Oracle Tuxedo Application Runtime for CICS

Reference Guide

11*g* Release 1 (11.1.1.1.0)

March 2010

ORACLE

Contents:

4. CICS Runtime Configuration Files

5. Environment Variables

6. Server Configuration

7. /Q Configuration for CICS Runtime

8. Security Configuration

9. CICS Commands and parameter coverage

10. Other Configuration

11. CICS Runtime Preprocessor

12. CICS Runtime Messages

CHAPTER 1

Introduction

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

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

The document includes the following chapters:

- CICS Runtime Concepts
- CICS Runtime Servers
- CICS Runtime Configuration Files
- Environment Variables
- Server Configuration
- /Q Configuration for CICS Runtime
- Security Configuration
- CICS Commands and parameter coverage
- Other Configuration
- CICS Runtime Preprocessor

- CICS Runtime Messages

CICS Runtime Concepts

Purpose

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

CICS Runtime Goals

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

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

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

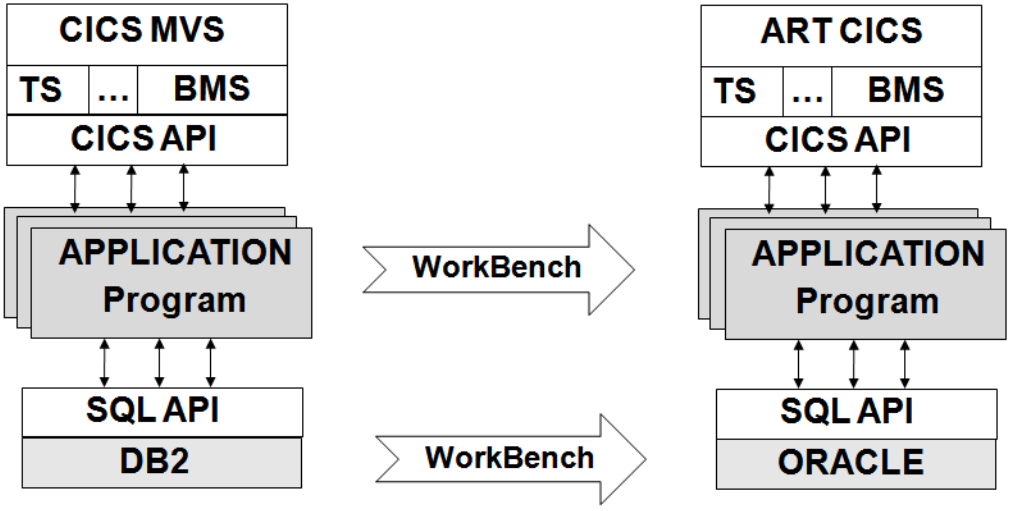
In fact, CICS Runtime gives customers the benefits of Tuxedo distributed architecture without impacting application APIs. It allows the key strengths of Tuxedo to be leveraged and allows routes to the future including SOA to be opened.

CICS Runtime Architecture

Software Development Perspective

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

Figure 2-1 Migration software environments



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

Programmatic Interface

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

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

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

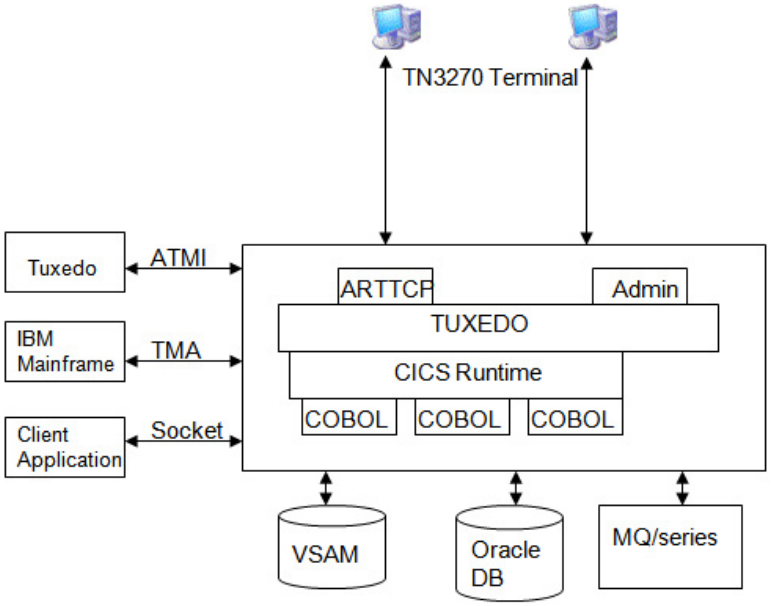
On the target platform, the same philosophy is used; an CICS Runtime Preprocessor (an CICS Runtime compile-time tool) transforms these EXEC CICS into calls to the CICS Runtime library.

For the software developer, there is little that changes. The CICS Runtime preprocessor automatically changes the CICS API that is called. There are some limitations in the command that can be used; these are described in CICS Commands and parameter 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 terms 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 Oracle Tuxedo, with all Oracle 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 an 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.

The following table gives an overview of how the source platform notions have an equivalent on the target platform.

Table 2-1

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

Administrative Tasks

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

This provides all the robust characteristics of Oracle 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 file

– Launch a few servers offering transactions on this machine

CICS Runtime Servers

The CICS Runtime Servers

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

3270 Terminals and User Session Management (ARTTCPL/ARTTCPH)

Description

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

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

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

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

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

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

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

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

Connection Server (ARTCNX)

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

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

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

The connection server also provides a few classical CICS transactions:

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

See Authentication Transactions for more information.

Synchronous Transactions and Program Management

Description

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

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

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

Processing

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

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

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

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

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

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

To summarize the differences:

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

Temporary Storage 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.

DPL servers (ARTDPL)

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

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

Asynchronous transaction servers (ARTATRN/ARTATR1)

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

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

Conversation server (ARTCTRN/ARTCTR1)

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

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

Delayed Asynchronous transaction (/Q Part)

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

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

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

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

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

Administration server (ARTADM)

The administration server is responsible for the administration of CICS Runtime.

In a distributed target environment, this server can be configured on each node in order to propagate the configuration. The configuration files only need to be configured on the master node; the administration servers propagate the configuration files to each slave node.

CHAPTER 4

CICS Runtime Configuration Files

Overview

The administration of CICS Runtime is based on Oracle 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 Oracle Tuxedo and Resource Files

A CICS resource like a transaction with all its characteristics (first program, restartable, …) is described in resource configuration files. The Oracle Tuxedo configuration elements, like how many servers of which group on which machine will offer this transaction is described in the Oracle 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 Oracle 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 describe a resource type: transaction, transaction class, program, files, TS Queue, …, with all the specific parameters relevant to this resource.

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

Structure

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

- Transaction configuration file
- Tranclasses Configuration File
- Programs Configuration File
- Files Configuration File
- TS Queue Model Configuration File
- ENQ-Model Configuration file
- Mapset Configuration File
- Typeterm Configuration File

Transaction configuration file

This table lists the transactions available to application users, with their characteristics.

The filename is `transactions.desc`.

Table 4-1 Transaction Parameters

Field Name	Type	Values	Description
TRANSACTION	X(4)	Mandatory	Name of the transaction
GROUP	X(10)	Mandatory	The group notion of CICS allowing a group of related resources to be declared and instantiated or not by a CICS system when starting.
DESCRIPTION	X(60)	Optional	A small textual comment zone for description of the resource.
PROGRAM	X(30)	Mandatory	Name of the first program to be called for this transaction.
ALIAS	X(4)	Optional	Used to define an alias for the transaction (usually lower case).
CMDSEC	Bool	NO | YES	The ESM to be called for system programming requests.
CONFDATA	Bool	NO | YES	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.
PRIORITY	9(3)	1 | n	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.

Table 4-1 Transaction Parameters

Field Name	Type	Values	Description
RESSEC	Bool	NO | YES	Specifies whether resource security checking is to be used for resources accessed by this transaction.
RESTART	Bool	NO | YES	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.
STATUS	X(10)	ENABLED| DISABLED	Specifies the transaction status. • ENABLED: Allows the transaction to be executed normally. • DISABLED: Prevents the transaction being executed.
TASKDATAKEY	X(5)	USER |CICS	
TPNAME	X(64)	Optional	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	YES|NO	Specifies whether the activity of this transaction is to be traced.

Table 4-1 Transaction Parameters

Field Name	Type	Values	Description
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. Default value: 0.

Tranclasses Configuration File

This table lists and define tranclasses available to regulate parallel transactions activities.

The filename is `tranclasses.desc`.

Table 4-2 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.

Table 4-2 Transclass Parameters

Field Name	Type	Values	Description
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

This table lists and define 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`.

Files Configuration File

This table lists and define files available to be referenced by the CICS application.

The filename is `files.desc`.

Table 4-3 Files Parameters

Field Name	Type	Values	Description
FILE	X(8)	Mandatory	Name of the file; logical name of the file used in EXEC CICS related to this file.
GROUP	X(10)	Mandatory	The group notion of CICS allowing a group of related resources to be declared and instantiated or not by a CICS system when starting.
DESCRIPTION	X(60)	Optional	A small textual comment zone for description of the resource.
DISPOSITION	X(5)	SHARE | OLD	Specifies the disposition of this file. (shared or exclusive).
DSNAME	X(60)		Specifies the data set name (as known to the operating system) to be used for this file.

Table 4-3 Files Parameters

Field Name	Type	Values	Description
JOURNAL	X(20)	NO | *journal*	Specifies whether you want automatic journaling for this file. The journalized data is in the format of the VSAM record and is used for user controlled journaling. The data to be journalized is identified by the JNLADD, JNLREAD, JNLSYNCREAD, JNLSYNCWRITE, and JNLUPDATE attributes. This Journal is for auditing.
KEYLENGTH	Short		Specifies the length in bytes of the logical key of records in remote files, and in coupling facility data tables that are specified with LOAD(NO). In the current release, we support neither remote (extra Tuxedo system) files, nor CFDT; In future we may support Remote file shipping and use this Key Length.
OPENTIME	X(8)	FIRSTREF |STARTUP	Specifies when the file is opened.
READINTEG	X(12)	UNCOMMITTED | CONSISTENT| REPEATABLE	Specifies the level of read integrity required for files defined with RLSACCESS(YES). You can use READINTEG to set a default level of read integrity for a file. This default is used by programs that do not specify one of the API read integrity options. On the target platform the exact semantic of the three levels of integrity may vary from exact CICS/VSAM semantic to another.

Table 4-3 Files Parameters

Field Name	Type	Values	Description
RECORDSIZE	Short	1 - 32767	Specifies the maximum length in bytes of records in a remote file or a coupling facility data table. The size specified can be in the range 1 through 32767. In the current release, we support neither remote (extra Tuxedo system) files, nor CFDT; In future we may support Remote file shipping and use this Record Size.
REMOTENAME	X(8)		Specifies the name of the file on the remote system.
REMOTESYSTEM	X(4)		On source, specifies the name of the connection that links the local system to the remote system where the file resides. On the target platform, will be used only in case of file shipping to another system, either another TUXEDO system or native CICS system.
STATUS	X(10)	ENABLED| DISABLED| UNENABLED	Specifies the initial status of the file following a CICS initialization. UNENABLED allows for explicit EXEC CICS SET FILE OPEN.

Journaling Attributes

Table 4-4 Journaling Attributes

Field Name	Type	Values	Description
JNLADD	X(6)	NONE| BEFORE| AFTER| ALL	Specifies if you want the add operations recorded on the journal nominated by the JOURNAL attribute. On the target platform the semantic is conserved with a simplification: for BEFORE/AFTER/ALL, a single record is logged.
JNLREAD	X(10)	NONE| UPDATEONLY| READONLY| ALL	Specifies the read operations you want recorded on the journal nominated by the JOURNAL attribute. Possible values are: • ALL: Journal all read operations. • NONE: Do not journal read operations. • READONLY: Journal only READ ONLY operations (not READ UPDATE operations). • UPDATEONLY: Journal only READ UPDATE operations (not READ ONLY operations).
JNLSYNCREAD	Bool	NO | YES	Specifies whether you want the automatic journaling records, written for READ operations to the journal, to be synchronous.
JNLSYNCWRITE	Bool	YES | NO	Specifies whether you want the automatic journaling records, written for WRITE operations to the journal, to be synchronous.
JNLUPDATE	Bool	NO | YES	Specifies whether you want REWRITE and DELETE operations recorded on the journal.

TS Queue Model Configuration File

This table lists and defines TS 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	NO | YES	Specifies whether or not queues matching this model are to be recoverable. On the target platform, a default queue is written to file, while a recoverable queue is stored in the RDBMS to provide recovery capabilities.
POOLNAME	X(8)	Optional	Specifies the 8-character name of the shared TS pool definition that you want to use with this TSMODEL definition. Will probably disappear in the future releases since there are other ways on target to arrive to the same result.

Table 4-5 TS Queue Parameters

Field Name	Type	Values	Description
REMOTE_SYSTEM	X(4)	optional	On source platform, specifies the name of the connection that links the local system to the remote system where the temporary storage queue resides. On the target platform, used only in case of TS shipping to another system, either another TUXEDO system or native CICS system.
REMOTEPREFIX XREMOTEPREFIX	X(16)	Optional	Specifies the character string that is to be used as the prefix on the remote system. The prefix may be up to 16 characters in length. These options are useful (on source and target platforms) only if one wants to translate queue name when shipping TS Queue access from one system to another.
SECURITY	Bool	NO | YES	Specifies whether security checking is to be performed for queues matching this model.

ENQ-Model Configuration file

This table 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.

Table 4-6 ENQ Model Parameters

Field Name	Type	Values	Description
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	E = Enabled D = Disabled.

Mapset Configuration File

This table lists and defines mapsets available to be referenced by the CICS application. See MAPSET Generator for a further description of how they are used.

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 4-7 Mapset Parameters

Field Name	Type	Values	Description
NAME	X(8)	Mandatory	Name of the mapset.
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.
FILENAME	X(79)	NO | YES	This specifies the physical (binary) file name of the mapset, which is generated by the tcxmapgen tool (refer to section). It will be searched in directories defined by the KIX_MAP_PATH environment variable i f 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.

Table 4-7 Mapset Parameters

Field Name	Type	Values	Description
swastatus	X(10)	ENABLED| DISABLED	Specifies the resource status.•If set to ENABLED, the resource is available.•If set to DISABLED, the resource is unavailable for use by the system.
Usage	X(10)	NORMAL | TRANSIEN T	This attribute specifies the caching scheme to be used once the MAPSET is loaded. NORMAL keeps the MAPSET loaded in a cache. Unload it when the cache overflows and it is the oldest, least used MAPSET in the cache.TRANSIENT unloads the MAPSET if it is not being used.

Typeterm Configuration File

This table lists and define Typeterms supported by ARTTCP.

The filename is `typeterms.desc`

The format of a TYPETERM definition is:

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

For example,

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

.

Table 4-8 Typeterm parameters

Field Name	Type	Values	Description
NAME	X(79)	Mandatory	Name of the typeterm.
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(79)	Optional	A small textual zone for description of the resource.
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.
logonmsg	Bool	NO | YES	Indicates whether the “Good Morning” (CSGM) transaction is automatically started on the terminal. Tuxedo ART provides a default CSGM transaction. Please refer to section for the configuration of the default “Good Morning” (CSGM) transaction.
outline	Bool	NO | YES	Indicates whether the terminal supports field outlining.
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

Table 4-8 Typeterm parameters

Field Name	Type	Values	Description
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.

CHAPTER 5

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_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 meaning no trace, 9 meaning maximum trace. The default value is 0 when the variable is not defined.

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.

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 compulsory server arguments SVRGRP & SVRID, plus only the arguments needed specifically by the server type.

In the CLOPT, the following syntax is used:

```
CLOPT="[servopts] -- generic-CICS Runtime-options server-specific-options"
```

Where:

Servopts
Represent options common to all Tuxedo servers described in servopts.

generic-CICS Runtime-options
Represent options specific to CICS Runtime, but available to all CICS Runtime servers.

server-specific-options
Are replaced by the options specific to the specific server.

Generic CLOPT options of CICS Runtime servers

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

In the description of each specific server, they are referred to as generic-CICS Runtime-options.

CICS System id 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[/:-].

Exclusion

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

List of Groups argument

This argument defines the list of resources group to be considered by this server(s) when loading resources.

Synopsis

```
-l group1:group2:…:groupn
```

Description

Groups in the resources configuration files are defined by 10 character strings. A server will only load 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 that it is reading.

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 -L pnetaddr [-m minh] [-M maxh] [-x session-per-handler] [-p
profile-name] [-D] [+H trace-level]"
```

Description

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

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

Parameters

The following CLOPT run-time parameters are recognized:

-n netaddr

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

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

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

-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.

[-D]

Enable Debug.

[+H trace-level]

Specify the trace level:

-1: trace off.

0: trace for all ARTTCPH.

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

Examples

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

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 TEST-1 grp1:group2"
```

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 compulsory for this server type, and must be defined as CONV=Y, because ARTSTRN is conversational.

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 option, see CICS System id argument.

-l GroupList

Mandatory option, see 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 -l group1:group2"
```

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 TEST -l group1:group2,…"
```

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 -l group1:group2"
```

ARTTSQ Configuration

Server Name

ARTTSQ – CICS Runtime Temporary Storage Queue Server

Synopsis

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

        CLOPT"[servopts] -- -l grp1:group2"
```

Description

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

ARTTSQ publishes two main kinds of services:

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

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

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

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

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

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

This unicity is verified when services are published.

Parameters

ARTTSQ

The following CLOPT run-time parameters are recognized:

[-l Group]

Mandatory option, see List of Groups argument for more information.

DBMS Constraints

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

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

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

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

Listing 6-1 TS_QCONTENT Creation

```
drop table TS_Q_CONTENT purge;

create table TS_Q_CONTENT

( TS_QUEUE   char(16) NOT NULL,

  TS_ITEM    number(8) NOT NULL,

  TS_LENGTH  number(8),

  TS_RAW     LONG RAW,

  primary key (TS_QUEUE, TS_ITEM)
```

```
);
```

Environment variables used

- KIXCONFIG
- KIX_TS_DIR
- KIX_TRACE_LEVEL
- KIX_MAP_PATH
- KIX_TECH_DIR

Examples

```
*SERVERS

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

        MIN=1 MAX=1 CLOPT=" -- -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 TEST -l grp1:group2"
```

Description

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

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

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

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

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

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

Conversely, these programs will not be available directly from `str` or `atr` servers.

Parameters

minn and maxn

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

CLOPT options

The following CLOPT run-time parameters are recognized:

-s SystemID]

Mandatory option, see CICS System id argument.

-l GroupList]

Mandatory option, see List of Groups argument.

Environment variables used

- KIXCONFIG
- KIX_CWA_SIZE
- KIX_CWA_IPCKEY
- KIX_TRACE_LEVEL
- KIX_TECH_DIR

Examples

```
*SERVERS

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

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

  CLOPT=" -- -s PROW -l group1:group2"
```

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 TEST] [-l
group1:group2:…]"
```

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 a technical 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 transactional.

minn and maxn

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

CLOPT

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

[-s SystemID]

Mandatory option, seeCICS System id argument for more information.

[-l GroupList]

Mandatory option, see List of Groups argument for more information.

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 -l group1:group2"
```

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 TEST] [-l group1:group2,…]"
```

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 -l group1:group2"
```

ARTCTRN Configuration

Server Name

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

Synopsis

```
ARTCTRN SRVGRP="identifier" SRVID="number" CONV=N MIN=minn MAX=maxn
RQADDR=QKIXCTR REPLYQ=Y CLOPT="[servopts] -- [-s TEST] [-l
group1:group2:…]"
```

Description

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

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

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

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

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

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

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

Parameters

CONV

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

minn and maxn

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

CLOPT

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

[-s SystemID]
Mandatory option, seeCICS System id argument for more information.

[-l GroupList]
Mandatory option, see List of Groups argument for more information.

Environment variables used

KIXCONFIG

KIX_CWA_SIZE

KIX_CWA_IPCKEY

KIX_QSPACE_IPCKEY

KIX_TRACE_LEVEL

KIX_TECH_DIR

Examples

```
*SERVERS

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

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

  CLOPT=" -- -s PROW -l group1:group2"
```

ARTCTR1 Configuration

Server Name

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

Synopsis

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

       CLOPT="[servopts] -- [-s TEST] [-l group1:group2,…]"
```

Description

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

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

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

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

Examples

```
*SERVERS

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

  CONV=N MIN=1 MAX=1

  CLOPT=" -- -s PROW -l group1:group2"
```

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 technical 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 a technical 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

CLOPT

No specific CLOPT run-time parameters are recognized.

Environment variables used

KIX_TRACE_LEVEL

Examples

```
*SERVERS

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

ARTADM Configuration

Server Name

ARTADM — Administration Server

Synopsis

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

Description

This server is responsible for the administration of CICS Runtime.

In a distributed target environment, this server can be configured on each node to achieve the configuration propagation. With these servers, the configuration files only need to be configured on the master node, and the administration servers propagate the configuration files to each slave node.

1. When starting up, the administration server running on the master node reads in all the configuration files located in directory ${KIXCONFIG}.
2. When each administration server running on a slave node starts up, it communicates with the administration server on the master node and fetches the contents of the configuration files.
3. The administration server on the slave node then writes to the corresponding configuration files in directory ${KIXCONFIG} on the slave node. New configuration files are created if none exist.

This server is optional. When configuration propagation is required, an administration server should be configured on every Tuxedo node.

- The ARTADM server on the master node must be started as the first server in the target application.
- The ARTADM server on the slave node must be started as the first server on the node.

The ARTADM server on the master node offers a service for the internal communications with the ARTADM servers running on the slave nodes.

- ‘getConfig' is used by the slave ARTADM server to get configuration information from the master ARTADM server.

Examples

```
*SERVERS

ARTADM SRVGRP="ADMGRP" SRVID=1000 RESTART=Y
```

/Q Configuration for CICS Runtime

/Q Configuration for CICS Runtime

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

In this case, the transaction request is deposited into a 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 will be created in the queue space.
- A TMQFORWARD server will be configured to receive messages from this queue and invoke the application transaction corresponding to the request.

/Q Configuration for delayed transactions

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

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

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

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

Listing 7-1 crdl and Queue Space "ASYNC_QSPACE"

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

```
16
Q
!end
```

Creating a Queue

Listing 7-2 ASYNC_QUEUE" using Tuxedo "qcreate" tool

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

qmadmin ${QMCONFIG} <<!end
qopen ASYNC_QSPACE
qcreate
ASYNC_QUEUE
fifo
none
2
```

```
30
80%
0%

qcreate
RPLYQ
fifo
none
2
30
80%
0%

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

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

Tuxedo /Q server configuration in the ubbconfig file

In the *GROUPS section

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

In the *SERVERS section

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

Security Configuration

Security configuration

Authentication configuration

CICS provides two system transactions for authentication purposes:

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

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

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

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

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

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

Tuxedo security mechanisms

ARTTCP supports two types of Tuxedo security mechanisms: application password (APP_PW) and user-level authentication (USER_AUTH).

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

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

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

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

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

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

Integration with the External Security Manager

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

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

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

Listing 8-1 COBOL CICS resource authorization interface

```
01 ret-code                    usage int.

LINKAGE SECTION.

01 AUTH-USERID              PIC X(30).

01 AUTH-GROUPID             PIC X(256).

01 AUTH-RSRCE-TYPE          PIC X(256).

01 AUTH-RSRCE-NAME          PIC X(512).

01 AUTH-ACCESS-TYPE         PIC X(6).

PROCEDURE DIVISION USING LK-AUTH-USERID LK-AUTH-GROUPID

                         LK-AUTH-RSRCE-TYPE LK-AUTH-RSRCE-NAME

                         LK-AUTH-ACCESS-TYPE.
```

Accepting

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

Returning

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

Codification

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

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

Security Profile Generator

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

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

genappprofile (1)

Name

`genappprofile` — Security Profile Generator

Synopsis

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

Description

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

Options

The command option is:

[-f <output_file>]

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

CICS Commands and parameter coverage

CICS commands supported

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

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

CICS Command and Parameter Support Table

Table 9-1 CICS command

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

Table 9-1 CICS command

CICS domain	CICS command	Command parameter	Status
APPC Mapped conversation	ALLOCATE (APPC)	SYSID (systemname)	
	CONNECT PROCESS	CONVID (name)	
		PROCNAME (data-area)	
		PROCLENGTH (data-value)	
		SYNCLEVEL (data-value)	Partial Support only SYNCLEVEL 0
	CONVERSE (APPC)	CONVID (name)	
		FROM (data-area)	
		FROMLENGTH (data-value)	
		FROMFLENGTH (data-value)	
		INTO (data-area)	
		TOFLENGTH (data-area)	
		TOLENGTH (data-area)	
BMS	RECEIVE MAP	MAP (name)	
		MAPSET (name)	
		INTO (data-area)	
		SET (ptr-ref)	
		TERMINAL	
		FROM (data-area)	
		LENGTH (data-value)	
BMS	SEND MAP	MAP (name)	

Table 9-1 CICS command

CICS domain	CICS command	Command parameter	Status
		MAPSET (name)	
		FROM (data-area)	
		LENGTH (data-value)	
		DATAONLY	
		MAPONLY	
		CURSOR (data-value)	
		ERASE	
		ERASEUP	
		DEFAULT	
		FREEKB	
		ALARM	
		FRSET	
		ACCUM	
		TERMINAL	
		NOFLUSH	
	PURGE MESSAGE		

Table 9-1 CICS command

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

Table 9-1 CICS command

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

Table 9-1 CICS command

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

Table 9-1 CICS command

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

Table 9-1 CICS command

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

Table 9-1 CICS command

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

Table 9-1 CICS command

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

Table 9-1 CICS command

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

Table 9-1 CICS command

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

Table 9-1 CICS command

CICS domain	CICS command	Command parameter	Status
Interval control	DELAY	INTERVAL (hhmmss)	
		SECONDS (data-value)	
		REQID (name)	
	RETRIEVE	INTO (data-area)	
		SET (ptr-ref)	
		LENGTH (data-area)	
	START	TRANSID (name)	
		INTERVAL (hhmmss)	
		FROM (data-area)	
		LENGTH (data-value)	
		TERMID (name)	Recognized
		USERID (data-value)	Partial The security stub is called with USERID and TRANSID
Program control	LINK	PROGRAM (name)	
		COMMAREA (data-area)	
		SYSID (systemname)	
		LENGTH (data-value)	
	RETURN	TRANSID (name)	
		IMMEDIATE	
		COMMAREA (data-area)	
		LENGTH (data-value)	

Table 9-1 CICS command

CICS domain	CICS command	Command parameter	Status
Program control	XCTL	PROGRAM (name)	
		COMMAREA (data-area)	
		LENGTH (data-value)	
Syncpoint	SYNCPOINT		
	SYNCPOINT ROLLBACK		
System's programmer's function	INQUIRE TRANSACTION	INQUIRE TRANSACTION (datavalue)	
		STATUS (cvda)	

Table 9-1 CICS command

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

Table 9-1 CICS command

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

Table 9-1 CICS command

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

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

Other Configuration

MAPSET Generator

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.

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

tcxmapgen(1)

Name

`tcxmapgen` — CICS Runtime MAPSET Generator.

Synopsis

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

Description

This utility compiles BMS source files (specified in the <file> parameter), which contain CICS mapset definitions, into binary form. The binary files are required for the CICS Runtime execution environment.

The compiler also produces a COBOL copybook.

Options

The command options are:

[-c]

No binary map file.

This specifies that only COBOL copybook (file.cpy) output file will be generated.

[-l]

Produce a listing.

This specifies that a listing output file (named file.lst) is to be produced.

[-m]

No symbolic include file.

This specifies that only binary mapset file (named file.mpdef) is produced.

[-o file]

Output file name prefix.

This specifies the name to be used for the generated output files. The compiler uses the file name with an appended extension when creating the output file names.

Example

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

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

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

Tuxedo ART System Transactions References

Authentication Transactions

CESN

The CESN transaction uses MAPSET CSIGNON. So 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"
```

CESF

No special configuration is required for CESF transaction.

"Good Morning" Transaction

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

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

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

CHAPTER 11

CICS Runtime Preprocessor

Overview

Definition

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

Pre-requisites

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

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

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

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

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

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

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

prepro-cics.pl

Name

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

Synopsis

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

Description

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

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

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

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

Options

notrec

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

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

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

type_output

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

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

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

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

Any other value will be considered as "normal".

Restrictions

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

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

Error Messages

Invalid CICS messages

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

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

Summaries may be:

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

IGNORE and HANDLE instructions messages are quite straightforward:

IGNORE should be constructed with CONDITION.

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

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

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

Non supported error messages

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

```
Instruction non supported
```

is generated.

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

CICS Runtime Messages

Messages

CICS Runtime Messages provide the following information:

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

Preprocessor messages

Error Messages

Invalid CICS messages

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

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

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

IGNORE and HANDLE instructions messages are quite straightforward:

"IGNORE should be constructed with CONDITION"

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

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

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

Other error messages

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

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

Maintenance messages

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

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

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