Oracle Tuxedo Application Runtime for CICS

Reference Guide

12*c* Release 1 (12.1.1)

September 2013

Oracle Tuxedo Application Runtime for CICS Reference Guide, 12*c* Release 1 (12.1.1)

Contents

CICS Runtime Configuration Files

VSAM Configuration File

Cics Runtime Integration with Non-3270s Terminal

Environment Variables

Server Configuration

Security Configuration

System Commands and Transactions

CICS Runtime Server Build Tool

CICS Commands and Parameters Coverage

CICS Runtime Messages

CICS Runtime Preprocessor

Configuring Oracle Tuxedo XA Connection to DB2 Using DB2 Connect

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
- VSAM Configuration File
- Cics Runtime Integration with Non-3270s Terminal
- Environment Variables
- Server Configuration
- Security Configuration
- CICS Commands and Parameters Coverage
- System Commands and Transactions
- CICS Runtime Server Build Tool
- CICS Runtime Preprocessor
- CICS Runtime Messages
- Configuring Oracle Tuxedo XA Connection to DB2 Using DB2 Connect

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.

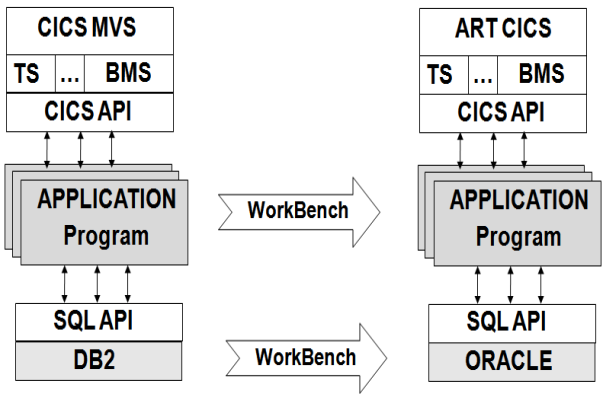
In fact, CICS Runtime gives customers the benefits of Tuxedo distributed architecture without impacting application APIs. It allows the key strengths of Tuxedo to be leveraged and allows routes to the future including SOA to be opened.

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.

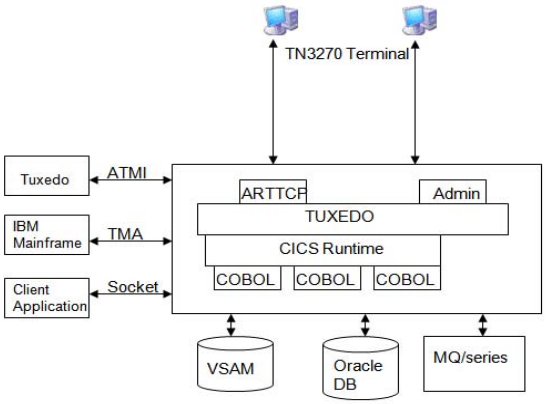
On the target platform, the same philosophy is used; an CICS Runtime Preprocessor (an CICS Runtime compile-time tool) transforms these EXEC CICS into calls to the CICS Runtime library.

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.
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 Cobol

ART CICS Runtime runs in Cobol Micro Focus or Cobol IT.

Micro Focus Cobol is still a reference. The Micro Focus features not supported by Cobol IT cannot be included in this product.

To keep compatible with ART CICS runtime, Cobol IT should be version 3.3.12 and later; CICS programs are required to be recompiled if once compiled by Cobol IT in previous version before.

BDB files can be used with Cobol IT only in XA mode.

Subprogram Calls and Link Level in Micro Focus Cobol

Subprogram working storage is initialized when the subprogram is called for the first time in a link level. After the return link, the working storage is initialized again when the subprogram is called for the first time in a new link at the same level.

For example:

```
LINK (level 1)+
   SP1 first call -(working initialized)
   RETURN
LINK (lev 1)
   SP1 first call -(working initialized)
```

Working storage is not initialized when subprogram is called in lower link levels.

For example:

```
LINK (lev 1)
   SP1 first call -(working initialized)
   LINK (Lev 2)
      SP1 first call -(working not initialized)
```

This behavior is specific to Micro Focus Cobol and not applicable to Cobol IT. On Cobol IT, there is no initialization no matter in which links level subprograms are called.

CICS Runtime C

ART CICS Runtime C support is still COBOL depended. All CICS C programs are required to be complied with COBOL/cob complier and are executed in COBOL runtime. Therefore, COBOL installer is mandatory and currently only Microfocus COBOL is supported.

For more information about CICS Runtime C, please refer to CICS Runtime C Program Support.

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)

Specifically, `ARTCNX` handles the request from `ARTTCPH` and `ARTSTRN/ARTSTR1`, `LUNAME` validation check, and `TERMID/LUNAME` assignment. All services that `ARTCNX` publishes and their functionalities are listed as below.

- `gensess` generates a session ID for each terminal; the session ID is maintained internally in ART CICS.
- `delsess` frees session ID when its terminal terminates.
- `connect` generates a locally unique (unique in each CICS region) `TERMID` and a globally unique `LUNAME` (unique in all CICS regions) for auto-install terminal, checks `LUNAME` validation for the `LUNAME` specified terminal, and changes `TERM` status to `ACQUIRED`.
- `disconnect` frees `TERMID` and `LUNAME`, and change `TERM` status to `RELEASED`.
- `inquire` handles `INQUIRE NETNAME/TERMID` request from `ARTSTRN`.
- `update` handles `SET TERMINAL` request from `ARTSTRN`.
- `authfail` outputs the error message in terminal if `CESN` logon fails.
- `CESN` specifies the Sign oN transaction

- CESF specifies the Sign ofF transaction
- CSGM specifies the Good Morning transaction (default Good Morning transaction)

Note: If ISC_ENABLE=YES is set, gensess and delsess will be published by ARTLOGN instead.

See “Authentication Transactions” in System Commands and Transactions for more information.

Logon Server (ARTLOGN)

This server offers technical services needed by terminal handlers when users log on ART CICS.

This server offers the following services:

ART_LOGON sends the "ART runtime welcome" panel and asks for APPLID input.

gensess generates a globally unique session ID (unique in all CICS regions) with 16 characters for each terminal.

delsess removes the session ID when the corresponding terminal disconnects.

Note: ARTLOGN should be only configured when ISC_ENABLE=YES is specified; otherwise, the server will not boot.

System and Resource Management Server (ARTSRM)

This server centralizes the management of ART runtime information, which is generated and queried by applications.

Note:

- ARTSRM for the same region must be configured in the same tuxedo group.
- ARTSRM for the same region can be only configured on one host in MP environment.
- To get correct running time information, ARTSRM must be configured and started when Oracle Tuxedo is booted.

Synchronous Transaction Servers (ARTSTRN/ARTSTR1)

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.

Tuxedo 12c EM pack monitors ARTTSQ, detailed TSQ properties, and statistics information can be retrieved via EM.

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.

Tuxedo 12c EM pack monitors ARTTDQ, detailed TDQ properties, and statistics information can be retrieved via EM.

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 EXCI interface as services, and manage the execution of these services.

Following are the services ARTDPL publishes:

- <applid>_info: Submits transactions to the CICS region associated with <applid>.
- default_info: This service is called if <applid> is not specified in EXCI interface by the client.
- <sysid>_<program>: This service is advertised as service name.
- <applid>_CSMI: This service is called if <transid> is not specified but <applid> is specified in EXCI interface by the client.
- CSMI: This service is called if both <applid> and <transid> are not specified in EXCI interface by the client.
- <applid>_MIRROR_<transid>: This service is called if both <applid> and <transid> are specified in EXCI interface by the client.
- MIRROR_<transid>: This service is called if <applid> is not specified but <transid> is specified in EXCI interface by the client.

Note:

- <applid>_CSMI, CSMI, <applid>_MIRROR_<transid>, and MIRROR_<transid> are new services introduced since ART CICS rolling patch 015.
- The old services, <applid>_info, default_info, and <sysid>_<program>, are not routed if the user client is based on ART CICS rolling patch 015.

Asynchronous Transaction Servers (ARTATRN/ARTATR1)

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

Note: A request using the EXEC CICS START TRANSID command TERMID option is managed by ARTSTRN/ARTSTR1.

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.

Synchronous Transactions Servers for Non-3270s Terminal Of Tuxedo ATMI Base (ARTWTRN/ARTWTR1)

Description

- Client Description

The non-3270s terminal is different from common 3270 terminal emulator which should follow the 3270 protocol; The non-3270s terminal could be a native Tuxedo client, Java client, and Web UI based on Tuxedo ATMI and should be implemented by customer themselves or 3rd-party.

- Server Description

The task of servers of ARTWTRN/ARTWTR1 is to offer application transactions and process the corresponding programs.

These severs provide a similar functionality to ARTSTRN/ARTSTR1 and present application transactions as Tuxedo services when receiving a transaction request from non-3270s terminals.

Theses servers are non-conversational and thus can manage pseudo-conversational CICS transactions.

Processing

When the server boots, it will advertise each transaction defined in `transactions.desc` as a Tuxedo service. When ARTWTRN/ARTWTR1 receives the request from non-3270s terminals, the server loads corresponding CICS program located at directory `$COBPATH` (`$COB_LIBRARY_PATH` For COBOL-IT), reorganizes the application data received from FML buffer according to related COPYBOOK, and passes data to loaded CICS program for processing. When CICS program returns by invoking CICS RETURN, the server will insert application data into FML buffer and tpreturn to clients.

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.

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
- TS Queue Model Configuration File
- ENQ-Model Configuration File
- TD Queue Extra Partition Configuration File
- TD Queue Intra Partition Configuration File
- Mapset Configuration File
- System Configuration File

- Terminal Configuration File
- Typeterm Configuration File
- Connection Configuration File
- Web Service 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`.

The format of a `list_of_groups.desc` definition is:

```
List_name;group_name
List_name;group_name
… …
List_name;group_name
```

For example,

```
LIST1;SIMPAPP
LIST1;SIMPAPP1
```

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	Reserved for future use. Used to define an alias for the transaction (usually lower case).
CMDSEC	Bool	N|Y	Reserved for future use. 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.
REMOTESYSTEM	X(4)	Optional	Specifies the name that identifies the intercommunication link on which the transaction attach request is sent.

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.

Table 4-4 Programs Parameters

Field Name	Type	Values	Description
LANGUAGE	X(8)	COBOL | C | C++	The language of the program, required to know how to communicate with it. Current release supports COBOL and C.
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.
RESCOUNT-STAT	X(10)	ENABLED| DISABLED	Specifies whether to enable the RESCOUNT information for the specified program. The default value is DISABLED, which is used when this field is not specified.

TS Queue Model Configuration File

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

The filename is `tsqmodel.desc`

Table 4-5 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.
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	Deprecated. There are other ways on target to arrive to the same result. Specifies the 8-character name of the shared TS pool definition that you want to use with this TSMODEL definition.

Table 4-5 TS Queue Parameters

Field Name	Type	Values	Description
REMOTESYSTEM	X(4)	Optional	Reserved for future use. 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.
REMOTEPREFIX XREMOTEPREFIX	X(16)	Optional	Reserved for future use. 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	Reserved for future use. Specifies whether security checking is to be performed for queues matching this model. The default value is N.

ENQ-Model Configuration File

Table 4-6 lists and defines ENQ Models available to be referenced by the CICS application.

The filename is `enqmodel.desc`.

Table 4-6 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.
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	Reserved for future use. E = Enabled. D = Disabled.

TD Queue Extra Partition Configuration File

Table 4-7 lists and defines extra partitions TD queues available to the CICS application.

The filename is `tdqextra.desc`.

Table 4-7 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.

Table 4-7 TD Queue Parameters

Field Name	Type	Values	Description
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.
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 4-7 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 4-7 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.
INTRDR	X(1)	Y | N	Specifies whether to define the TDQ as internal reader for JCL. • Y: The extra TDQ is internal reader for JCL. • N or Not specified: Normal TDQ.
BLOCKFORMAT	X(1)	B | U	Specifies the BLOCK format: • B: Block format • U: Unblocked format

TD Queue Intra Partition Configuration File

Table 4-8 lists and defines intra partitions TD queues available to the CICS application.

The filename is `tdqintra.desc`.

Table 4-8 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 4-8 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.

Table 4-8 TD Queue Parameters

Field Name	Type	Values	Description
ATIFACILITY	X(1)	T	Indicates a TERMINAL when it is set to T.
FACILITYID	X(4)	Optional	The terminal name when ATIFACILITY is set to T.

Mapset Configuration File

Table 4-9 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 4-9 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.

Field Name	Type	Values	Description
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
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 | TRANSIENT	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.

System Configuration File

Table 4-10 lists and defines system available to the CICS application.

The filename is system.desc.

The format of a system.desc definition is:

```
[SYSIDNT]
<field_name_1>=<field_value_1>
<field_name_2>=<field_value_2>
… …
```

For example,

```
[KIXA]
APPLID = DBDCKIXA
INITPARM =(DBCONNA='instance1,database1,artuser1,abc123')
[KIXB]
APPLID = DBDCKIXB
INITPARM =(DBCONNA='instance2,database2,artuser2,123abc')
```

Notes: Blank lines and comment lines are supported. You can use an asterisk in column 1 to code comments, or to remove temporarily an initialization parameter from a particular execution of CICS.

Using a "\", you can continue parameters on another line. Oracle Tuxedo Application Runtime for CICS concatenates the operands, omitting the remaining blanks after "\" in the current line. In this usage, only blank characters can appear after "\" in the current line; otherwise, "\" will be treated as a backslash.

To load and/or discover the defined system initialization parameters, CLOPT "`-s`" is required to specify.

For example,

```
ARTSTRN
CLOPT="-- -s KIXA"
```

In this example, `ARTSTRN` will only load the system section named `KIXA` (`KIXB` will not be loaded).

To define `APPLID` of current CICS region, you can either specify CLOPT "`-a`" (Listing 4-1) or specify `APPLID` parameters in `system.desc` (Listing 4-2).

Listing 4-1 Example to Specify CLOPT "-a"

```
*SERVERS
CLOPT="-- -a abcdefgh"
```

Listing 4-2 Example to Specify APPLID

```
[KIXR]
APPLID=1234567a
```

Oracle Tuxedo Application Runtime for CICS will firstly check whether CLOPT "-a" is specified. If it is, its value will be used as JOBNAME; otherwise, the value of APPLID set in system.desc file will be used as JOBNAME. If neither "-a" nor APPLID in system.desc is set, default value DBDCCICS will be used.

In either ways, if the value is more than 8 characters, it will be truncated to contain only the first 8 characters; if the value is less than 8 characters, APPLID will be padded spaces afterwards to be just 8 characters.

Table 4-10 System Parameters

Field Name	Type	Values	Description
SYSIDNT	X(4)	Mandatory	Name of a system section to define a CICS region. The length of name is 1 - 4 characters. The default name is CICS. The name of a system section and that of the defined CICS region should be the same. **Note:** If more than one sections share a same SYSIDNT, the final configuration for this CICS region will possess all the parameters defined in these sections. If a parameter is defined more than once in these sections, only the last one will make sense.
APPLID	X(8)	DBDCCICS|appid, Optional	Identifies the CICS region in the VTAM network. The length of name is 1 - 8 characters. The default value is DBDCCICS.

Table 4-10 System Parameters

Field Name	Type	Values	Description
INITPARM	X(60)	Mandatory	Specifies initialization parameters in system initialization table (SIT) for each CICS application program that invokes ASSIGN INITPARM. The INITPARM format is INITPARM=(pgmname_1='parmstring_1'[,...,pgmname_n='parmstring_n']). We now support at most 255 sets of program='parmstring'. The value of program should consist of 1 - 8 alphanumeric characters and will be truncated if exceeding the length limit. The value of parmstring should be at most 60 characters and will be truncated if exceeding the length limit. Parameter string should be enclosed by single quotes; any quotes imbedded in the string must be duplicated. **Note:** In each section, if INITPARM parameters are specified more than one time, only the last definition will take effect.
LGNMSG	Bool	NO | YES, Optional	A system initialization parameter which specifies whether logon data is available to an application program.
GMTRAN	X(4)	Optional	A system initialization parameter which specifies the name of the transaction that is initiated by ATI when terminals are logged on to ART CICS. The GMTRAN format is GMTRAN={CSGM|CESN|transaction-id}, where the default value is CSGM. **Note:** To enable GMTRAN, it's also required to specify LOGONMSG=YES in typeterms.desc; however, if LOGONMSG is specified to NO, the screen will be cleared.

Table 4-10 System Parameters

Field Name	Type	Values	Description
BMS	X(60)	DDS|NODDS, Optional	Specifies whether to enable DDS (Device Dependent Support). The default value is DDS, meaning DDS is enabled. With DDS, BMS uses the following rules to select a physical map: • BMS appends the suffix specified in ALTSUFFIX to map set names if the screen size being used is the same as the alternate screen size. In other word, if a transaction has SCRNSIZE(ALTERNATE) specified in the TYPETERM definition, or if the default screen has the same size as the alternate screen, BMS map selection routines try to load the map set with the suffix specified in ALTSUFFIX. • If there is no such map set found, BMS tries to load an unsuffixed map set version.
TYPE	X(8)	ARTCICS | TMASNA	Optional. The default value is ARTCICS.

Note: Section names and keys are case-insensitive; however, values are case-sensitive. The max length of each key is 64 characters and that of each value is 32767 characters.

Terminal Configuration File

Table 4-11 lists and defines `terminal` available to the CICS application.

The filename is `terminals.desc`.

The format of a `terminal` definition is:

```
[terminal]
<field_name_1>=<field_value_1>
<field_name_2>=<field_value_2>
… …
```

For example,

```
[terminal]
name=0001
```

```
netname=CICS0001
katakana=YES
[terminal]
name=0002
netname=CICS0002
katakana=YES
```

ART CICS stores all the terminal runtime status by using Tuxedo /Q. For more information about configuring and using Tuxedo /Q, please see Implementing Asynchronous CICS Delayed Transactions.

Table 4-11 Terminal Parameters

Field Name	Type	Values	Description
ALTSUFFIX	X(1)	0-9|a-z|A-Z, Optional	Specifies a 1-character ALTSUFFIX name for the terminal. Only an alphanumeric character or a blank, which means no suffix, is accepted. The value is case-insensitive and will be converted to upper-case automatically.
KATAKANA	x (3)	NO | YES, Optional	Specifies whether KATAKANA support is required. NO: Default value. KATAKANA function is disabled. YES: KATAKANA function is enabled. Other values: Invalid values. System will automatically search KATAKANA in typeterms.desc.
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.
NAME	X(4)	Mandatory	Terminal identifier. **Note:** When there are multiple entries with same TERMID, the last entry takes effect. If the length exceeds, the value will be ignored and warning message will be logged in ULOG; if the length is shorter than four characters, spaces will be padded on the right.

Table 4-11 Terminal Parameters

Field Name	Type	Values	Description
NETNAME	X(8)	Mandatory	Network name of the terminal. **Note:** The value of NETNAME must be unique; otherwise, ARTCNX server fails to boot and reports the error output in ULOG and stdout. If the length exceeds, the value will be ignored and warning message will be logged in ULOG; if the length is shorter than eight characters, spaces will be padded on the right.
INSERVICE	X(3)	NO | YES, Optional	Specifies the status of the terminal that is being defined. YES: Transactions may be initiated and messages may automatically be sent to the terminal. NO: The terminal can neither receive messages nor transmit input. The default value is YES. If this field is defined to neither of above values, it is treated as YES.

Table 4-11 Terminal Parameters

Field Name	Type	Values	Description
ATI	X(3)	NO | YES, Optional	Specify whether or not the terminal is available for use by transactions that are automatically initiated from within CICS. YES: The terminal is available for use by transactions that are automatically initiated from within CICS. NO: The terminal is not available for use by transactions that are automatically initiated from within CICS. If this field is defined to neither of above values, it is treated as UNKNOWN. However, in the event that the same field in typeterms.desc is defined to UNKNOWN as well, the final value of this field is YES.

Table 4-11 Terminal Parameters

Field Name	Type	Values	Description
TTI	X(3)	NO | YES, Optional	Specifies whether transactions can be initiated at the terminal by a user. YES: Transactions can be initiated at the terminal by a user. If you also specify ATI(YES), transactions can also be initiated automatically. In this case, the automatic transaction initiation, either by transient data control or interval control, sets a condition in an appropriate terminal control table terminal entry. If both ATI and TTI are specified as YES, and if there is no transaction at the terminal, terminal control initiates the user-defined task. This task is expected to send messages to the terminal. For a terminal used in the processing of transactions such as inquiries or order entries, specify TTI(YES) and ATI(NO). This also applies to a display station or hard-copy terminal to which no messages are sent without a terminal request and through which transactions are entered. Note that this is the only specification allowed for 3790 inquiry logical units. NO: Transactions cannot be initiated at the terminal by a user. If you specify NO, specify ATI(YES) to allow transactions to be initiated automatically. An example of this type of terminal is one that is located in a remote location, such as a warehouse, and is unattended but may receive messages. If this field is defined to neither of above values, it is treated as UNKNOWN. However, in the event that the same field in typeterms.desc is defined to UNKNOWN as well, the final value of this field is YES.

Notes: The section name must be set to "terminal"; all keywords are case-insensitive.

ART CICS reserves "."; users can use it neither in NAME nor in NETNAME when defining terminals.desc.

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
katakana=YES
```

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.
ALTSUFFIX	X(1)	0-9|a-z|A-Z Optional	Specifies a 1-character ALTSUFFIX name for the typeterm. Only an alphanumeric character or a blank which means no suffix is accepted. The value is case-insensitive and will be converted to upper-case automatically.
DESCRIPTION	X(79)	Optional	A small textual zone for description of the resource.

Table 4-12 Typeterm Parameters

Field Name	Type	Values	Description
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.
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.

Table 4-12 Typeterm Parameters

Field Name	Type	Values	Description
outline	Bool	NO | YES	Indicates whether the terminal supports field outlining or not.
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.
KATAKANA	x(3)	NO | YES, Optional	Specifies whether KATAKANA support is required. The default value is NO. KATAKANA function can be enabled only if KATAKANA is specified to YES. **Note:** You can either set KATAKANA in terminals.desc or typeterms.desc; however, only if KATAKANA in terminals.desc is not specified to YES or NO, KATAKANA specified in typeterms.desc will take effect.

Table 4-12 Typeterm Parameters

Field Name	Type	Values	Description
ATI	X(3)	NO | YES, Optional	Specifies whether transactions can start at the terminal by automatic transaction initiation. YES: Transactions can start at the terminal by automatic transaction initiation. NO: Transactions cannot start at the terminal by automatic transaction initiation. ATI(YES) allows transactions to be started at the terminal by transient data control or by an EXEC CICS START command issued by another transaction. If there is already a transaction at the terminal, the ATI transaction is held until it ends. If you specify ATI(YES), you must specify an IOAREALEN of at least one byte, except for DEVICE(APPC) when ATI and IOAREALEN have forced default values of YES and 0. If ATI is specified as YES and CREATESESS is specified as YES, and if a transaction is initiated when the terminal is not ACQUIRED, it is automatically acquired. If this field is defined to neither of above values, it is treated as YES.

Table 4-12 Typeterm Parameters

Field Name	Type	Values	Description
TTI	X(3)	NO | YES, Optional	Specifies whether transactions can be initiated at the terminal by a user. YES: Transactions can be initiated at the terminal by a user. If you also specify ATI(YES), transactions can also be initiated automatically. In this case, the automatic transaction initiation, either by transient data control or interval control, sets a condition in an appropriate terminal control table terminal entry. If both ATI and TTI are specified as YES, and if there is no transaction at the terminal, terminal control initiates the user-defined task. This task is expected to send messages to the terminal. For a terminal used in the processing of transactions such as inquiries or order entries, specify TTI(YES) and ATI(NO). This also applies to a display station or hard-copy terminal to which no messages are sent without a terminal request and through which transactions are entered. Note that this is the only specification allowed for 3790 inquiry logical units. NO: Transactions cannot be initiated at the terminal by a user. If you specify NO, specify ATI(YES) to allow transactions to be initiated automatically. An example of this type of terminal is one that is located in a remote location, such as a warehouse, and is unattended but may receive messages. If this field is defined to neither of above values, it is treated as YES.

Connection Configuration File

Table 4-13 lists and defines connections that can be loaded by ART CICS application servers. Connection configuration file is mandatory to set for ARTCTRN but optional to set for other servers.

The filename is connections.desc.

The format of a connection definition is:

```
[CONNID]
<field_name_1>=<field_value_1>
<field_name_2>=<field_value_2>
...
<field_name_n>=<field_value_n>
```

Table 4-13 Connection Parameters

Field Name	Type	Values	Description
CONNID	X(4)	Mandatory	Name of the connection.
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.
PROTOCOL	APPC | LU61	Mandatory	Specifies the type of protocol that implements the connection.
NETNAME	X(8)	Mandatory	Specifies the APPLID of a CICS region, which *receives* the connection.
MAXIMUM	0~999 , 0~999	Mandatory (only for APPC)	Specifies the maximum number of sessions in the modeset. The format is MAXIMUM=value_1,value_2, where Value_1 must be greater than or equal to value_2.
RECEIVECOUNT	999	Mandatory (only for LU61)	Specifies the number of receive sessions; receive sessions can only receive before sending.
SENDCOUNT	999	Mandatory (only for LU61)	Specifies the number of send sessions; send sessions must send before they can receive.

Web Service Configuration File

Table 4-14 defines the web services available to the CICS application.

The filename is webservice.desc.

Table 4-14 Connection Parameters

Field Name	Type	Values	Description
SERVICE	string	Mandatory	Name of the service
REQUEST	string	Mandatory	View name of the request
RESPONSE	string	Mandatory	View name of the response
NEWCPY	string	N | Y Optional	Specifies the type of copybook: • Yes or Y: The copybook is generated by viewc32. • N or Not specified: The copybook is old.
TRANSACTION	string	N | Y Optional	Specifies the service to support or not support the transaction. If this field is not specified, the service does not support the transaction.

VSAM Configuration File

Overview

VSAM configuration file specifies the VSAM accessor and related attribute of each VSAM file used in ART CICS runtime. VSAM configuration file is generated by Oracle Tuxedo Application rehosting Workbench and should not be modified.

ART CICS runtime will find VSAM configuration file through the environment variable `$DD_VSAMFILE`, which could be exported or defined in the envfile and used to indicate the location of VSAM configuration file.

VSAM Configuration Parameters

Table 5-1 lists and defines VSAM file configuration parameters available to be referenced by the CICS application.

The filename is `desc.vsam`.

Table 5-1 VSAM Configuration Parameters

Field Name	Type	Values	Description
NAME	X(8)	Mandatory	Name of the VSAM file. Logical name of VSAM file used in EXEC CICS related to this file.
ACCESSOR	X(256)	Mandatory	Name of the VSAM accessor referenced by CICS application according to the corresponding VSAM file. The VSAM accessor is generated by ART Workbench.
ORGANIZATION	X(1)	Mandatory	I/i: Indexed Record R/r: Relative Record S/s: Sequential Record

Table 5-1 VSAM Configuration Parameters

Field Name	Type	Values	Description
RECORD TYPE	X(1)	Mandatory	F/f: Fixed Length Record I/i: Indefinited Length Record
RECORD LENGTH	NUM 1-32768	Mandatory, if TYPE is Fixed	The length of each fixed record in VSAM file.
KEY POSITION	NUM 1 - 32768	Mandatory, if ORGANIZATION is Indexed	The start position of KEY of each record in VSAM file.
KEY LENGTH	NUM 1 - 256	Mandatory, if ORGANIZATION is Indexed	The length of KEY of each record in VSAM file.

Note: The file status, after CICS Runtime is booted up, is ENABLESTATUS(ENABLED) and OPENSTATUS(OPEN).

Cics Runtime Integration with Non-3270s Terminal

Interface

Non-3270s terminal should submit request and get response through Tuxedo ATMI `tpcall()`. The service name of `tpcall()` should follow the convention described in the following section; in addition, data buffer type exchanged with server ARTWTRN/ARTWTR1 could only be Tuxedo FML32. The detailed FML32 fields and each data structure will be described in the following section.

Service Name Convention

In order to distinguish the service invoked by non-3270s clients from the service invoked by TN3270 terminal, ARTWTRN/ARTWTR1 sever advertises each CICS transaction as a Tuxedo service with the name prefix WEB_. When client submits the request to ARTWTRN/ARTWTR1, the service name of tpcall must follow the convention.

Fml32 Buffer Definition

The interface between non-3270s client and ARTWTRN/ARTWTR1 could only be Tuxedo FML32 buffer. All the application data and some terminal control information will be encapsulated in the FML32 buffer.

Table 5-2 lists all FML32 FIELDs, which will be published in $KIXDIR/include/msgflds32.

Table 5-2FML 32 Buffer Definition for CICS Runtime Integration with Non-3270s Terminal

Field Name	Type	Tuxedo Buffer Type	Length	Comments
CX_WEB_TRANSID	Outbound	CARRAY	4	Transaction name to be invoked. This field will be added/changed by ARTWTRN/1. Client can get the transaction name from this field and use it for the next tpcall.
CX_WEB_MAPSET_NAME	Outbound	CARRAY	7	The name of the current MAPSET, which will be displayed in Client's UI. This field will be added/changed by ARTWTRN/1. Client can get the MAPSET name from this field.
CX_WEB_MAP_NAME	Outbound	CARRAY	7	The name of the current MAP, which will be displayed in Client's UI. This field will be added/changed by ARTWTRN/1. Client can get the MAP name from this field.

Table 5-2FML 32 Buffer Definition for CICS Runtime Integration with Non-3270s Terminal

Field Name	Type	Tuxedo Buffer Type	Length	Comments
CX_WEB_APPDATA	Outbound / Inbound	CARRAY	Depends on Copybook	The application data requiring to be displayed in the Client's UI or processed by transaction. This field will be added/changed/read by both ARTWTRN/1 and Client. When Client receives the FML buffer, it needs to get the application data from this field and split into each map filed according to Copybook. Client could also re-organize latest application data according to Copybook and send it to ARTWTRN/1. When receiving the FML buffer, ARTWTRN/1 will get the application data from this field and pass the data to transaction for processing.
CX_WEB_CTRL_CHAR	Outbound	CARRAY	12	The terminal control characters of the command "CICS SEND", "CICS SEND MAP", "CICS SEND TEXT", and "CICS SEND CONTROL". At times, CICS APIs above will be invoked with some control options, such as FREEKB, ALARM, ERASE, ERASEUP, CURSOR, etc; therefore, these control options should be inserted to this FML field. The field will be added/changed by ARTWTRN/1. Client can get the control options from this field and handle them accordingly.

Table 5-2FML 32 Buffer Definition for CICS Runtime Integration with Non-3270s Terminal

Field Name	Type	Tuxedo Buffer Type	Length	Comments
CX_WEB_AID	Inbound	CARRAY	3	The attention ID and cursor position when doing "CICS RECEIVE" and "CICS RECEIVE MAP". When 3270 terminal sends the application data to the CICS runtime, it will always contain the AID and current cursor position of the screen; therefore, this filed will be added/changed by Client. ARTWTRN/1 will get data from this field and pass the data to transaction for processing.
CX_SESSIONID	Outbound / Inbound	STRING	16	The session ID of client. If it's not filled in client side, ARTWTRN/1 will generate one randomly.
CX_TERMID	Outbound / Inbound	CARRAY	4	The term ID of client. If it's not filled in client side, ARTWTRN/1 will generate one by randomly.
CX_USERID	Outbound / Inbound	CARRAY	8	The user ID of client. If it's not filled in client side, ARTWTRN/1 will fill it with "WEBGUEST".

Note: Outbound means the field is only included in the FML buffer from ARTWTRN/1 to Client; Inbound means the field is only included in the FML buffer from Client to ARTWTRN/1.

CX_WEB_APPDATA

Mainframe CICS provides a utility to generate the Copybook from the Mapset file. The Copybook is used to define the structure of the application data when transaction invokes CICS SEND/RECEIVE MAP.

At ART CICS runtime, we also provide a corresponding utility tcxmapgen, which generates the Copybook from Mapset file; regarding that the structure of Copybook generated by tcxmapgen is almost the same as the one generated in Mainframe CICS, application data exchanged between the Client and ARTWTRN/1 must follow the definition in the Copybook of each Mapset.

For CICS SEND/RECEIVE and CICS SEND TEXT, ARTWTRN/1 and Client just need to insert the plain textual characters into the field CX_WEB_APPDATA.

The head file might not match the copybook generated from BMS file by ART CICS offline utility tcxmapgen at runtime due to data structure alignment on open system. It might need customers to adjust copybook by padding appropriate FILLER.

CX_WEB_CTRL_CHAR

In Mainframe CICS, an outbound data stream is a data stream sent from the application program to the terminal with the following format:

```
|Command|Write Control Character|Indicate Cursor|Orders|Data|Orders
|Data|...
```

CMD defines the function to be performed by the terminal. ARTWTRN/1 will support W and EW. It's a one byte.

WCC instructs how to perform the CMD function. It's a one byte with the following format:

```
| 0 |   1   |    2/3   |   4   |   5   |    6     |   7   |

| * | Reset | Printout | Start | Sound | Keyboard | Reset |

|   |       | Format   | Print | Alarm | Restore  | MDT   |
```

If SEND MAP/TEXT command sets cursor option, IC should be inserted into the header of 3270 data stream with the following format in the case of SEND command without cursor option. If this command sets option ERASE, the cursor will set to the upper left corner of the screen.

```
| SBA | ROW | COL | IC |
```

Besides CMD, WCC, and IC described above, other control characters could be also contained in the header of 3270 data stream, such as EUA.

At ART CICS runtime, we define a FML field CX_WEB_CTRL_CHAR, a CARRAY with 12 bytes, to store these control characters; however, in this release, we plan to support CMD, WCC, and IC, all of which are mostly used. The data sequence of this field is as below:

```
| CMD | WCC | SBA | ROW | COL | IC | ...
```

Only the first 6 characters will be used in this field, the rest 6 characters will be reserved with X'00', and the code of these characters will be set as EBCDIC, not ASCII. SBA, ROW, COL, and IC will be set with X'00' in the buffer if there is no need to set cursor; If IC is

not set via the option cursor of CICS commands, but set in MAPSET/MAP file, no `IC` will be set in this field and customers need to handle `IC` themselves.

`CX_WEB_AID`

In Mainframe CICS, an inbound data stream is sent from the terminal to the application program and contains an attention identifier (AID) followed by cursor or by data directly. The AID describes the action that caused the transmission of the inbound data stream with the following format:

```
| AID | ROW | COL |
```

At ART CICS runtime, we defined a FML field `CX_WEB_AID` to store these three characters and `AID` will always be function key of the keyboard, such as Enter, Clear, F01~F24, etc. `ROW` and `COL` mean the current cursor position when terminal sends the inbound data stream to application program. The code of these characters will be set as EBCDIC, not ASCII.

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
- 1 – transaction / command audit 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 for support team. The higher level covers all traces deliverred by lower level.

KIX_USER_TRACE

This optional variable defines the type of traces, one trace file per connection or all traces in the same file.

KIX_USER_TRACE=SID represents one trace file per connection. The other value is STD, in this case traces will be in each stdout of server, if the variable is not set to STD is assumed.

When you want to use transaction / command audit tool against mulitple homonymous ART servers, SID should be used.

KIX_TRACE_PATH

If KIX_USER_TRACE is set to SID, KIX_TRACE_PATH variable must be set. It defines the directory containing the trace files (the file name is ended by ".trc").

This variable is not used to store other files.

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.

KIX_SPOOL_JOB_AUTO_SUBMIT

This is an optional environment variable for SPOOL functions. Only if KIX_SPOOL_JOB_SUBMIT is not set, KIX_SPOOL_JOB_AUTO_SUBMIT submits the spool files automatically when it is set to YES or yes.

For example: KIX_SPOOL_JOB_AUTO_SUBMIT=yes

COB_ENABLE_XA

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.

DD_VSAMFILE

DD_VSAMFILE is a mandatory environment variable that indicates the location of VSAM configuration file "desc.vsam" generated by ART Workbench, if the CICS program contains some VSAM operations.

ISC_ENABLE

ISC_ENABLE is an optional environment variable. Set ISC_ENABLE=YES to enable the following ISC related features:

- ASSIGN ABDUMP/ABPROGRAM/ASRAINTRPT/ASRAKEY/ASRAPSW/ASRAREGS/ASRASPC/ASRASTG/INITPARM/INITPARMLEN/KATAKANA/NETNAME/SOSI/USERNAME
- CANCEL
- INQUIRE/SET FILE
- INQUIRE PROGRAM RESCOUNT
- INQUIRE SYSTEM JOBNAME
- INQUIRE TERMINAL ALTSUFFIX/USERID
- INQUIRE TRANSACTION PROGRAM
- INQUIRE TERMINAL/NETNAME ACQSTATUS CREATESESS
- ISSUE DISCONNECT
- ISSUE PASS and EXTRACT LOGONMSG
- SEND LAST
- SET/INQUIRE TERMINAL
- SET TERMINAL ACQSTATUS CREATE ACQUIRED RELEASED
- START TRANSID with <ANY> TERMID
- START TRANSID TERMID AFTER
- START TARNSID REQID

- START TRANSID SYSID
- Configurable default transaction - GMTRAN (Good morning transaction)
- 3270 printer support

Note:

- When ISC_ENABLE=YES, ART CICS cannot validate LUNAME across multi-CICS regions. Users must ensure LUNAME is unique across different CICS regions.
- The validation is completed after users log on a CICS region (that is when users pass LOGON screen).

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" 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.

To join 3270 terminal to ART Runtime, you must specify the MAXWSCLIENTS parameter in the MACHINES section of the UBBCONFIG file. MAXWSCLIENTS is the only parameter that has special significance for ARTTCPL. MAXWSCLIENTS tells the Oracle ART at boot time how many accesser slots to reserve exclusively for 3270 terminals.

For MAXWSCLIENTS, specify the maximum number of 3270 terminal that may connect to a node. The default is 0. If not specified, terminal may not connect to the machine being described.

The syntax is MAXWSCLIENTS=number.

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.

Information (including password) exchanged between TN3270 terminal emulator and ARTTCP server is vulnerable unless SSL is enabled.

-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

```
*MACHINES

DEFAULT:

MAXWSCLIENTS = 20

...

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

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 Static 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.

The auxiliary TSQ data is lost after the server is down (e.g. stop or crash).

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 8-1 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.

The extra partition TDQ data is lost after this server is down (e.g. stop or crash).

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.

These programs are screenless programs which cannot interact directly with terminal users.

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 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.

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.

Note: You must configure at least one ARTDPL for each CICS region.

`-L List_name(s)`

Mandatory argument, see Dynamic List of Groups Argument.

`-l GroupList`

Mandatory option, see Dynamic List of Groups Argument.

Note: For both `-l` and `-L` options, if there are multiple ARTDPL in one ART CICS region and `-s` is configured in such region, please set the same value for all `-l` specified in this region, and set the same value for all `-L` specified in this region. For more information about `-L` configurations, please refer to List of Groups Configuration File.

`-S`

Optional argument. ARTDPL servers can receive service requests from Tuxedo client, such as JCA client. In the Tuxedo client, you just need to add the `COMMAREA` data to the FML field `CXMW_MESSAGE`, and then `TPCALL` the corresponding program which is published as a service by ARTDPL. After receiving the service request, ARTDPL executes the corresponding program and finally `TPRETURN` the `COMMAREA` data returned by program with FML field `CXMW_MESSAGE`.

Sometimes, the length of the COMMAREA data returned by program is larger than the length of the COMMAREA data initialized in Tuxedo client. In this case, you need to add one more FML field CXMW_COMMAREAINLENGTH before TPCALL to the Tuxedo client. The CXMW_COMMAREAINLENGTH must be specified the total length of COMMAREA which is initialized in Tuxedo client, and its data type is long. In this way, ARTDPL can TPRETURN COMMAREA data with the total length CXMW_COMMAREAINLENGTH which is definitely larger than the length of the data filled in COMMAREA.
When the length of COMMAREA data returned by program is smaller than the total length CXMW_COMMAREAINLENGTH, ARTPDL normally adds X'00' (null padding character) at the end in the return FML field CXMW_MESSAGE. If you want to strip the NULL padding characters to reduce the data communications between the Tuxedo client and ARTDPL and improve the performance, add "-S" in the CLOPT options of ARTDPL. ARTDPL server will strip the padding characters and then TPRETURN the COMMAREA data with FML field CXMW_MESSAGE.

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=Y 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 (CONVERSE) 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 mandatory to be set as Y for this server type.

minn and maxn

Specify the initial and maximum number of servers to be used to start with this configuration.

Note: If it's required to invoke multiple sessions simultaneously, users should both configure MAXIMUM in connections.desc and configure multiple ARTCTRN servers using ARTCTRN minn and maxn parameters in UBB. For more information, please refer to UBBCONFIG.

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.

-l GroupList
: Mandatory option, see Static 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_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=Y 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=Y 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=Y MIN=1 MAX=1

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

ARTWTRN Configuration

Server Name

ARTWTRN – CICS Runtime application server for synchronous non-3270s clients oriented transactions with MAXACTIVE > 1.

Synopsis

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

Description

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

These servers are non-conversational in order to support pseudo-conversational CICS transactions.

1. When starting, an ARWTRN server publishes one service per transaction it offers.
2. Non-3270s clients submit the request to ARTWTRN via tpcall and then ARTWTRN/1 will receive the request.
3. One ARTWTRN server offering this service receives the request with the associated application data from FML buffer and then processes the transaction.
4. After processing the transaction, on the RETURN {TRANSID}, a reply is sent to the client and the ARTWTRN server finishes the conversation by a tpreturn() to return the application data and TRANSID with FML buffer.
5. If users change the application data and re-submit it to transaction, Non-3270s clients could submit the request to ARTWTRN according the TRANSID received in step 4.

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 an option for this server type; however, it must be defined as `CONV = N` because ARTWTRN is non-conversational.

minn and maxn

Specify the initial and maximum number of servers, respectively. For more information, see 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

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

MIN=2 MAX=3

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

ARTWTR1 Configuration

Server Name

ARTWTR1 – CICS Runtime application server for synchronous non-3270s clients oriented transactions with MAXACTIVE = 1.

Synopsis

```
ARTWTR1 SRVGRP="identifier" SRVID="number" CONV=N MIN=1 MAX=1
CLOPT="[servopts] -- -s System_ID -a Application_ID-L list1:list2,…"
```

Description

Theses servers are a specialized version of ARTWTRN servers presenting only transactions with MAXACTIVE = 1; by contrast, ARTWTRN servers present only transactions with MAXACTIVE > 1.

It is critical and verified by WTR1 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 WTR1 Servers.

The rest of the configuration and behavior of WTR1 servers are exactly the same as WTRN servers.

Examples

```
*SERVERS
```

```
ARTWTR1 SRVGRP="TCPGRP" SRVID=1000 RESTART=Y GRACE=0
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

-t

A string of command-line options that is unique in global scope for ARTCNX and is passed to the ARTCNX when it is booted. The following run-time parameter is recognized:
[-t] (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.

Notes:

- ARTCNX will check whether -t is unique in global scope when booting; if it is not, ARTCNX fails to boot.
- You can use -t to change the auto-assigned TERMID prefix, but make sure the auto-assigned TERMID is not same as the one defined in terminals.desc.

-s

Specifies the SYSID of a CICS region (the region which ARTCNX belongs to). The SYSID is limited to 4 characters and must be unique in each CICS region.

-a

Specifies the APPLID of a CICS region (the region which ARTCNX belongs to). The APPLID is limited to 8 characters.

Note: Users must set either `-s` or `-a` to configure at least one `ARTCNX` for each CICS region to associate the initialization parameters which are defined in `system.desc`; otherwise, if users set `-s` together with `-a`, warning messages will occur in `ULOG`, only `-s` will take effect, and `-a` will be ignored.

-l:

Specifies the resource group where `ARTCNX` should install. The resource is always terminals.

-L:

Specifies the list of resource groups where `ARTCNX` should install.

Note: For both `-l` and `-L` options, if there are multiple ARTCNX in one ART CICS region and `-s` is configured in such region, please set the same value for all `-l` specified in this region, and set the same value for all `-L` specified in this region. For more information about `-L` configurations, please refer to List of Groups Configuration File.

Environment Variables Used

KIX_TRACE_LEVEL

Examples

```
*SERVERS

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

ARTLOGN Configuration

Server Name

`ARTLOGN` – CICS Runtime Logon Server

Synopsis

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

Description

This server offers the following technical services needed by terminal handlers when users log on ART CICS.

ART_LOGON sends the "ART runtime welcome" panel and asks for APPLID input.

gensess generates a globally unique session ID (unique in all CICS regions) with 16 characters for each terminal.

delsess removes the session ID when the corresponding terminal disconnects.

Notes:

- ARTLOGN must be unique in a CICS Runtime system.
- ARTLOGN should be only configured when ISC_ENABLE=YES is specified; otherwise, the server will not boot.

Parameters

CONV

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

Environment Variables Used

ISC_ENABLE

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.

ARTSRM Configuration

Server Name

`ARTSRM` — CICS System and Resource Management Server.

Synopsis

```
ARTSRM SRVGRP="identifier" SRVID="number" MIN=minn MAX=maxn RQADDR=QKIX110
REPLYQ=Y CLOPT="[servopts] -- -s System_ID -a Application_ID -L
list1:list2:… -l GroupList"
```

Description

`ARTSRM` centralizes the management ART runtime information, which is generated and queried by applications. This server is also used for time control options for delaying the start of a transaction.

`ARTSRM` for the same region must be configured in the same tuxedo group.

If the server is rebooted, runtime information will be lost.

Two system servers must be configured in the UBBCONFIG file before ARTSRM can work correctly:

- TMUSREVT

  This server is used to support updating terminals attributes.

  When a terminal updates another terminal attribute, ARTSRM publishes the event to notify the terminal that being updated.

- TMSYSEVT

  This server is used to release the file resource.

  When some servers die, ARTSRM releases the dead server acquired file resource.

Parameters

CLOPT options

The following CLOPT run-time parameters are recognized:

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

-l GroupList
: Mandatory option, see Static 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.

Related Features

It is required to set ARTSRM to enable the following related features.

- ASSIGN ABDUMP/ABPROGRAM/ASRAINTRPT/ASRAKEY/ASRAPSW/ASRAREGS/ASRASPC/ASRASTG/INITPARM/INITPARMLEN/KATAKANA/NETNAME/SOSI/USERNAME
- CANCEL
- INQUIRE/SET FILE
- INQUIRE PROGRAM RESCOUNT
- INQUIRE SYSTEM JOBNAME

- INQUIRE TERMINAL ALTSUFFIX/USERID
- INQUIRE TRANSACTION PROGRAM
- INQUIRE TERMINAL/NETNAME ACQSTATUS CREATESESS
- ISSUE DISCONNECT
- ISSUE PASS and EXTRACT LOGONMSG
- SEND LAST
- SET/INQUIRE TERMINAL
- SET TERMINAL ACQSTATUS CREATE ACQUIRED RELEASED
- START TRANSID with <ANY> TERMID
- START TRANSID TERMID AFTER
- START TARNSID REQID
- START TRANSID SYSID
- Configurable default transaction - GMTRAN (Good morning transaction)
- 3270 printer support

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.

The `appkey` is passed in `AUTH-GROUPID`.

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 9-1 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 [[-d </Q space_name> -q </Q queue_name>] | [-t
<transaction_name>]] [-p <profile>];
```

Parameters

space_name
Specifies the `/Q QSPACE` name.

queue_name
Specifies the `/Q QUEUE` name.

transaction_name
Specifies the transaction to be triggered. If `-d` and `-q` are specified, `-t` is ignored.

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.

System Commands and Transactions

System Commands

Table 1 System Commands

Name	Description
cpy2view32(1)	Generates VIEW32 definition from COPYBOOK
Mif2View32(1)	Generates VIEW32 definition per metadata input file and updates the given FML32 definition file.
tcxcsdcvt (1)	CICS CSD Converter
tcxmapgen(1)	CICS Runtime MAPSET Generator
artadmin (1)	ART CICS Runtime administration
kixrpt (1)	Transaction / Command Audit Tool

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.

Mif2View32(1)

Name

`Mif2View32`—Generates VIEW32 definition file according to the metadata input file and updates the given FML32 definition file.

Synopsis

```
Mif2View32 -i miffile -o viewfile -f fml32file  -w webservice.desc [-v]
```

Description

`Mif2View32` generates VIEW32 definition file according to the metadata input file and updates the given FML32 definition file. The generated nest VIEW32 definition file keeps the same structure with MIF; therefore, we can map the C structure to an FML32 structure.

Options

`Mif2View32` supports the following options.

`-i miffile`
Specifies the metadata input file name.

`-o viewfile`
Specifies the output View32 definition file name.

`-f fml32file`
Specifies the updated FML32 definition file.

`-w webservice.desc`
Specifies `webservice.desc` configuration file.

`-v`
Indicates whether to output details or not.

Example(s)

1. Listing 10-5 provides an MIF example.

Listing 10-5 MIF Example

```
############
# SERVICE: * DFH0XCMNOperation
############
service=DFH0XCMNOperation
servicetype=service
export=Y
inbuf=FML32
outbuf=FML32
tuxservice=DFH0XCMNOperation
```

```
errbuf=FML32
servicemode=webservice
param=DFH0XCMNOperation
count=1
requiredcount=0
fldnum=10025
type=fml32
access=in
paramschema=XSD_E:DFH0XCMNOperation@http://www.DFH0XCMN.DFH0XCP3.Request.com
(
param=ca_request_id
count=1
requiredcount=0
fldnum=10019
type=string
access=in
paramschema=XSD_E:ca_request_id@http://www.DFH0XCMN.DFH0XCP3.Request.com
primetype=string
param=ca_response_message
count=1
requiredcount=0
fldnum=10021
type=string
access=in
paramschema=XSD_E:ca_response_message@http://www.DFH0XCMN.DFH0XCP3.Request.com
primetype=string
```

```
param=ca_inquire_request
count=1
requiredcount=0
fldnum=10009
type=fml32
access=in
paramschema=XSD_E:ca_inquire_request@http://www.DFH0XCMN.DFH0XCP3.Request.
com
(
param=ca_list_start_ref
count=1
requiredcount=0
fldnum=10017
type=short
access=in
paramschema=XSD_E:ca_list_start_ref@http://www.DFH0XCMN.DFH0XCP3.Request.c
om
primetype=unsignedShort
param=ca_cat_item
count=15
requiredcount=0
fldnum=10001
type=fml32
access=in
paramschema=XSD_E:ca_cat_item@http://www.DFH0XCMN.DFH0XCP3.Request.com
(
param=ca_item_ref
count=1
```

```
requiredcount=0
fldnum=10013
type=short
access=in
paramschema=XSD_E:ca_item_ref@http://www.DFH0XCMN.DFH0XCP3.Request.com
primetype=unsignedShort
param=on_order
count=1
requiredcount=0
fldnum=10029
type=short
access=in
paramschema=XSD_E:on_order@http://www.DFH0XCMN.DFH0XCP3.Request.com
primetype=unsignedShort
)
)
)
```

2. Listing 10-6 provides a VIEW32 output file example.

Listing 10-6 VIEW32 Output Example

```
VIEW ca_cat_item_v
#TYPE CNAME FBNAME COUNT FLAG SIZE NULL
short ca_item_ref_v_times ca_item_ref_times 1 - - 0
short ca_item_ref_v ca_item_ref 1 - - 0
short on_order_v_times on_order_times 1 - - 0
short on_order_v on_order 1 - - 0
```

```
END

VIEW ca_inquire_request_v
#TYPE CNAME FBNAME COUNT FLAG SIZE NULL
short ca_list_start_ref_v_times ca_list_start_ref_times 1 - - 0
short ca_list_start_ref_v ca_list_start_ref 1 - - 0
short ca_cat_item_v_times ca_cat_item_times 1 - - 0
struct ca_cat_item_v ca_cat_item 15 - - 0
END

VIEW DFH0XCMNOperation_v
#TYPE CNAME FBNAME COUNT FLAG SIZE NULL
short ca_request_id_v_times ca_request_id_times 1 - - 0
string ca_request_id_v ca_request_id 1 - 255 '\0'
short ca_response_message_v_times ca_response_message_times 1 - - 0
string ca_response_message_v ca_response_message 1 - 255 '\0'
short ca_inquire_request_v_times ca_inquire_request_times 1 - - 0
struct ca_inquire_request_v ca_inquire_request 1 - - 0
END
```

---

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 only COBOL copybook (`.cpy`) output file is generated.

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

`[-m]`
: Specifies that only binary mapset file (`.mpdef`) is produced.

`[-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.

kixrpt (1)

Name

`kixrpt.sh` — `ARTKIX` transaction/command report utility.

Synopsis

`kixrpt.sh [-d "YYYY/MM/DD[hh:mm:ss]"] [-D "YYYY/MM/DD[hh:mm:ss]"] [-c] trace`

Description

kixrpt reports transaction/command timing statistics information, including total number of transactions/commands running in specified time slot, total time cost, the average running time, etc. For each transaction/command, such tool reports the details of running information: total number, total time cost, average time cost, failure count, etc.

kixrpt analyzes the standard output of ARTKIX servers to provide a summary of transaction / command processing time within the servers. The report shows the number of times dispatched and average elapsed time in milliseconds of each transaction / command in the covered period. kixrpt takes its input from the ARTKIX trace file or a directory (like KIX_TRACE_PATH) including trace files. The traces are valid by being set as below:

KIX_TRACE_LEVEL=1 (or higher level)

Options

kixrpt supports the following options:

-d "YYYY/MM/DD[hh:mm:ss]"
: Limits the start time. The default is no limitation.

-D "YYYY/MM/DD[hh:mm:ss]"
: Limits the end time. The default is no limitation.

-c
: Generates the clear trace output instead of statistics report.

Example

1. Set following environment variables before ARTKIX system is started:

   ```
   KIX_TRACE_LEVEL=1
   TP_USER_TRACE=SID
   TRACE_PATH=${APPHOME}/LOGS/traces
   ```

2. After running some time, the statistic tool, kixrpt.sh, can be used to generate the report with given time slot.

   ```
   $ kixrpt.sh -d 2012/06/21 -D 2012/06/22 $TRACE_PATH
   ```

3. Listing 10-7 provides the output example:

Listing 10-7 kixrpt Output Example

```
Transactions:
-------------------------------------------------------------
CPMI(10000)totalTime(2254147)avgTime(225)errNum(0)
-------------------------------------------------------------
totals(10000)totalTime(2254147)avgTime(225)

Commands:
-------------------------------------------------------------
KIX__LINK(10000)totalTime(1047970)avgTime(104)errNum(0)
KIX__DELETEQ_TS(20000)totalTime(9377)avgTime(0)errNum(0)
KIX__RETURN(20000)totalTime(1016)avgTime(0)errNum(0)
KIX__WRITEQ_TS(380000)totalTime(153888)avgTime(0)errNum(0)
KIX__ASSIGN(300000)totalTime(19702)avgTime(0)errNum(0)
-------------------------------------------------------------
totals(730000)totalTime(1231953)avgTime(1)

Programs:
-------------------------------------------------------------
TUX2CXD(10000)totalTime(1047970)avgTime(104)
-------------------------------------------------------------
totals(10000)totalTime(1047970)avgTime(104)
```

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"
```

CICS Runtime Server Build Tool

Overview

Definition

CICS Runtime server build tool constructs one or more Oracle ART CICS server(s) load module. All supported servers which can be built by this build tool are listed in Table 10-1. With a set of prefixes for the server names, Oracle Tuxedo Application Runtime for CICS now allows users to customize server names started with these prefixes. The asterisk (*)character is a wildcard character that can be replaced with zero or more characters. Each prefix stands for one class of servers with special functions as shown on Table 10-1. For example, you can build a server named "ARTSTRN_UDB".

Table 10-1 Servers Need to be Built by CICS Server Build Tool

Server Name	Description
ARTSTR1* ARTSTRN*	The task of these servers is to offer application transactions and process the corresponding programs.
ARTTSQ*	The role of these servers is to centralize the management of the TS Queue operations which are requested by applications.
ARTDPL*	ARTDPL servers publish programs that are callable by EXEC CICS LINK as services and manage the execution of these services.

Table 10-1 Servers Need to be Built by CICS Server Build Tool

Server Name	Description
ARTATR1* ARTATRN*	These servers publish transactions callable by EXEC CICS START TRANSID as services named ASYNC_{Transaction_Name} and manage execution of these services.
ARTCTR1* ARTCTRN*	These servers publish transactions callable by EXEC CICS CONVERSE as services named {SysId}_{Transaction_Name} and manage execution of these services.
ARTWTR1* ARTWTRN*	CICS Runtime application server for synchronous (non-conversational) non-3270s clients.

Administration

To make this tool work correctly, several environment variables must be set firstly as shown in Table 10-2.

Table 10-2 Required Environment Variables

Variable Name	Description
TUXDIR	Mandatory. Install directory of Tuxedo. If it is not set, the tool will stop with a clear error message.
KIXDIR	Mandatory. Install directory of ART CICS Runtime. It's <ART_INSTALL_DIR>/Cics_RT. If it is not set, the tool will stop with a clear error message. Besides, "$KIXDIR/bin" must be included in "$PATH" to execute buildartcics.
COBDIR | COBOLITDIR	Mandatory. Install directory of COBOL compiler (COBDIR is for MF COBOL; COBOLITDIR is for COBOL-IT). One and only one can be set in the environment. If neither or both of them exist, the tool will stop with a clear error message.

Table 10-2 Required Environment Variables

Variable Name	Description
DB2DIR | ORACLE_HOME	Mandatory in some situations. Optional for others. Install directory of database (DB2DIR is for UDB and is mandatory when "-r UDB_XA*" is specified; ORACLE_HOME is for Oracle and is mandatory when "-r Oracle_XA*" is specified). If either of them does not exist when required, the tool will stop with a clear error message.
MT_DB_LOGIN	Mandatory for ARTTSQ* servers built with UDB database. Optional for others. The connect parameter for DB2 database is used in: db2 connect to […] General Syntax: "db-alias user username using passord"
DB2INSTANCE	Mandatory for ARTTSQ* servers built with UDB database. Optional for others. Specifies current DB instance.
MQM	Optional. Stands for the install dir of WebSphere MQ. If not set, it equals to "/usr/mqm" for AIX platforms and "/opt/mqm" for Linux and Solaris platforms.

Notes:

- To use COBOL-IT compiler, please make a copy or create a symbol link of "$COBOLITDIR/bin/cobmf" as "$COBOLITDIR/bin/cob". For example, you can use following command: ln -s $COBOLITDIR/bin/cobmf $COBOLITDIR/bin/cob.
- For Solaris platform, please make sure that $LD_LIBRARY_PATH includes library path for 64-bit "libc*" libraries, which are located in <SUN_STUDIO_INSTALL_DIR>/lib/v9.
- To build ARTTSQ with UDB, please make sure the table "TS_Q_CONTENT" already exists in the Database; if not, you can run "<ARTINSTALL_DIR>/Cics_RT/tools/crtstable_UDB" after all required environment variables are set correctly.

buildartcics

All servers can be built by following synopsis of this tool, except for ARTTSQ that does not support multiple RMs. If "`-M`" is specified when ARTTSQ is built, a warning message will be displayed and only the first "`-r`" will be taken as the resource manager.

Name

`buildartcics` – Constructs one or more Oracle ART CICS server(s) load module.

Synopsis

```
buildartcics [-v] [-M] [-r rmname] –o svrname
```

Description

`buildartcics` is used to construct one or more Oracle ART CICS server(s) load module. This command invokes "`buildserver`" to build the specified ART CICS server(s).

`-v`
: Specifies that `buildartcics` should work in verbose mode. In particular, it writes the compilation command to its standard output.

`-o svrname`
: Specifies the name of the server to load. This option is mandatory. If "`svrname`" is a standalone filename without any "/" character, the server will be generated to `$KIXDIR/bin` by default.

`-M`
: Specifies multiple resource managers associated with this server. This option is mandatory if you want to associate your service with multiple XA complaint resource managers. If this option is not specified and you try to boot a server with a configuration file, in which this server is specified in a non-multiple resource manager server group, a warning message will be printed in the user log; in addition, the sever will revert to a general server associated with one resource manager if the "`-r`" option is specified when `buildserver` command is executed.

`-r rmname`
: Specifies the resource manager associated with this server. Multiple "`-r rmname`" can be specified if "`-M`" is used. The value rmname must appear in the resource manager table located in $TUXDIR/udataobj/RM.

Note: Currently, supported resource managers include `Oracle_XA*`, `UDB_XA*`, `BERKELEY-DB*`, and `MQSeries_XA_RMI*`; therefore, the "`rmname`" must use one of these four strings as a prefix.

Specially, for ARTTSQ* servers, only RMs provided by Oracle or DB2 UDB are supported. If other RMs are specified when ARTTSQ* servers is built, the TSQ data may not be stored correctly. Besides, to build ARTTSQ* servers with "`TMS_UDB`" RM, please make sure that table "TS_Q_CONTENT" exists in the DB2 Database; if this table doesn't exist, you can create it by running "`$KIXDIR/tools/crtstable_UDB`".

Example

```
buildartcics –M –r Oracle_XA –r BERKELEY-DB –o ARTSTRN
```

CICS Commands and Parameters Coverage

CICS Commands and Parameters Coverage

Supported CICS Commands

The following tables describe the CICS commands and parameters that are supported by Oracle Tuxedo ART for CICS.

Note: Commands and parameters not listed in Table 12-1 and Table 12-2 are not supported.

CICS Command and Parameter Support Table

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
ABEND	ABEND	ABCODE(name)	
		CANCEL	
	HANDLE ABEND	CANCEL	
		LABEL(label)	Support HANDLE ABEND generated by command ABEND; partially support system ABEND.
		PROGRAM(name)	
		RESET	Recognized

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
APPC Mapped conversation	ALLOCATE (APPC)	NOQUEUE	Recognized
		STATE(cvda)	
		SYSID(systemname)	
	CONNECT PROCESS	CONVID(name)	
		PROCLENGTH(data-value)	
		PROCNAME(data-area)	
		STATE(cvda)	
		SYNCLEVEL(data-value)	
	CONVERSE (APPC)	CONVID(name)	
		FROM(data-area)	
		FROMLENGTH(data-value)	
		FROMFLENGTH(data-value)	
		INTO(data-area)	
		MAXFLENGTH(data-value)	
		MAXLENGTH(data-value)	
		NOTRUNCATE	
		SET(ptr-ref)	
		STATE(cvda)	
		TOFLENGTH(data-area)	
		TOLENGTH(data-area)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
APPC Mapped conversation	EXTRACT PROCESS	CONVID(name)	
		SYNCLEVEL(data-area)	
	FREE (APPC)	CONVID(name)	
		STATE(cvda)	
	ISSUE CONFIRMATION	CONVID(name)	
		STATE(cvda)	
	RECEIVE (APPC)	CONVID(name)	
		FLENGTH(data-area)	
		INTO(data-area)	
		LENGTH(data-area)	
		MAXFLENGTH(data-value)	
		MAXLENGTH(data-value)	
		NOTRUNCATE	
		SET(ptr-ref)	
		STATE(cvda)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
APPC Mapped conversation	SEND (APPC)	CONFIRM	
		CONVID(name)	
		FLENGTH(data-value)	
		FROM(data-area)	
		INVITE	
		LAST	
		LENGTH(data-value)	
		STATE(cvda)	
		WAIT	
	WAIT CONVID (APPC)	CONVID(name)	
		STATE(cvda)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
BMS	PURGE MESSAGE		Not supported by ARTWTRN/ARTWTR1
	RECEIVE MAP	FROM(data-area)	Not supported by ARTWTRN/ARTWTR1
		INTO(data-area)	
		LENGTH(data-value)	Not supported by ARTWTRN/ARTWTR1
		MAP(name)	
		MAPSET(name)	
		SET(ptr-ref)	Not supported by ARTWTRN/ARTWTR1
		TERMINAL	
	SEND CONTROL	ACCUM	Not supported by ARTWTRN/ARTWTR1
		ALARM	
		CURSOR(data-value)	
		DEFAULT	Not supported by ARTWTRN/ARTWTR1
		ERASE	
		ERASEAUP	Not supported by ARTWTRN/ARTWTR1
		FREEKB	
		FRSET	
		PRINT	
		TERMINAL	Not supported by ARTWTRN/ARTWTR1
		WAIT	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
BMS	SEND MAP	ACCUM	Not supported by ARTWTRN/ARTWTR1
		ALARM	
		CURSOR(data-value)	
		DATAONLY	Not supported by ARTWTRN/ARTWTR1
		DEFAULT	
		ERASE	
		ERASEAUP	
		FREEKB	
		FROM(data-area)	
		FRSET	
		LENGTH(data-value)	
		MAP(name)	
		MAPONLY	Not supported by ARTWTRN/ARTWTR1
		MAPSET(name)	
		PRINT	
		NOFLUSH	Not supported by ARTWTRN/ARTWTR1
		TERMINAL	
		WAIT	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
BMS	SEND PAGE	RELEASE	Not supported by ARTWTRN/ARTWTR1
		RETAIN	
		TRAILER(data-area)	
		TRANSID(name)	
	SEND TEXT	ACCUM	Not supported by ARTWTRN/ARTWTR1
		ALARM	
		CURSOR(data-value)	
		ERASE	
		FREEKB	
		FROM(data-area)	
		HEADER(data-area)	Not supported by ARTWTRN/ARTWTR1
		JUSTIFY(data-value)	
		LENGTH(data-value)	
		NLEOM	
		PRINT	
		TRAILER(data-area)	Not supported by ARTWTRN/ARTWTR1
		TERMINAL	
		WAIT	
Built-in Functions	BIF DEEDIT	FIELD(data-area)	
		LENGTH(data-value)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
Channel	DELETE CONTAINER (CHANNEL)	CHANNEL(data-value)	
	ENDBROWSE CONTAINER	BROWSETOKEN(data-value)	
	GET CONTAINER (CHANNEL)	CHANNEL(data-value)	
		FLENGTH(data-area)	
		INTO(data-area)	
		NODATA	
		SET(ptr-ref)	
	GETNEXT CONTAINER	BROWSETOKEN(data-value)	
	MOVE CONTAINER (CHANNEL)	AS(data-value)	
		CHANNEL(data-value)	
		TOCHANNEL(data-value)	
	PUT CONTAINER (CHANNEL)	CHANNEL(data-value)	
		FLENGTH(data-value)	
		FROM(data-area)	
	STARTBROWSE CONTAINER	BROWSETOKEN(data-area)	
		CHANNEL(data-value)	
	START CHANNEL	CHANNEL(name)	
Console Support	WRITE OPERATOR	TEXT(data-value)	We provide a stub function. This function can be replaced by the integration team for project needs.
		TEXTLENGTH(data-value)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
Diagnostic Services	DUMP TRANSACTION		All its options are recognized by ART CICS.
	ENTER TRACENUM		• All its options are recognized by ART CICS. • ENTER TRACENUM was written as ENTER TRACEID in Mainframe CICS v3 or earlier.

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
Environment services	ADDRESS	COMMAREA(ptr-ref)	
		CWA(ptr-ref)	
		EIB(ptr-ref)	
		TCTUA(ptr-ref)	
		TWA(ptr-ref)	
	ADDRESS SET	SET(data-area/ptr-ref)	
		USING(ptr-ref/data-area)	
	ASSIGN	ABCODE(data-area)	
		ABDUMP(data-area)	
		ABPROGRAM(data-area)	
		APPLID(data-area)	
		ASRAINTRPT(data-area)	Recognized
		ASRAKEY(cvda)	Recognized
		ASRAPSW(data-area)	Recognized
		ASRAREGS(data-area)	Recognized
		ASRASPC(cvda)	Recognized
		ASRASTG(cvda)	Recognized
		CHANNEL(data-area)	
		CWALENG(data-area)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
Environment services	ASSIGN	INITPARM(data-area)	In each section, if INITPARM parameters are specified more than one time, only the last definition will take effect.
		INITPARMLEN(data-area)	
		KATAKANA(data-area)	
		NETNAME(data-area)	
		OPID(data-area)	Recognized
		PROGRAM(data-area)	
		SOSI(data-area)	
		STARTCODE(data-area)	
		SYSID(data-area)	
		TCTUALENG(data-area)	
		TERMCODE(data-area)	Recognized
		TWALENG(data-area)	
		USERID(data-area)	
		USERNAME(data-area)	
Exception Support	HANDLE CONDITION	condition(label)	
	IGNORE CONDITION	condition	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
File control services	DELETE	DATASET(filename)	
		FILE(filename)	
		GENERIC(VSAM KSDS only)	Not supported in File2File when using COBOL-IT.
		KEYLENGTH(data-value)	
		NUMREC(data-area)(VSAM KSDS only)	Recognized
		RBA	• Not supported in File2File when using COBOL-IT. • DELETE RBA can only be used in KSDS dataset.
		RIDFLD(data-area)	
		SYSID(systemname)	Recognized
	ENDBR	DATASET(filename)	
		FILE(filename)	
		REQID(data-value)	Recognized
		SYSID(systemname)	Recognized

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
File Control Services	READ	DATASET(filename)	
		EQUAL	
		FILE(filename)	
		GENERIC	Not supported in File2File when using COBOL-IT.
		GTEQ	
		INTO(data-area)	
		KEYLENGTH(data-value)	
		LENGTH(data-area)	
		RBA	Not supported in File2File when using COBOL-IT.
		RIDFLD(data-area)	
		RRN	Recognized
		SET(ptr-ref)	
		SYSID(systemname)	Recognized
		UPDATE	Recognized

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
File Control Services	READNEXT	DATASET(filename)	
		FILE(filename)	
		INTO(data-area)	
		KEYLENGTH(data-value)	
		LENGTH(data-area)	
		RBA	Not supported in File2File when using COBOL-IT.
		RIDFLD(data-area)	
		RRN	Recognized
		SET(ptr-ref)	
		SYSID(systemname)	Recognized
	READPREV	DATASET(filename)	
		FILE(filename)	
		INTO(data-area)	
		KEYLENGTH(data-value)	
		LENGTH(data-area)	
		RBA	Not supported in File2File when using COBOL-IT.
		RIDFLD(data-area)	
		RRN	Recognized
		SET(ptr-ref)	
		SYSID(systemname)	Recognized

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
File Control Services	RESETBR	DATASET(filename)	
		EQUAL	
		FILE(filename)	
		GENERIC	Not supported in File2File when using COBOL-IT.
		GTEQ	
		KEYLENGTH(data-value)	
		RBA	Not supported in File2File when using COBOL-IT.
		RIDFLD(data-area)	
	REWRITE	DATASET(filename)	
		FILE(filename)	
		FROM(data-area)	
		LENGTH(data-value)	
		SYSID(systemname)	Recognized

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
File Control Services	STARTBR	DATASET(filename)	
		EQUAL	
		FILE(filename)	
		GENERIC	Not supported in File2File when using COBOL-IT.
		GTEQ	
		KEYLENGTH(data-value)	
		RBA	Not supported in File2File when using COBOL-IT.
		REQID(data-value)	Recognized
		RIDFLD(data-area)	
		RRN	Recognized
		SYSID(systemname)	Recognized
	UNLOCK	DATASET	Recognized
		FILE(filename)	
		TOKEN(data-area)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
File Control Services	WRITE	DATASET(filename)	
		FILE(filename)	
		FROM(data-area)	
		KEYLENGTH(data-value)	
		LENGTH(data-value)	
		RBA	• Not supported in File2File when using COBOL-IT. • WRITE RBA can only be used in ESDS dataset.
		RIDFLD(data-area)	
		RRN	Recognized
		SYSID(systemname)	Recognized
Interval Control Services	ASKTIME	ABSTIME(data-area)	
	CANCEL	REQID(name)	Not supports for cancelling the remote requests issued by DELAY command.
		SYSID(systemname)	
		TRANSID(name)	
Interval Control Services	DELAY	FOR	
		HOURS(data-value)	
		INTERVAL(hhmmss)	The default value is INTERVAL(0).
		MINUTES(data-value)	
		REQID(name)	
		SECONDS(data-value)	
		TIME(hhmmss)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
Interval Control Services	FORMATTIME	ABSTIME(data-area)	
		DATE(data-area)	
		DATEFORM(data-area)	
		DATESEP(data-value)	
		DAYCOUNT(data-area)	
		DAYOFMONTH(data-area)	
		DAYOFWEEK(data-area)	
		DDMMYY(data-area)	
		DDMMYYYY(data-area)	
		FULLDATE(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)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
Interval Control Services	RETRIEVE	INTO(data-area)	
		LENGTH(data-area)	
		QUEUE(data-area)	
		RTERMID(data-area)	
		RTRANSID(data-area)	
		SET(ptr-ref)	
Interval Control Services	START	AFTER	
		AT	
		FROM(data-area)	
		HOURS(data-value)	
		INTERVAL(hhmmss)	
		LENGTH(data-value)	If not specified, START LENGTH will be automatically set. Its default value is in "LENGTH OF data-area" format; "data-area" is the one specified in FROM.
		MINUTES(data-value)	
		NOCHECK	Recognized
		PROTECT	
		QUEUE(name)	This option cannot be used with TERMID.
		REQID(name)	
		RTERMID(name)	
		RTRANSID(name)	
		SECONDS(data-value)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
Interval Control Services	START	SYSID(systemname)	
		TERMID(name)	The Oracle Tuxedo server, TMUSREVT, must be configured in the UBBCONFIG file to support this command.
		TIME(hhmmss)	
		TRANSID(name)	TERMID option cannot be used together with the RTRANSID/RTERMID/QUEUE option for the START TRANSID command.
		USERID(data-value)	The security stub is called with USERID and TRANSID.

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
LUTYPE6.1 Conversation	ALLOCATE (LUTYPE6.1)	SYSID(systemname)	
	CONVERSE (LUTYPE6.1)	FROM(data-area)	
		FROMFLENGTH(data-value)	
		FROMLENGTH(data-value)	
		INTO(data-area)	
		MAXFLENGTH(data-value)	
		MAXLENGTH(data-value)	
		NOTRUNCATE	
		SESSION(name)	
		SET(ptr-ref)	
		TOFLENGTH(data-area)	
		TOLENGTH(data-area)	
	FREE (LUTYPE6.1)	SESSION(name)	
	RECEIVE (LUTYPE6.1)	FLENGTH(data-area)	
		INTO(data-area)	
		LENGTH(data-area)	
		MAXFLENGTH(data-value)	
		MAXLENGTH(data-value)	
		NOTRUNCATE	
		SESSION(name)	
		SET(ptr-ref)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
LUTYPE6.1 Conversation	SEND (LUTYPE6.1)	FLENGTH(data-value)	
		FROM(data-area)	
		INVITE	
		LAST	
		LENGTH(data-value)	
		SESSION(name)	
		WAIT	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
Program Control	LINK	CHANNEL(name)	
		COMMAREA(data-area)	
		LENGTH(data-value)	
		PROGRAM(name)	
		SYNCONRETURN	
		SYSID(systemname)	
	RETURN	CHANNEL(name)	
		COMMAREA(data-area)	
		IMMEDIATE	
		INPUTMSG(data-area)	
		INPUTMSGLEN(data-value)	
		LENGTH(data-value)	
		TRANSID(name)	
	XCTL	CHANNEL(name)	
		COMMAREA(data-area)	
		LENGTH(data-value)	
		PROGRAM(name)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
Spool Interface (JES)	SPOOLCLOSE	DELETE	Recognized
		TOKEN(data-area)	
	SPOOLOPEN OUTPUT	ASA	Recognized
		CLASS(data-value)	
		NOCC	
		NODE(data-value)	Only support for local node if there is no /*XEQ NODE line defined in JCL file for the automatic submission to JES.
		PUNCH	Recognized
		RECORDLENGTH(data-value)	
		TOKEN(data-area)	
		USERID(data-value)	Supports only INTRDR
	SPOOLWRITE	FLENGTH(data-value)	
		FROM(data-area)	
		TOKEN(data-area)	
Storage Control	GETMAIN	FLENGTH(data-value)	
		INITIMG(data-value)	
		SET(ptr-ref)	
	FREEMAIN	DATA(data-area)	
		DATAPOINTER(ptr-value)	
Syncpoint	SYNCPOINT		
	SYNCPOINT ROLLBACK	ROLLBACK	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
System Commands	INQUIRE CONNECTION	ACCESSMETHOD(cvda)	Only VTAM is returned.
		CONNSTATUS(cvda)(APPC and MRO only)	Only ACQUIRED is returned. CONNSTATUS equals to ACQSTATUS.
		NETNAME(data-area)	
	INQUIRE NETNAME		• For all other supported INQUIRE NETNAME options, please refer to INQUIRE TERMINAL. • Only support for terminals in the same CICS region.
	INQUIRE SYSTEM	JOBNAME(data-area)	
		SHUTSTATUS(cvda)	Only NOTAPPLIC is returned.
	INQUIRE TERMINAL	ACCESSMETHOD(cvda)	• Partial CVDA values are supported for the following options: – ACCESSMETHOD: VTAM – CREATESESS: NOCREATE – REMOTESYSTEM: Blank – SERVSTATUS: INSERVICE and OUTSERVICE – TRACING: STANTRACE • Only support for terminals in the same CICS region.
		ALTSUFFIX(data-area)	
		ACQSTATUS(cvda)(VTAM only)	
		CREATESESS(cvda)(VTAM only)	
		NETNAME(data-area)	
		NEXTTRANSID(data-area)	
		REMOTESYSTEM(data-area)	
		SERVSTATUS(cvda)	
		SIGNONSTATUS(cvda)	
		TRACING(cvda)	
		TRANSACTION(data-area)	
		USERID(data-area)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
System Commands	INQUIRE TRANCLASS	MAXACTIVE(data-area)	The following parameters are not applicable in CICS Runtime environment: ACTIVE PURGETHRESH QUEUED
	INQUIRE TRANSACTION	PROGRAM(data-area)	
		STATUS(cvda)	
	SET CONNECTION	CONNSTATUS(cvda)(APPC only)	• Recognized. • CONNSTATUS equals to ACQSTATUS.
	INQUIRE FILE	ACCESSMETHOD(cvda)	Only VSAM is returned.
		DSNAME(data-area)	The default value of DSNAME is the VSAM file name; once specified by SET FILE DSNAME, INQUIRE FILE DSNAME will return the specified value.
		EMPTYSTATUS(cvda)(VSAM only)	Only NOEMPTYREQ is returned.
		ENABLESTATUS(cvda)	
		LSRPOOLID(data-area)(VSAM only)	Only 0 is returned.
		MAXNUMRECS(data-area)(data tables only)	Only 0 is returned.
		OPENSTATUS(cvda)	
		TABLE(cvda)(VSAM and CFDT only)	Only NOTTABLE is returned.
		TYPE(cvda)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
System Commands	INQUIRE PROGRAM	RESCOUNT(data-area)	To use this option, RESCOUNT-STAT must be set to ENABLED in programs.desc.
	SET FILE	DSNAME(data-value)	
		ENABLESTATUS(cvda)	The supported CVDA values are DISABLED and ENABLED.
		OPENSTATUS(cvda)	The supported CVDA values are CLOSED and OPEN.
	SET TERMINAL **Note:** The terminals must be in the same CICS region.	ATISTATUS(cvda)	The supported CVDA values are ATI and NOATI.
		CREATESESS(cvda) (VTAM only)	The supported CVDA value is CREATE.
		SERVSTATUS(cvda)	The supported CVDA values are INSERVICE and OUTSERVICE.
		TERMSTATUS(cvda) (VTAM only)	The supported CVDA values are ACQUIRED and RELEASED.
		TTISTATUS(cvda)	The supported CVDA values are NOTTI and TTI.

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
Task Control	DEQ	LENGTH(data-value)	Mandatory, only the enqueues and dequeues on data values are supported, not the enqueues on address.
		MAXLIFETIME(cvda)	The supported CVDA values are TASK and UOW.
		RESOURCE(data-area)	
	ENQ	LENGTH(data-value)	Mandatory, only the enqueues and dequeues on data values are supported, not the enqueues on address.
		MAXLIFETIME(cvda)	The supported CVDA values are TASK and UOW.
		NOSUSPEND	
		RESOURCE(data-area)	
	SUSPEND		
Temporary Storage	DELETEQ TS	QNAME(name)	
		QUEUE(name)	
		SYSID(systemname)	Recognized

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
Temporary Storage	READQ TS	INTO(data-area)	
		ITEM(data-value)	
		LENGTH(data-area)	
		NEXT	
		NUMITEMS(data-area)	
		QNAME(name)	
		QUEUE(name)	
		SET(ptr-ref)	
		SYSID(systemname)	Recognized
	WRITEQ TS	AUXILIARY	
		FROM(data-area)	
		ITEM(data-area)	
		LENGTH(data-value)	
		MAIN	
		NOSUSPEND	
		NUMITEMS(data-area)	
		QNAME(name)	
		QUEUE(name)	
		REWRITE	
		SYSID(systemname)	Recognized

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
Terminal Control	CONVERSE (terminal)	ASIS	
		ALTERNATE	
		CTLCHAR(data-value)	
		ERASE	
		FROM(data-area)	
		FROMFLENGTH(data-value)	
		FROMLENGTH(data-value)	
		INTO(data-area)	
		MAXFLENGTH	
		MAXLENGTH	
		NOTRUNCATE	
		STRFIELD	
		SET(ptr-ref)	
		TOFLENGTH(data-area)	
		TOLENGTH(data-area)	
	EXTRACT LOGONMSG	INTO(data-area)	
		LENGTH(data-area)	
		SET(ptr-ref)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
Terminal Control	HANDLE AID	ANYKEY	
		CLEAR	
		ENTER	
		OPERID	Recognized
		PA1-PA3	
		PF1-PF24	
	ISSUE DISCONNECT (default)		
	ISSUE PASS	FROM(data-area)	
		LENGTH(data-value)	
		LUNAME(name)	
	RECEIVE	BUFFER	Not supported by ARTWTRN/ARTWTR1
		FLENGTH(data-value)	
		INTO(data-area)	
		LENGTH(data-value)	
		MAXFLENGTH(data-value)	Not supported by ARTWTRN/ARTWTR1
		MAXLENGTH(data-value)	
		NOTRUNCATE	Not supported by ARTWTRN/ARTWTR1
		SET(ptr-ref)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
Terminal Control	SEND	ALTERNATE	
		CTLCHAR(data-value)	When bit 2/3 of WCC is '00' and bit 4 is '1', the LENGTH and FLENGTH must not be greater than 1896.
		DEFRESP	
		ERASE	
		FLENGTH(data-value)	Not supported by ARTWTRN/ARTWTR1
		FROM(data-area)	
		LAST	
		LENGTH(data-value)	
		STRFIELD	
		WAIT	Not supported by ARTWTRN/ARTWTR1
Transient Data	DELETEQ TD	QUEUE(name)	
		SYSID(systemname)	
	READQ TD	INTO(data-area)	
		LENGTH(data-area)	
		QUEUE(name)	
	WRITEQ TD	FROM(data-area)	
		LENGTH(data-value)	
		QUEUE(name)	
		SYSID(systemname)	

Table 12-1 CICS Command (COBOL Support)

Category	CICS Command	Command Parameter	Notes
Web Service	INVOKE WEBSERVICE	CHANNEL(name)	
		OPERATION(data-area)	
		URI(data-area)	
		WEBSERVICE(name)	

Note:

- For each command, if none of its listed "command parameters" is mandatory to set originally, ART CICS supports users to code none of them.
- RESP, RESP2, and NOHANDLE are supported on all commands.
- Recognized parameters are processed by the pre-processor and have no effect on the behavior of CICS Runtime.

Table 12-2 CICS Command (C Support)

Category	CICS Command	Command Parameter	Notes
Environment services	ADDRESS	EIB (ptr-ref)	
		COMMAREA (ptr-ref)	
Interval Control Services	RETRIEVE	INTO (data-area)	
		LENGTH (data-value)	
Program Control	LINK	COMMAREA(data-area)	
		LENGTH(data-value)	
		PROGRAM(name)	
	RETURN		
Terminal Control	RECEIVE	INTO (data-area)	
		LENGTH (data-value)	

Table 12-2 CICS Command (C Support)

Category	CICS Command	Command Parameter	Notes
Transient Data	WRITEQ TD	FROM(data-area)	
		LENGTH(data-value)	
		QUEUE(name)	

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 12-1 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 12-2 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 12-3 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 12-4 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 EIB Fields

The following table describes the EIB fields that are supported by Oracle Tuxedo ART for CICS.

Table 12-3 Supported EIB Fields

EIB fields	Notes
EIBAID	
EIBATT	Recognized
EIBCALEN	
EIBCOMPL	Recognized
EIBCONF	Recognized
EIBCPOSN	
EIBDATE	
EIBDS	

Table 12-3 Supported EIB Fields

EIB fields	Notes
EIBEOC	Recognized
EIBERR	Recognized
EIBERRCD	Recognized
EIBFMH	Recognized
EIBFN	
EIBFREE	Recognized
EIBNODAT	Recognized
EIBRCODE	
EIBRECV	
EIBREQID	
EIBRESP	
EIBRESP2	
EIBRLDBK	Recognized
EIBRSRCE	
EIBSIG	Recognized
EIBSYNC	Recognized
EIBSYNRB	Recognized
EIBTASKN	
EIBTIME	
EIBTRMID	
EIBTRNID	

Supported BMS Macros

The following describes the BMS Macros that are supported by Oracle Tuxedo ART 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)

```
Psid
```

TERM=3270-2

SUFFIX=(One alphanumeric character or blank)

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)

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

CASE=MIXED

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)

```
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
```

Supported ECI C API Parameters

The following table describes the ECI C API parameters that are supported by Oracle Tuxedo ART for CICS.

Notes: Only ECI v1 API `CICS_ExternalCall` (`ECI_Parms`) is supported.

Table 12-4 ECI C API Parameters

Call Type	Parameter	Notes
ECI_SYNC / ECI_ASYNC	eci_call_type	
	eci_program_name	eci_program_name is the program name defined in programs.desc in ART CICS.
	eci_userid	If security is not enabled on ART CICS side, eci_userid is not checked on ECI Emulator.
	eci_password	If security is not enabled on ART CICS side, eci_password is not checked on ECI Emulator.
	eci_extend_mode	
	eci_luw_token	
	eci_version	
	eci_commarea	
	eci_commarea_length	
	eci_system_name	It cannot be NULL since no default system name is allowed.
	eci_userid2	
	eci_password2	
	eci_timeout	
ECI_GET_REPLY	eci_call_type	
	eci_commarea	
	eci_commarea_length	
	eci_version	

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.

ARTDPL Messages

Abend messages

When abend occurs, following message is logged:

```
Abend <abend code> detected in transaction <transaction name> program
<program name>
```

The abend code is assigned to `URCODE`. The first character of abend code is assigned to highest byte of `URCODE`, the second character is assigned to the second high byte, the third character is assigned to the third byte, and the last character of abend code is assigned to least byte of `URCODE`. The abend code is also filled in the field `CX_ABENDCODE` of FML buffer.

CICS Runtime Preprocessor

This chapter contains the following sections:

Overview

The CICS Runtime Preprocessor prepares COBOL and C programs to execute under Oracle Tuxedo Application Runtime for CICS.

prepro-cics.pl

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 must be installed. Some technical copy files used by `prepro-cics.pl` are delivered under cpylib CICS 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 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.

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.

In order to support the feature “TSAM call path for CICS and DB”, the preprocessor is asked to be executed ahead of “`procob`” (for Oracle DB) or “`db2 prep`” (for IBM UDB) commands when compiling CICS programs containing “`EXEC SQL`” statements.

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`.

prepro-cics-C.pl

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 Runtime.

1. CICS Runtime must be installed. Some technical header files used by `prepro-cics-C.pl` are delivered under `include` directory in CICS Runtime module.
2. `prepro-cics-C.pl` does not support "TSAM call path for CICS and DB" functions.
3. If the program contains "`EXEC SQL`" statements, the preprocessor is asked to be executed ahead of db pre-complier commands (db2 prep for IBM UDB or pro*C for Oracle DB).

Name

`prepro-cics-C.pl` — A function that reads a C program file specified by `source_file`, and outputs the C program file with CICS instructions translated on `output` file.

Synopsis

```
prepro-cics-C.pl [-help] [-source_format <fixed>] [-o <output file name>]
[-B] source_file
```

Description

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

`prepro-cics-C.pl` performs only one pass and processes lines one by one. That is, it reads a line from `source_file`, outputs one or several lines, 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.

Options

-help

`help` shows the usage of `prepro-cics-C.pl`.

-source_format

`source_format` specifies the source format of C program. There are two possibilities:

- `fixed` - (default) means that each line of a C program should be in fixed format.
- `free` - specifies free style of C program.

Note: Only `fixed` format is supported currently.

-o

`o` specifies output file name and path.

-B

`B` specifies that the source is an EXCI program.

source_file

`source_file` specifies input file name and path.

Restrictions

- `/**/` is used for single line comments. Do not put a comment in the middle of an `EXEC CICS` command.
- Keep `EXEC CICS` as a whole in one line.
- Keep multiple CICS commands in one line is not support.

- `#pragma` will be automatically translated to comments.
- The preprocessor expects the inputted C program to be in fixed format.
- Keep C `main()` function, its parameter list, and parenthesis in one line. For example,

  ```
  void main(int argc, char **argv)
  ```

Error Messages

`prepro-cics-C.pl` may print the following error messages:

- Cannot open `source_file`
- Errors occurred while CICS C code is translated
- `PROGRAM` should be constructed with `EXCI LINK`
- `LENGTH` should be constructed with `COMMAREA` in `EXCI LINK`
- `DATALENGTH` should be constructed with `COMMAREA` in `EXCI LINK`
- `RETCODE` should be constructed with `EXCI LINK`
- No rules match the following instructions:
  - Instruction `rulename` non supported
  - Instruction `rulename` is obsolete

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.