocessor checks the commands for ambiguity before translation.

Other Error Messages

The following error messages occur naturally if the preprocessor is used with the wrong options:

- Cannot open file <file name> means that the CICS instruction file is not present or not readable.
- Cannot open $dir/KIX--***.cpy means that the preprocessor was asked to generate copies in a wrong place (nonexistent or read-only directory…).

Maintenance Messages

These messages are encountered if your CICS instruction file is corrupted.

- <keyword> defined as <Pic clause 1> and <Pic clause 2> in same group.
- <keyword> defined twice in same rule.
- A part of a (|) construct has no required keyword.

- Instruction <instruction name> uses <nb1> keyword(s) but describes <nb2> keyword(s).
- Keyword <keyword> in instruction <instruction name> is not described.
- Instructions <instruction name 1> and <instruction name 2> share all their required keywords.
- Line not recognized.

Configuring Oracle Tuxedo XA Connection to DB2 Using DB2 Connect

This chapter contains the following topics:

Prerequisite

Before you start to configure Tuxedo XA connection to DB2 running on mainframe using DB2 Connect, you need to:

- Install DB2 Connect for Universal System on the machine running Tuxedo.
- Configure TCP/IP communication on both mainframe and the machine running Tuxedo.

DB2 Connect Configuration

This section describes the steps required to configure DB2 Connect on open system to connect to DB2 server running on mainframe. These steps must be performed by users who have the necessary system privileges and special expertise, such as your network or system administrator, or your DB2 administrator.

DB2 Instance Creation

This chapter describes how to use db2icrt to create DB2 instance which will be used to connect to database residing on mainframe. The db2icrt command creates DB2 instances in the instance owner's home directory.

Note: The `DB2 DB2ICRT` command is not available for a non-root installation of DB2 database products on Linux and UNIX operating system.

On Linux or UNIX operating systems, this utility is located in the `DB2DIR/instance` directory, where DB2DIR represents the location where the DB2 Connect is installed. On Windows operating system, this utility is located under the DB2PATH\bin directory where DB2PATH is the location where the DB2 Connect is installed.

- Command Syntax:

  ```
  db2icrt -[ h, d, p, a, s, u ] Instname
  ```

  Example:

  ```
  $DB2DIR/instance/db2icrt -u db2art db2art
  ```

The `db2icrt` command takes the following parameters:

`-h | -?`
: Displays the usage information.

`-d`
: Turns debug mode on. Use this option only when instructed by DB2 database support.

`-a AuthType`
: Specifies the authentication type (SERVER, CLIENT or SERVER_ENCRYPT) for the instance. The default is SERVER.

`-p PortName`
: Specifies the port name or number used by the instance. This option does not apply to client instances.

`-s InstType`
: Specifies the type of instance to create. Use the -s option only when you are creating an instance other than the default associated with the installed product from which you are running db2icrt. Valid values are: Client, standalone, ese or wse.

`-u Fenced ID`
: Specifies the name of the user ID under which fenced user-defined functions and fenced stored procedures will run. The -u option is required if you are not creating a client instance.

`InstName`
: Specifies the name of the instance which is also the name of an existing user in the operating system. This has to be the last argument of the db2icrt command.

DB2 Instance Configuration

This section describes how to use CATALOG command to setup the connection to DB2 server residing on mainframe. You can also use DB2 Client Configuration Assistant (CCA), a graphic user interface tool instead, but it is not covered in this section

DB2 CATALOG

DB2 maintains a set of tables that contain information about the data that DB2 controls. These tables are collectively known as the catalog.

The catalog tables contain information about DB2 objects such as tables, views, and indexes. When you create, alter, or drop an object, DB2 inserts, updates, or deletes rows of the catalog that describe the object.

The DB2 catalog consists of tables of data about everything defined to the DB2 system, including table spaces, indexes, tables, copies of table spaces and indexes, and storage groups. The system database DSNDB06 contains the DB2 catalog.

Before starting this step, you need to check:

- Whether the DB2 server on mainframe is started, and TCP/IP communication is up or not.
- Assign an unused TCP port for the DB2 instance listener and add it into your local configuration "/etc/services" like this:

  db2c_db2art 60000/tcp

Next, use `CATALOG TCP/IP NODE`, `CATALOG DCS DATABASE` and `CATALOG DATABASE` step by step to finish the connection setup.

CATALOG TCP/IP NODE

The `CATALOG TCP/IP NODE` command syntax is as follows:

```
catalog [ ADMIN ] [ TCP/IP protocol ] node [ Node-name ] remote [
Hostname ] server [ Service-name ] with [ comment-string ]
```

Example:

```
db2 catalog tcpip node wasa-host remote wasa server 4001 with "catalog
remote host wasa:4001 to local alias wasa-host"
```

`CATALOG TCP/IP NODE` takes the following parameters:

`ADMIN`
: Specifies that a TCP/IP administration server node is to be cataloged. This parameter cannot be specified if the `SECURITY SOCKS` parameter is specified.

`TCP/IP Protocol`
: Specifies TCP/IP protocol used, could be: TCPIP, TCPIP4, TCPIP6

`Node-name`
: The nodename of the TCPIP, TCPIP4, or TCPIP6 node represents a local nickname you can set for the machine that contains the database you want to catalog. Only specify TCPIP4 when specifying an IPv4 IP address, and only specify TCPIP6 when specifying an IPv6 IP address.

`Hostname`
: The hostname or the IP address of the node where the target database resides. IP address can be an IPv4 or IPv6 address. The hostname is the name of the node that is known to the TCP/IP network. The maximum length of the hostname is 255 characters.

`Service-name`
: Specifies the service name or the port number of the server database manager instance. The maximum length is 14 characters. This parameter is case sensitive.

  If a service name is specified, the services file on the client is used to map the service name to a port number. A service name is specified in the server's database manager configuration file, and the services file on the server is used to map this service name to a port number. The port number on the client and the server must match.

  A port number, instead of a service name, can be specified in the database manager configuration file on the server, but this is not recommended. If a port number is specified, no service name needs to be specified in the local services file.

CATALOG DCS DATABASE

This command stores information about remote host (z/OS or OS/390) or iSeries (OS/400) databases in the Database Connection Services (DCS) directory. These databases are accessed through an Application Requester (AR), such as DB2 Connect. Having a DCS directory entry with a database name matching a database name in the system database directory invokes the specified AR to forward SQL requests to the remote server where the database resides.

Note: If you just need access the data residing on universal system, such as UNIX, Linux, you could use `CATALOG DATABASE` only.

The `CATALOG DCS DATABASE` command syntax is as follows:

`catalog dcs database [ Database-name ]` as `[ Target-database-name ]` with `[comment-string ]`

Example:

`db2 catalog dcs database db2wasa` as `qwa1` with `"catalog the remote host database qwa1 to local db2wasa"`

`CATALOG DCS DATABASE` takes the following parameters:

Database-name
Specifies the alias of the target database to catalog. This name should match the name of an entry in the database directory that is associated with the remote node.

Target-database-name
Specifies the name of the target host or iSeries database to catalog.

DB2 CATALOG DATABASE

This command stores database location information in the system database directory. The database can be located either on the local workstation or on a remote database partition server.

Note: If you need access the data residing on local or remote universal system, such as UNIX, Linux, you could use it.

The `DB2 CATALOG DATABASE` command syntax is as follows:

```
catalog database [ Database-name ] as [ Alias ] at node [ Node-name ]
authentication [ Authentication-type ] with [ comment-string ]
```

Example:

```
db2 catalog database db2wasa at node wasa-host authentication dcs with
"catalog the local db2wasa with dcs authentication type"
```

`DB2 CATALOG DATABASE` takes the following parameters:

`Database-name`
Specifies the name of the database to catalog.

`Alias`
Specifies an alias as an alternate name for the database being cataloged. If an alias is not specified, the database manager uses database-name as the alias.

`Node-name`
Specifies the name of the database partition server where the database being cataloged resides. This name should match the name of an entry in the node directory. If the node name specified does not exist in the node directory, a warning is returned, but the database

is cataloged in the system database directory. The node name should be cataloged in the node directory if a connection to the cataloged database is desired.

`Authentication-type`
: The authentication value is stored for remote databases (it appears in the output from the `LIST DATABASE DIRECTORY` command) but it is not stored for local databases

DB2 START UP

After all the steps of CATALOG, you need to update some database manager configuration and start DB2 Connect.

- Update the SVCENAME of your DB2 instance to the listener added into "/etc/services" before.

  ```
  db2 update dbm cfg using SVCENAME db2c_db2art
  ```

- Set the communication protocol to TCP/IP.

  ```
  db2set DB2COMM=tcpip
  ```

- Configure the auto start and start DB2.

  ```
  db2set DB2AUTOSTART=yes
  db2start
  ```

Oracle Tuxedo Configuration

What follows is the description of the process to configure Oracle Tuxedo for accessing the database residing on mainframe with DB2 Connect. There are some differences based on whether Tuxedo is working with a 64-bit instance of DB2 Database or a 32-bit instance of DB2.

- Set the `DB2INSTANCE` environment variable to reference the instance that contains the databases that you want Tuxedo to use. Set the PATH variable to include the DB2 Connect directories. Confirm the User ID and Password that can connect to the DB2 databases, you could find some example below:

  ```
  export DB2INSTANCE=db2art
  export DB2DIR=/opt/ibm/db2_connect/V9.1
  ```

- Update some database manager configuration parameters accordingly.
  - `Update the tp_mon_name database manager configuration parameter with the value TUXEDO`. This parameter identifies the name of the transaction processing (TP) monitor product being used.

Valid value includes CICS, MQ, CB, SF, TUXDEO, TOPEND, WAS, blank or some other value.

```
db2 update dbm cfg using tp_mon_name TUXEDO
```

– Update the spm_name database manager configuration parameter with the hostname of the machine has DB2 Connect installed. This parameter identifies the name of the sync point manager (SPM) instance to the database manager.

Valid value applies to:

- Database server with local and remote clients
- Database server with local clients
- Partitioned database server with local and remote clients

```
db2 update dbm cfg using spm_name bjaix ("bjaix" is the hostname of
the machine which installs and configures DB2 Connect)
```

– Update the max_connections to be greater than max_coordagents to activate the XA Concentrator. DB2 Connect's connection concentrator technology allows DB2 Connect Enterprise Edition servers to provide support to thousands of users simultaneously executing business transactions, while drastically reducing resources required on the S/390 host or AS/400 database servers.

Note: If the application is accessing data residing on DB2 for z/OS and OS/390(R), DB2 for iSeries, or DB2 for VM&VSE, the DB2 Connect XA concentrator should be required.

You can activate the concentrator feature by setting the value of MAX_CONNECTIONS to any number greater than the default. The default value for MAX_CONNECTIONS is equivalent to the value of MAX_COORDAGENTS. Because each application will have one logical agent, MAX_CONNECTIONS actually controls the number of applications that can be connected to the database instance, while MAX_COORDAGENTS controls the number of inbound connections that can be active at any time. MAX_CONNECTIONS will take a numeric range from MAX_COORDAGENTS up to 64,000. The default number of logical agents is equal to MAX_COORDAGENTS.

```
db2 update dbm cfg using max_connections 500
db2 update dbm cfg using max_coordagents 200
```

- Add a definition for DB2 Connect to the Tuxedo resource manager definition file ($TUXDIR/udataobj/RM). In the examples that follow, UDB_XA is the locally-defined Tuxedo resource manager name for DB2 Connect, and db2xa_switch_std is the DB2-defined name for a structure of type xa_switch_t.

```
# DB2 UDB
UDB_XA:db2xa_switch_std:-L${DB2DIR}/lib -ldb2
```

- Build the Tuxedo transaction monitor server (TMS) for DB2:

  ```
  ${TUXDIR}/bin/buildtms -r UDB_XA -o ${TUXDIR}/bin/TMS_UDB
  ```

- Write your own application servers which accesses the DB2 database residing on mainframe, and you can follow the examples below. The -r option specifies the resource manager name, the -f option specifies the files that contain the application business logic, the -s option specifies the application service names for this server, and the -o option specifies the output server file name.

  ```
  ${TUXDIR}/bin/buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2 -o
  UDBserver
  ```

- Set up the Tuxedo configuration file to reference the DB2 server. In the *GROUPS section of the UBBCONFIG file, add an entry similar to:

  ```
  UDB_GRP   LMID=simp GRPNO=3
  TMSNAME=TMS_UDB TMSCOUNT=2
  OPENINFO="UDB_XA:db=sample,uid=username,pwd=password,tpm=tuxedo"
  ```

  Where the TMSNAME parameter specifies the transaction monitor server that you built previously, and the OPENINFO parameter specifies the resource manager name. This is followed by the database name, and the DB2 database user ID and password, which are used for authentication.

  The application servers that you built previously are referenced in the *SERVERS section of the Tuxedo configuration file.

- Start Tuxedo application.

  ```
  tmboot -y
  ```

Summary

This chapter summarizes all the steps needed and provides methods to check the connection and error detection.

1. Firstly, let's recap the step-by-step procedures:

   ```
   db2icrt -a server -u db2art db2art
   ```

   ```
   db2 catalog tcpip node wasa-host remote wasa server 4001
   ```

   ```
   db2 catalog dcs database db2wasa as qwa1
   ```

   ```
   db2 catalog database db2wasa at node wasa-host authentication dcs
   ```

```
db2 update dbm cfg using SVCNAME db2c_db2art
db2set DB2COMM=tcpip
db2set DB2AUTOSTART=yes
db2stop & db2start
```

2. And then, let's check whether the connection is reachable:

```
export DB2INSTANCE=db2art
export DB2BASE=db2wasa
db2 connect to $DB2BASE user user-name using password
```

3. And check the output, if the screen shows some database connection information as below, it means you are able to access the database residing on mainframe, and you can create table or insert data as you required. If screen shows some errors with error code, maybe there are some failures there. Please find the most common errors and its correction in the APPENDIX.

```
Database Connection Information
Database server= DB2 OS/390 9.1.5
SQL authorization ID= BEAUSR1
Local database alias= DB2QWA1
```

4. Finally, let's configure the Tuxedo application and start it up.

```
export TUXDIR=/home/db2art/tuxedo11gR1
export DB2INSTANCE=db2art
export DB2DIR=/opt/ibm/db2_connect/V9.1
db2 update dbm cfg using tp_mon_name TUXEDO
db2 update dbm cfg using spm_name bjaix
db2 update dbm cfg using max_connections 500
db2 update dbm cfg using max_coordagents 200
Add "UDB_XA:db2xa_switch_std:-L${DB2DIR}/lib -ldb2" into
$TUXDIR/udataobj/RM
${TUXDIR}/bin/buildtms -r UDB_XA -o ${TUXDIR}/bin/TMS_UDB
${TUXDIR}/bin/buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2 -o
UDBserver
Configure OPENINFO in ubbconfig:
OPENINFO="UDB_XA:db=sample,uid=username,pwd=password,tpm=tuxedo"
tmboot -y
```

Trouble Shooting

This section lists the most common symptoms of connection problems encountered when using DB2 Connect. In each case, you are provided with:

- A combination of a message number and a return code (or protocol specific return code) associated with that message. Each message and return code combination has a separate heading, and the headings are ordered by message number, and then by return code.
- A symptom is provided, usually in the form of a sample message listing.
- A suggested solution is provided, indicating the probable cause of the error. In some cases more than one suggested solution may be provided.

`SQL1403N`

Symptom: SQL1403N The username and/or password supplied is incorrect.

Solution: User fails to authenticate at the DB2 Connect workstation. Determine whether the user is supposed to be authenticated at the DB2 Connect workstation.

If yes, make sure that the correct password is provided on the CONNECT statement if necessary.

If no, the system database directory entry must have been incorrectly cataloged using `AUTHENTICATION SERVER` (this is the default if `AUTHENTICATION` is not specified explicitly). If this is the case, then re-catalog the entry using `AUTHENTICATION DCS` or `CLIENT`.

Password is not available to send to the target server database. If the system database directory entry is cataloged using `AUTHENTICATION DCS`, then a password has to be flowed from the DB2 Client to the target server database. On certain platforms, for example AIX, the password can only be obtained if it is provided on the `CONNECT` statement.

`SQL5043N`

Symptom: Support for one or more communications protocols failed to start successfully. However, core database manager functionality started successfully.

Perhaps the TCP/IP protocol is not started on the DB2 Connect gateway. There may have been a successful client connection previously

Solution: This warning is a symptom which signals that DB2 Connect, acting as a gateway for remote clients, is having trouble handling one or more client communication protocols. These protocols can be TCP/IP, APPC and others, and usually the message indicates that one of the communications protocols defined to DB2 Connect is not configured properly.

Often the cause may be that the DB2COMM profile variable is not defined, or is defined incorrectly. Generally, the problem is the result of a mismatch between the DB2COMM variable and names defined in the database manager configuration (for example, svcname or tpname).

One possible scenario is having a previously successful connection, then getting the SQL5043 error message, while none of the configuration has changed. This could occur using the TCP/IP protocol, when the remote system abnormally terminates the connection for some reason. When this happens, a connection may still appear to exist on the client, and it may become possible to restore the connection without further intervention by issuing the commands shown below.

Most likely, one of the clients connecting to the gateway still has a handle on the TCP/IP port. On each client machine that is connected to the gateway.

`SQL30061`

Symptom: Connecting to the wrong host or AS/400 database server location - no target database can be found.

Solution: The wrong server database name may be specified in the DCS directory entry. When this occurs, SQLCODE -30061 is returned to the application.

Check the DB2 node, database, and DCS directory entries. The target database name field in the DCS directory entry must correspond to the name of the database based on the platform.

`SQL30081 with Return Code 79`

Symptom: SQL30081N A communication error has been detected.

Communication protocol being used: "TCP/IP". Communication API being used: "SOCKETS".

Location where the error was detected: "". Communication function detecting the error:

```
"connect".  Protocol specific error code(s): "79", "*", "*".
SQLSTATE=08001
```

Solution: This error can occur in the case of a remote client failing to connect to a DB2 Connect gateway. It can also occur when connecting from the DB2 Connect gateway to a host.

The DB2COMM profile variable may set incorrectly on the DB2 Connect gateway. Check this. For example, the command db2set db2comm=tcpip should appear in sqllib/db2profile when running DB2 Extended Enterprise Edition on AIX.

There may be a mismatch between the TCP/IP service name and/or port number specifications at the DB2 client and the DB2 Connect gateway. Verify the entries in the TCP/IP services files on both machines.