

Oracle Tuxedo Application Runtime for CICS

User Guide

11g Release 1 (11.1.1.3.0)

December 2011

ORACLE®

Copyright © 2011, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Introduction to CICS Runtime

Introduction to the CICS Runtime Environment	1-1
Purpose	1-1
How This Book Is Organized.	1-1

Overview of the CICS Runtime

General Architecture	2-1
The CICS Runtime Library	2-2
The CICS Runtime Tuxedo Servers	2-4
Mandatory Servers	2-4
Optional Servers	2-4
Server Configuration	2-5

Initial Configuration of the CICS Runtime

CICS Runtime Configuration	3-1
The UNIX ~/.profile File	3-1
The Tuxedo System Files	3-3
The CICS Runtime Resource Configuration Files	3-9
Verifying the Initial Setting Configuration	3-12
Using the Tuxedo tadmin psr Commands	3-12
Using the Tuxedo tadmin psc Commands	3-14
Using the CSGM CICS Good Morning Transaction	3-15

Implementing CICS Applications

Presentation of the z/OS Simple Application	4-2
Introduction	4-2
Description of the CICS Simple Application Components	4-2
Configuring a Standard CICS Application With CICS Runtime	4-3
CICS Runtime Configuration	4-4
Verifying the CICS Application Installation	4-12
Using the Tuxedo tadmin psr Commands	4-12
Using the Tuxedo tadmin psc Commands	4-13
Using the CICS Runtime Application	4-14
Presentation of Simple Application on COBOL-IT / BDB	4-16
Configuring ubbconfig File in CICS Runtime	4-16
Building BDB TMS Server	4-18
Exporting Variables Before Booting Up ART Servers	4-18
Implementing Synchronous CICS Transactions With a Limited Number of Parallel Instances	4-18
The Special Case of Transaction Classes With MAXACTIVE=1	4-19
Modification of the ubbconfig File for Sequential Transactions	4-19
Checking the ARTSTR1 Configuration	4-23
Implementing Asynchronous CICS Non-Delayed Transactions	4-25
Modifying the Tuxedo ubbconfig File to Manage Asynchronous Transactions	4-25
Using Parallel Asynchronous Transactions	4-25
Using Non-Parallel Asynchronous Transactions	4-27
Implementing Asynchronous CICS Delayed Transactions	4-29
Creating the Tuxedo /Q Features	4-29
Modifying the Tuxedo ubbconfig File to Manage the Tuxedo /Q Queue	4-30
Implementing CICS Application Using Temporary Storage (TS) Queues	4-34

Implementing Unrecoverable TS Queues	4-37
Implementing Recoverable TS Queues	4-37
Managing TD Queue Intrapartitions.	4-41
Presentation of the Mechanism on Source Platform	4-41
Automatic Transaction Initiation (ATI)	4-42
Presentation of the Mechanism on Target Platform	4-43
Runtime CICS Configuration of TD Queue Intrapartition	4-44
Implementing Distributed Program Link (DPL)	4-48
To Detect That DPL Is Needed	4-48
Modifying the Tuxedo ubbconfig File to Manage the DPL	4-51
Declaring Remote Programs in CICS Runtime.	4-54
Implementing CICS Common Work Area (CWA)	4-60
Implementing a CICS Transaction Work Area (TWA)	4-61
Implementing CICS Transaction Trigger Monitor (ARTCKTI)	4-64
Work Flow	4-64
Command Configuration	4-66
CICS Runtime Logs	4-66
Tuxedo System Log	4-66
The CICS Runtime Server Logs	4-70
Disabling and Enabling Programs	4-74
To Disable Programs	4-74
To Enable Programs	4-75
Checking the Change in Program Status	4-75
Removing and Adding Applications for CICS Runtime.	4-77

Reference

Cross Reference of .desc Configuration Files Used by CICS Runtime Servers	5-1
---	-----

Oracle Tuxedo Application Runtime for CICS CSD Converter

Overview	6-1
Resource Definition Online (RDO) Mapping	6-1

UDB Linking

Installation Time UDB Linking	A-1
Rebuilding Servers for UDB	A-2

Rebuilding ART Servers for CICS

Rebuilding the ART CICS Servers	B-1
---------------------------------------	-----

External CICS Interface (EXCI)

Overview	C-1
EXCI in Oracle Tuxedo Application Runtime	C-2
Supported EXCI Interface	C-2
Precompiler Controls	C-3
Access Authorization	C-3
ART-KIX Implementation	C-4

Introduction to CICS Runtime

Introduction to the CICS Runtime Environment

Purpose

This guide provides explanations and instructions for configuring and using Oracle Tuxedo Application Runtime for CICS (CICS Runtime) when developing and running On Line Transaction Processing (OLTP) applications on a UNIX/Linux platform.

This guide describes the steps required to implement and perform COBOL CICS transactions, whether they are migrated from z/OS CICS or newly written for UNIX applications.

To illustrate this purpose, the User Guide provides a detailed description of the deployment and administration of the Simple Application in a UNIX environment.

This guide helps you to:

- Configure CICS Runtime software.
- Declare components to CICS Runtime.
- Run a CICS Application.

How This Book Is Organized

This guide is divided into three main chapters:

- [“Overview of the CICS Runtime” on page 2-1](#): introduces the general principles of the CICS Runtime.
- [“Initial Configuration of the CICS Runtime” on page 3-1](#): describes how to set parameters to make CICS Runtime operational before implementing CICS applications.
- [“Implementing CICS Applications” on page 4-1](#): details how to configure the CICS Runtime to use CICS applications including examples moving from simple to more-and-more complex cases.

Additionally,

- [“Reference” on page 5-1](#): contains information describing the .desc files used by the different CICS Runtime servers.

Overview of the CICS Runtime

General Architecture

In a z/OS environment, CICS is used to establish transactional communications between end-users and compiled programs via screens.

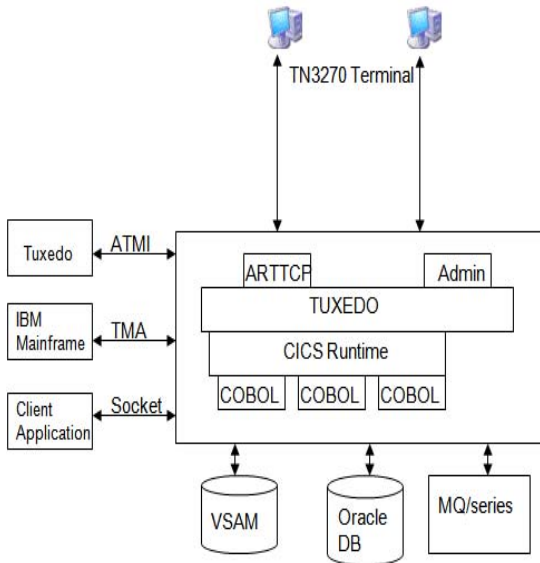
CICS is a middleware that implements the control and integrity of shared resources, providing developers with a bag of API (EXEC CICS ... END-EXEC statements) to dialog with CICS inside programs mainly developed on z/OS in COBOL, PL1 and Assembler languages.

Once all the components of z/OS CICS applications (COBOL programs and data) are migrated to a UNIX/linux platform using Oracle Tuxedo Application Runtime Workbench, CICS Runtime enables them to be run unchanged using an API emulation on top of the native Tuxedo features.

On a UNIX platform, Tuxedo performs many of the functions performed by CICS on a z/OS platform concerning the integrity of resources and data used in transactional exchanges, including those used for applications that are distributed across several machines. However, Tuxedo does not manage some specific native CICS z/OS features such as screen map handling. To provide these features on the target platform, CICS Runtime acts as a technical layer, located between Tuxedo and the converted CICS applications.

The following schema describes the global architecture of CICS Runtime.

Figure 2-1 Oracle Tuxedo Application Runtime for CICS Architecture



CICS Runtime is composed of two major parts:

- CICS Runtime Preprocessor and CICS Runtime library
- CICS Runtime Tuxedo Servers and their Resource Configuration Files

The CICS Runtime Library

In z/OS CICS applications, all the interactions with the resources managed by CICS are made thru the EXEC CICS API.

A CICS Preprocessor transforms these statements into calls to CICS library:

Listing 2-1 z/OS CICS Calls

```
*EXEC CICS  
*    RECEIVE MAP      ('RTSAM10')
```

```

*           MAPSET ('RTSAM10')
*           INTO   (RTSAM10I)

*END-EXEC.

MOVE 'xxxxxxxxxxxxx00203' TO DFHEIV0
MOVE 'RTSAM10' TO DFHC0070
MOVE 'RTSAM10' TO DFHC0071
CALL 'DFHEI1' USING DFHEIV0 DFHC0070
                    RTSAM10I DFHDUMMY DFHC0071.

```

On UNIX, the CICS Runtime Preprocessor transforms these EXEC CICS into calls to the CICS Runtime library:

Listing 2-2 CICS Runtime Calls

```

*EXEC CICS
*   RECEIVE MAP   ('RTSAM10')
*           MAPSET ('RTSAM10')
*           INTO   (RTSAM10I)
*END-EXEC.

INITIALIZE KIX--INDICS
MOVE LOW-VALUE TO KIX--ALL-ARGS
. . .
ADD 1 TO KIX--ARGS-NB
SET KIX--INDIC-MAPSET(KIX--ARGS-NB) TO TRUE
MOVE 'RTSAM10' TO KIX--MAPSET OF KIX--BMS-ARGS
ADD 1 TO KIX--ARGS-NB
SET KIX--INDIC-MAP(KIX--ARGS-NB) TO TRUE
MOVE 'RTSAM10' TO KIX--MAP OF KIX--BMS-ARGS

```

```
CALL "KIX__RECEIVE_MAP" USING KIX--INDICS KIX--ALL-ARGS
```

The CICS Runtime Tuxedo Servers

The CICS Runtime Tuxedo servers are used to manage CICS features not natively present in Tuxedo.

Some of these servers are mandatory in order to make CICS Runtime available, others are optional depending on the usage of specific EXEC CICS statements in CICS Applications.

Mandatory Servers

- The Terminal Connection servers (TCP servers: ARTTCPH and ARTTCPL servers): manage user connections and sessions to CICS applications thru 3270 terminals or emulators.
- The Connection server ARTCNX: manages the user session and some technical transactions relative to security (CSGM: Good Morning Screen, CESN: Sign On, CESF: Sign off).
- The Synchronous Transaction server ARTSTRN: manages standard synchronous CICS transactions that can run simultaneously.
- The Administration server (ARTADM): required for CICS Runtime administration

Optional Servers

- The Synchronous Transaction servers ARTSTR1: manages CICS synchronous transaction applications that can not run simultaneously but only sequentially (one at a time).
- The Asynchronous Transaction servers ARTATRN and ARTATR1: are similar to the ARTSTRN and ARTSTR1 but for asynchronous transactions started by EXEC CICS START TRANSID statements.
- The TS Queue servers ARTTSQ, TMQUEUE and TMQFORWARD: manage the use of CICS Temporary Storage Queues - files managed by CICS thru specific commands.
- The TD Queue servers ARTTDQ: centralizes the TD Queue operations management requested by applications.
- The Distributed Program Link server ARTDPL: runs DPL programs.
- Converse Management server ARTCTRN and ARTCTR1.

Server Configuration

The CICS Runtime Tuxedo servers are configured in:

1. The ubbconfig file once compiled to the tuxconfig file, is the file read by Tuxedo at start up that defines all the servers to be launched and their parameters.
2. The CICS Runtime resource configuration files for the CICS resources managed by CICS Runtime servers are declared.

The CICS Runtime Resource Configuration Files

Reminder about z/OS Resource Management

On z/OS, all the technical components used by CICS applications (terminals, transactions, programs, maps, files ...) are named CICS resources and must be declared to CICS using a dedicated configuration file called CSD.

Each resource declared must belong to a resource Group name. This enables a set of resources bound together constituting a technical or a functional application to be managed (install, delete, copy to another CSD...).

Once created, one or more CICS groups can be declared in a CICS List name. All or part of these List names are given to CICS at startup to install their CICS groups, and thus make available all the resources defined in these groups.

CICS Runtime Resource Management

CICS Runtime manages only a subset of the resource types previously defined in the CICS CSD file on z/OS. Each resource type definition of this subset is stored inside its own dedicated Resource Configuration file. All these files are located in the same UNIX directory.

The Group name notion is kept to preserve the same advantages as on the z/OS platform. For this purpose, each resource defined in the configuration files must belong to a CICS Group name.

CICS Runtime manages the following resources:

- Tranclasses (`transclasses.desc` file)

This file contains all the distinct Transaction classes (Tranclasses) referenced by the CICS Transactions. In CICS Runtime, a Tranclass is a feature defining whether several instances of the same transaction can be run simultaneously or sequentially.

- Transactions (`transactions.desc` file)

A transaction is a CICS feature allowing a program to be run indirectly thru a transaction code either manually from a 3270 screen or from another COBOL CICS program.

A transaction belongs to a transaction class in order to define whether this transaction must be run exclusively.

- Programs (`programs.desc` file)

This file contains a list of all COBOL or C programs invoked thru `EXEC CICS START`, `LINK` or `XCTL` statements.

- TS Queue Model (`tsqmodel.desc` File)

Contains all the TS Queue models referenced by TS Queues used in the CICS programs.

A TS Queue model defines properties that complete or replace those defined in the CICS API that manages Temporary Storage Queues. The names of these TS Queues must match a mask defined in the TS Queue model. In CICS Runtime, these models are mainly used to define whether TS Queues are recoverable or not.

- Mapsets (`mapsets.desc` file)

This file contains all the mapsets referenced by the CICS applications. A mapset is a CICS resource, but also a physical component containing one or more screens (maps) used in the exchanges between CICS applications and end-users.

These resources are used through dedicated CICS statements like `EXEC CICS SEND MAP` or `RECEIVE MAP` inside COBOL programs.

- Typeterms (`typeterms.desc` file)

Contains all of the 3270 terminal types supported by the CICS Runtime TCP servers.

- Enqmodel (`enqmodel.desc`)

defines named resources for which the `EXEC CICS ENQ` and `EXEC CICS DEQ` commands have a sysplex-wide scope.

- File (`files.desc`)

defines the physical and operational characteristics of files

- Extra TDQUEUE (`tdqextra.desc`)

Defines the attributes of extra transient data queues

- Intra TDQUEUE (`tdqintra.desc`)

Defines the attributes of intra transient data queues

Note: ART CICS Runtime provides the `tcxcscvrt` utility to automatically convert the CICS CSD file to CICS Runtime resource configuration files. For more information, see [Oracle Tuxedo Application Runtime for CICS CSD Converter](#).

Overview of the CICS Runtime

Initial Configuration of the CICS Runtime

CICS Runtime Configuration

Before installing a CICS application, certain technical variables and paths must be defined in order to create the CICS Runtime environment.

These operations must be completed before configuring individual CICS applications for use with CICS Runtime.

CICS Runtime uses the following files:

- The UNIX System `~/profile` file to centralize values and paths used by the CICS Runtime for its own needs or for Tuxedo.
- The Tuxedo `envfile` which contains parameters, variables and paths used by Tuxedo.
- The Tuxedo `ubbconfig` file to declare all the required CICS Runtime Tuxedo servers.
- The CICS Runtime resource configuration files used by the CICS Runtime Tuxedo servers.

The UNIX `~/profile` File

For UNIX users, most required variables are defined in the `.profile` file that centralizes all of the common variables and paths used by a user for commands and applications.

Set up in this file all of the common variables and paths that will be used later in the different configuration files required by CICS Runtime or by the other technical software or middleware invoked by it (Oracle, Tuxedo, MQ Series ...).

Initial Configuration of the CICS Runtime

This file should then be exported.

Set the following variables in the initial settings of ~/.profile file

Table 3-1 .profile Variables

Variable	Value	Usage	Variable usage
TUXDIR	Set up at Installation time	Compulsory. Directory containing the Installed Tuxedo product. The default location is /usr/tuxedo	TUXEDO
TUXCONFIG	Set up at Installation time	Compulsory. Full path name of the Tuxedo tuxconfig file	TUXEDO
KIXDIR	Set up at Installation time	Compulsory. Absolute path of the directory containing the CICS Runtime product	CICS Runtime
APPDIR	\${KIXDIR}/bin	Compulsory. Directory containing the CICS Runtime Servers Binaries	CICS Runtime
KIXCONFIG	Set up at Installation time	Compulsory. Directory where the Resources Configuration Files of the CICS Runtime are located	CICS Runtime
KIX_TS_DIR	Set up at Installation time	Compulsory. Directory used for the non-recoverable CICS Queue TS.	CICS Runtime

Listing 3-1 .profile file Initial Settings Example

```
export TUXDIR=/product/TUXEDO11GR1# Directory containing the Installed
Tuxedo product

export TUXCONFIG=${HOME}/SIMAPP/config/tux/tuxconfig# Full path name of the
Tuxedo tuxconfig file

export KIXDIR=${HOME}/KIXEDO# Absolute path of the CICS Runtime product
directory

export APPDIR=${KIXDIR}/bin # Directory containing the CICS Runtime
Servers Binaries
```

```
export KIXCONFIG=${HOME}/SIMAPP/config/resources # Directory for resources
files (*.desc)

export KIX_TS_DIR=${HOME}/SIMAPP/KIXTSDIR# Directory for TS no recovery
```

The Tuxedo System Files

The Tuxedo Envfile File

This envfile contains variables and paths used by Tuxedo and CICS Runtime. These parameters should be set in addition to those set by the Tuxedo Administrator.

Set the following variables in the initial settings of the envfile file.

Table 3-2 envfile Variables

Variable	Value	Usage
LC_MESSAGES	C	UNIX formats of informative and diagnostic messages
OBJECT_MODE	64	UNIX 64 bits architecture
APPDIR	\${APPDIR}	TUXEDO environment.
TUXCONFIG	\${TUXCONFIG}	TUXEDO environment
USER_TRACE	SID	TUXEDO environment. Trace Type (one per user)
KIXCONFIG	\${KIXCONFIG}	CICS Runtime directory containing its resource files
PATHTS	\${KIX_TS_DIR}	CICS Runtime directory used for the unrecoverable Temporary Storage

Listing 3-2 envfile Initial Settings Example

```
# <TUXDIR>
#     Refers to the location where you installed TUXEDO. The default
#     location is "/usr/tuxedo".
#
```

Initial Configuration of the CICS Runtime

```
# <APPDIR>
#     Refers to the fully qualified directory name where your application
#     runs (i.e., the location of the libraries, mapdefs, and MIB files).
#
# <TUXCONFIG>
#     Refers to the fully qualified binary version of the TUXEDO
#     configuration file. (This is usually the "tuxconfig" in the $APPDIR
#     directory.)
#
# Copyright ©1998, BEA Systems, Inc., all rights reserved.
#-----
-
# TUXEDO environment
APPDIR=${KIXDIR}/bin
CONFDIR=${APPHOME}/config/tux
TUXCONFIG=${CONFDIR}/tuxconfig
FLDTBLDIR32=${KIXDIR}/src
FIELDTBLS32=msgflds32
OBJECT_MODE=64

#resource files directory
KIXCONFIG=${APPHOME}/config/resources

# Command executable paths
HAB_TRAN=none

# Other environment
PATHTS=${KIX_TS_DIR}
```

```
LC_MESSAGES=C
```

```
# End
```

The Tuxedo ubbconfig File

Some CICS Runtime Tuxedo servers are absolutely needed while others can be optionally started but are not absolutely necessary at this time.

The Mandatory Servers

These servers must be started to run CICS Runtime and verify that the initial settings are correct by being able to display the CICS Runtime Good Morning screen (Host Connection Welcome Screen).

- The Terminal Control Program Listener (`ARTTCPL` server) is needed because it establishes communication between end-users and CICS Runtime applications thru maps displayed on 3270 terminals or emulators.
- The Connection Server (`ARTCNX` server) is also required because it offers technical connections services during the user connection and disconnection phases. It is also used to display the CICS system transactions CICS Runtime Good Morning screen thru the System Transaction CSGM.
- The Administration Server (`ARTADM` server) is also required for CICS Runtime administration.

The Optional Servers

These servers do not need to be launched because they are only used by CICS applications not yet installed.

To not start these servers, comment-out the corresponding line in your ubbconfig file before recompiling.

- The Synchronous Transaction Servers (`ARTSTRN` and `ARTSTR1`) that manage synchronous transaction CICS applications
- The Asynchronous Transaction Servers (`ARTATRN` and `ARTATR1`) that manage asynchronous transaction CICS applications.

Initial Configuration of the CICS Runtime

- The Temporary Storage Server (ARTTSQ server) that manage TS QUEUES used in COBOL CICS programs.
- The Tuxedo /Q TMQUEUE and TMQFORWARD servers that are only used for delayed CICS Transactions

Listing 3-3 ubbconfig Initial Server Configuration Example

```
*SERVERS

ARTTCPL SRVGRP=TCP00

        SRVID=101

        CLOPT="-o /home2/work9/demo/Logs/TUX/sysout/stdout_tcp -e
/home2/work9/demo/Logs/TUX/sysout/stderr_tcp -- -M 4 -m 1 -L //deimos:2994
-n //deimos:2992"

ARTCNX   SRVGRP=GRP01

        SRVID=15

        CONV=Y

        MIN=1 MAX=1 RQADDR=QCNX015 REPLYQ=Y

        CLOPT="-o /home2/work9/demo/Logs/TUX/sysout/stdout_cnx -e
/home2/work9/demo/Logs/TUX/sysout/stderr_cnx -r --"
```

Where:

***SERVERS**

Is the Tuxedo ubbconfig keyword indicating server definitions.

For the ARTTCPL server:

SRVGRP

Is the Tuxedo Group Name to which ARTTCPL belongs.

SRVID

Is the identifier of a ARTTCPL Tuxedo Server.

CLOPT

Is a quoted text string passed to the server containing its parameters.

-o

Indicates the file is used for the standard output messages of the server.

-e

Indicates the file is used for the error output messages of the server.

-M 4

Indicates the maximum number of TCPL handler processes is 4.

-m 1

Indicates that the minimum number of TCPL handler processes is 1.

-L //deimos:2994

Indicates the internal URL address used by TCPL and TCPH for their own communication.

-n //deimos:2992

Indicates the URL address where the TN3270 terminals connect to TCPL.

For the ARTCNX server:

SRVGRP

Is the Tuxedo Group Name to which ARTCNX belongs.

SRVID

Is the identifier of a Tuxedo Server of ARTCNX.

CONV=Y

Indicates that this server operates in a conversational mode.

MIN=1 and MAX=1

Indicates that only one instance of this server must be run.

REPLYQ=Y

Indicates that this server will respond.

RQADDR=QCNX015

Name of the Tuxedo queue used for the responses.

CLOPT

Is a quoted text string passed to the server containing its parameters

- o Indicates the file is used for the standard output messages of the server.
- e Indicates the file is used for the error output messages of the server.
- r Is a Tuxedo parameter used to produce statistical reports.

The Mandatory Server Groups

To be started, a Tuxedo Server must be defined in a Tuxedo Server Group previously defined in the ubbconfig file. As the ARTTCPL and ARTCNX servers are mandatory, verify that their groups are defined, present and not commented-out, in the ubbconfig file.

In our example, ARTTCPL belongs to the Tuxedo Server Group TCP00 (SRVGRP=TCP00) and ARTCNX belongs to the Server Group (SRVGRP=GRP01); therefore the ubbconfig file contains these two Server Group definitions in the following example:

Listing 3-4 Server Group Definitions

```
*GROUPS
DEFAULT:      LMID=KIXR

# Applicative groups
TCP00         LMID=KIXR
              GRPNO=1
              TMSCOUNT=2

GRP01
  GRPNO=11
  ENVFILE="/home2/work9/demo/config/tux/envfile"
```

Where:

***GROUPS**

Tuxedo ubbconfig Keyword indicating definitions of Servers Groups.

LMD=

Name of the CICS.

GRPNO=

Tuxedo Group.

TMSCOUNT=

Number of Tuxedo Transaction Manager Servers.

ENVFILE

Path of the Tuxedo envfile.

The Optional Server Groups

These groups are used to contain the optional servers. The first group is used by the Tuxedo Server Servers Groups: ARTSTRN, ARTSTR1, ARTATRN, ARTATR1, ARTTSQ used by CICS Applications. The second one is used only for TS QUEUE management.

The CICS Runtime Resource Configuration Files

All of the following files must exist in the `$(KIXCONFIG)` path, even when empty, for CICS Runtime to be operational.

The Mandatory Populated Files1. The `typeterms.desc` Configuration File

This file used by the TCP servers, describes the different kinds of terminals used with a 3270 terminal or emulator.

Listing 3-5 typeterm Description Example

```
[typeterm]
name=IBM-3279-5E
color=YES
defscreencolumn=80
defscreenrow=24
```

Initial Configuration of the CICS Runtime

```
description="IBM 327x family terminal"  
highlight=YES  
logonmsg=YES  
outline=NO  
swastatus=ENABLED  
uctran=NO  
userarealen=0
```

Where

[typeterm]

Keyword to define a terminal type.

name=

Type of terminal.

color=YES

Indicates whether the terminal uses extended color attributes.

defscreencolumn= 80

Number of columns of the terminal.

defscreenrow=24

Number of rows of the terminal

description="..."

Comment about the terminal.

highlight=YES

Indicates that this terminal supports the highlight feature.

logonmsg=YES

Indicates that "Good Morning" (CSGM) transaction is automatically started on the terminal at logon time.

outline=NO

Indicates that this terminal does not support field outlining.

swastatus=ENABLED

Indicates that this terminal type is available for use by the system.

uctran=NO

Indicates that the lowercase alphabetic characters are not to be translated to uppercase

userarealen=0

The terminal control table user area (TCTUA) area size for the terminal.

2. The `mapsets.desc` Configuration File

This file must contain at least the following definition to start the CSGM transaction and see the Good Morning screen.

Listing 3-6 `mapsets.desc` Example

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

Where:

name=

Is the logical mapset name used inside the programs in the `EXEC CICS SEND/RECEIVE MAP(map name) MAPSET(mapset name) ... END-EXEC` statements.

filename=

Is the physical path containing the binary file resulting from the compilation of a mapset file source coded in a CICS z/OS BMS format.

Note: For the particular case of the ABANNER system mapset, the filename is located under the `#{KIXDIR}` directory. The bracketed text `<KIXDIR>` must be replaced by the value of the `#{KIXDIR}` variable of your UNIX `~/ .profile` system file.

In our example the result will be:

Listing 3-7 `mapsets.desc` Example with `#{TUXDIR}` Substitution

```
[mapset]
```

```
name=ABANNER
```

```
filename=/product/art11gR1/Cics_RT/sysmap/abanner.mpdef
```

The Optional Initially Populated Files

All the following files can be initially left empty:

1. The transclasses.desc Configuration File
2. The transactions.desc Configuration file
3. The programs.desc Configuration File
4. The tsqmodel.desc Configuration File
5. The mapsets.desc Configuration File

The contents and use of these files is described later.

Note: If these files are left empty, when Tuxedo launches the CICS Runtime servers, some error messages "CMDTUX_CAT:1685: ERROR: Application initialization failure" could be displayed after the boot message of the ARTSTRN, ARTSTR1, ARTATRN and ARTATR1 servers indicating that the CICS Runtime considers this to be an anomaly.

The real number and type of servers displaying these messages depends on the servers initially launched by your ubbconfig file.

In this case, the servers concerned will not be mounted.

For the moment, ignore these error messages, they do not impact the Initial Setting.

Verifying the Initial Setting Configuration

Using the Tuxedo tadmin psr Commands

Once all the files have been modified (and compiled for the ubbconfig), stop and restart Tuxedo to take their modifications into account.

The first control is to check that they are individually correctly accepted by Tuxedo and Oracle by a visual control of the boot messages of the Tuxedo CICS Runtime Tuxedo servers.

Once this first check is made, you can enter the Tuxedo `tmadmin psr` command to check that all the CICS Runtime servers are running and that their messages conform to the Tuxedo documentation and this document.

When the mandatory servers `ARTADM`, `ARTTCPL`, and `ARTCNX` *only* are started, the following messages are displayed:

Listing 3-8 `tmadmin psr` Command Example

```
# tmadmin
tmadmin - Copyright (c) 2007-2010 Oracle.
Portions * Copyright 1986-1997 RSA Data Security, Inc.
All Rights Reserved.
Distributed under license by Oracle.
Tuxedo is a registered trademark.

> psr
Prog Name      Queue Name    Grp Name      ID RqDone Load Done Current Service
-----
BBL            200933        KIXR           0      1      50 ( IDLE )
ARTTCPL 00001.00101 TCP00          101      0      0 ( IDLE )
ARTCNX        QCNX015        GRP01          15      3     150 ( IDLE )

> quit
#
```

Note: The BBL Server is a Tuxedo System Server which can be compared to a CICS server on z/OS.

Using the Tuxedo tadmin psc Commands

You can also check that the required Tuxedo services are running using the `tadmin psc` command.

These services should include the System Transactions managed by CICS Runtime:

- CSGM: The Good Morning Screen
- CESN: Sign On transaction
- CESF: Sign Off transaction

Listing 3-9 tadmin psc Command Example

```
# tadmin
tadmin - Copyright (c) 2007-2010 Oracle.
Portions * Copyright 1986-1997 RSA Data Security, Inc.
All Rights Reserved.
Distributed under license by Oracle.
Tuxedo is a registered trademark.

> psc
Service Name Routine Name Prog Name Grp Name ID Machine # Done Status
-----
authfail      cnxsvc      ARTCNX   GRP01   15    KIXR    0 AVAIL
CESF          cnxsvc      ARTCNX   GRP01   15    KIXR    0 AVAIL
CESN          cnxsvc      ARTCNX   GRP01   15    KIXR    0 AVAIL
CSGM          cnxsvc      ARTCNX   GRP01   15    KIXR    2 AVAIL
disconnect    cnxsvc      ARTCNX   GRP01   15    KIXR    0 AVAIL
connect       cnxsvc      ARTCNX   GRP01   15    KIXR    1 AVAIL

> quit
```

#

Note: From a certain point of view, this Tuxedo command is equivalent to the z/OS CICS system transaction `CEMT I TRAN(...)` which allows you to display the available transactions in a given z/OS CICS environment.

Using the CSGM CICS Good Morning Transaction

Once this first audit is made, you can access CICS Runtime with a 3270 Terminal or Emulator using the following URL address `${HOSTNAME}:${TCPNETADDR}`.

Where:

`${HOSTNAME}`

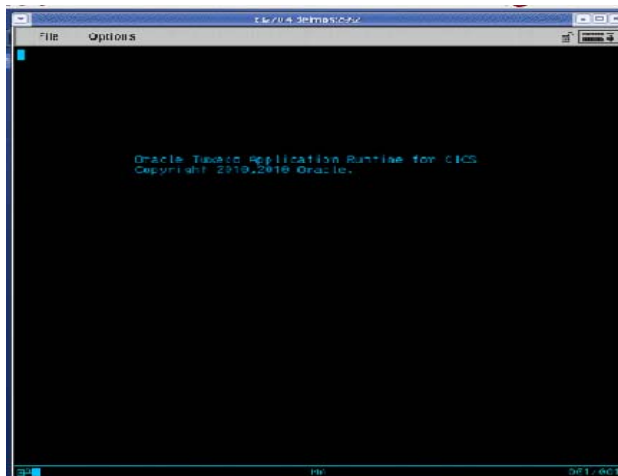
Is the System UNIX variable containing the name of the UNIX machine on which you are running CICS Runtime.

`${TCPNETADDR}`

Is the port number specified by the `-n` parameter of the `ARTTCP` server in the Oracle Tuxedo `UBBCONFIG` file.

The following screen is displayed on a UNIX X11 Window after running the command `# x3270 deimos:2992`:

Figure 3-1 Screen After Running the Command `#3270 deimos:2992`



Initial Configuration of the CICS Runtime

Successfully displaying this screen signifies you can continue implementing CICS applications using CICS Runtime.

Implementing CICS Applications

This chapter contains the following sections:

- [Presentation of the z/OS Simple Application](#)
- [Verifying the CICS Application Installation](#)
- [Presentation of Simple Application on COBOL-IT / BDB](#)
- [Implementing Synchronous CICS Transactions With a Limited Number of Parallel Instances](#)
- [Implementing Asynchronous CICS Non-Delayed Transactions](#)
- [Implementing Asynchronous CICS Delayed Transactions](#)
- [Implementing CICS Application Using Temporary Storage \(TS\) Queues](#)
- [Implementing Distributed Program Link \(DPL\)](#)
- [Implementing CICS Common Work Area \(CWA\)](#)
- [Implementing a CICS Transaction Work Area \(TWA\)](#)
- [Implementing CICS Transaction Trigger Monitor \(ARTCKTI\)](#)
- [CICS Runtime Logs](#)
- [The CICS Runtime Server Logs](#)
- [Disabling and Enabling Programs](#)

Presentation of the z/OS Simple Application

Introduction

This application was initially developed on a z/OS platform implementing COBOL programs used in batch and CICS contexts with VSAM and QSAM files and DB2 tables.

Data was unloaded from z/OS and converted and reloaded on a UNIX platform using Oracle Tuxedo Application Rehosting Workbench.

The language components were converted or translated from z/OS to UNIX by Oracle Tuxedo Application Rehosting Workbench.

These components use two major Oracle Tuxedo Application components, Batch Runtime and CICS Runtime, to emulate the technical centralized features of their original z/OS environment. Here, we will focus on the particular case of the CICS Runtime implementing COBOL Programs using CICS statements and DB2 statements.

This Simple Application manages the customers of a company thru a set of classical functions like creation, modification and deletion.

Description of the CICS Simple Application Components

All of the CICS components are declared with the same name, in the z/OS CICS CSD File. All of the resource declarations are made inside a z/OS CICS GROUP named `PJ01TERM`. This group is declared in the z/OS CICS LIST `PJ01LIST` used by CICS at start up to be automatically installed.

Mapsets

Table 4-1 Simple Application Mapsets

Name	Description
RSSAM00	Customer maintenance entry menu
RSSAM01	Customer data inquiry screen
RSSAM02	Customer data maintenance screen (create, update and delete customer)
RSSAM03	Customer list screen

Programs

Table 4-2 Simple Application Programs

Name	Description
RSSAT000	Customer maintenance entry program
RSSAT001	Customer data inquiry program
RSSAT002	Customer data maintenance program (new customer, update and delete customer)
RSSAT003	Customer list program

Transactions Codes

Table 4-3 Simple Application Transactions Codes

Name	Description
SA00	Main entry transaction code (program RSSAT000)
SA01	Customer inquiry (program RSSAT001)
SA02	Customer maintenance (program RSSAT002)
SA03	Customer list (program RSSAT003)

VSAM File

Table 4-4 Simple Application VSAM File

DDName	DataSetName	Description
ODCSF0	PJ01AAA.SS.VSAM.CUSTOMER	VSAM Main Customer File

Configuring a Standard CICS Application With CICS Runtime

The first example uses the CICS Simple File-to-Oracle application which uses only a z/OS VSAM File converted into a UNIX Oracle Table.

In our example, all of the UNIX components resulting from platform migration are stored in the `trf` directory.

The COBOL programs and BMS mapsets should be compiled and available as executable modules in the respective directories `${HOME}/trf/cobexe` and `${HOME}/trf/MAP_TCP`.

CICS Simple File-to-Oracle Application UNIX Components

COBOL Program Files

The `${HOME}/trf/cobexe` directory contains the Simple Application CICS executable programs:

- `${HOME}/trf/cobexe/RSSAT000.gnt`
- `${HOME}/trf/cobexe/RSSAT001.gnt`
- `${HOME}/trf/cobexe/RSSAT002.gnt`
- `${HOME}/trf/cobexe/RSSAT003.gnt`

The Mapset Files

The `${HOME}/trf/MAP_TCP` directory contains the Simple Application Data z/OS BMS mapsets compiled:

- `${HOME}/trf/MAP_TCP/RSSAM00.mpdef`
- `${HOME}/trf/MAP_TCP/RSSAM01.mpdef`
- `${HOME}/trf/MAP_TCP/RSSAM02.mpdef`
- `${HOME}/trf/MAP_TCP/RSSAM03.mpdef`

CICS Runtime Configuration

For a standard application, in addition to the initial settings, the following CICS resources in the same Group must be implemented:

- Basic CICS transactions (synchronous and simultaneous).
- COBOL Programs without SQL statements, CICS TS queues.
- Mapsets.
- VSAM file (logical name and associated data accessors).

To configure these resources:

1. Declare these resources in their respective CICS Runtime Resource Configuration File.
2. Configure the CICS Runtime Tuxedo Servers Groups and Servers to manage these resources. See [Reference](#) for a full description of which configuration files are used with each server.

Declaring CICS Resources to the CICS Runtime

Each resource is declared in the file corresponding to its type (program, transaction ...). Each resource defined in a resource file must belong to a group.

In the following examples using the CICS Simple File-to- Oracle Application, we will use the CICS Runtime Group name `SIMPAPP` and all our `*.desc` files will be located in the `${home}/trf/config/resources` directory.

Note: In these configuration files, each line beginning with a "#" is considered as a comment and is not processed by CICS Runtime

Declaring CICS Transactions Codes

These declarations are made by filling the `transactions.desc` file for each transaction you have to implement.

For each transaction you have to declare in a csv format

1. The name of the transaction (mandatory).
2. The CICS Runtime Group name (mandatory).
3. A brief description of the transaction (optional, at least one blank).
4. The name of the program started by this transaction (mandatory).

In the File-to-Oracle Simple Application example, we have to declare four transactions: `SA00`, `SA01`, `SA02` and `SA03` in the `SIMPAPP` Group, starting the corresponding COBOL programs `RSSAT000`, `RSSAT001`, `RSSAT002` and `RSSAT003`.

Once filled, the `transactions.desc` file looks like this:

Listing 4-1 Simple Application transactions.desc File

```
#Transaction Name ;Group Name ; Description ;Program Name
SA00;SIMPAPP; Home Menu Screen of the Simple Application;RSSAT000
```

```
SA01;SIMPAPP; Customer Detailed Information Screen of the Simple
Application;RSSAT001

SA02;SIMPAPP; Customer Maintenance Screen of the Simple
Application;RSSAT002

SA03;SIMPAPP; Customer List of the Simple Application;RSSAT003
```

Declaring a CICS COBOL or C Program

All the programs used by the transactions previously declared, directly or indirectly through EXEC CICS statements like LINK, XCTL, START ... must be declared in the same Group.

These declarations are made in the `programs.desc` file for each program to implement.

For each program you have to declare in a csv format:

1. The name of the program (mandatory)
2. The CICS Runtime Group name (mandatory)
3. A brief description of the program (optional, at least one blank)
4. The language in which the program is written (C or COBOL (default))

In our Simple Application example, the only programs needed are RSSAT000, RSSAT001, RSSAT002 and RSSAT003 which are all coded in the COBOL language

Once filled, the `programs.desc` file looks like this:

Listing 4-2 Simple Application programs.desc File

```
#PROGRAM;GROUP;DESCRIPTION;LANGUAGE;

RSSAT000;SIMPAPP; Home Menu Program of the Simple Application ;COBOL

RSSAT001;SIMPAPP; Customer Detailed Information Program of the Simple
Application ;COBOL

RSSAT002;SIMPAPP; Customer Maintenance Program of the Simple Application

RSSAT003;SIMPAPP; Customer List of the Simple Application ;COBOL
```

Note: Nothing is declared in the language field of RSSAT002, meaning that the LANGUAGE of this program is COBOL by default.

Declaring CICS Mapsets

To converse with end-users thru 3270 terminals or emulators, declare to CICS Runtime all of the physical mapsets (*.mpdef file) used in the COBOL programs previously defined thru the specific EXEC CICS statements described above in this document.

These declarations are made by filling the mapsets.desc file for each mapset you have to implement.

The input format of each of your mapset definitions must respect the following format description:

1. On the first free physical line, type the [mapset] keyword.
2. On the next line, enter the keyword name= followed by the name of your mapsets.
3. On the next line, enter the keyword filename= followed by the physical path of your physical mapsets (.mpdef file).

In our Simple Application example, the mapsets used in our COBOL programs are RSSAM00, RSSAM01, RSSAM02 and RSSAM03.

Once filled, the mapsets.desc file looks like this:

Listing 4-3 Simple Application mapsets.desc File

```
[mapset]
name=ABANNER
filename=<KIXDIR>/sysmap/abanner.mpdef [mapset]
name=RSSAM00
filename=<HOME>/demo/MAP_TCP/RSSAM00.mpdef
[mapset]
name=RSSAM01
filename=<HOME>/demo/MAP_TCP/RSSAM01.mpdef
[mapset]
```

```
name=RSSAM02  
filename=<HOME>/demo/MAP_TCP/RSSAM02.mpdef  
[mapset]  
name=RSSAM03  
filename=<HOME>/demo/MAP_TCP/RSSAM03.mpdef
```

Note: The `mapsets.desc` file does not accept UNIX variables, so a fully expanded path must be provided in this file.

- `<KIXDIR>`: must be replaced by the value of the `#{KIXDIR}` variable of the `~/ .profile`.
- `<HOME>`: must be replaced by the value of the `#{HOME}` variable of the `~/ .profile`.

Declaring ISAM Files Resulting From a z/OS VSAM File Conversion

Previously, before declaring one or more files to CICS Runtime, all the physical components, files, accessor programs, COBOL Copybooks etc. must have been generated by the Oracle Tuxedo Application Rehosting Workbench Data components.

Among all the components built or converted by the Oracle Tuxedo Application Rehosting Workbench Data components, only accessor programs on converted VSAM files are used by CICS Runtime. The reason is that, once migrated, no file can be directly accessed. The file can only be accessed indirectly through an accessor program dedicated to the management of this file (one and only one accessor program per source file).

The Simple Application uses only the `CUSTOMER` Oracle table, resulting from the Oracle Tuxedo Application Rehosting Workbench Data Conversion of the z/OS VSAM KSDS file `PJ01AAA.SS.VSAM.CUSTOMER`.

So, for our File-to-Oracle application example, we have only one accessor, `RM_ ODCSF0` (RM for Relational Module), to declare to CICS Runtime.

Note: `ODCSF0` represents the logical name previously defined in CICS that pointed to the physical file name `PJ01AAA.SS.VSAM.CUSTOMER`. Consequently, it is also the only file name known from the CICS COBOL program to access this file by `EXEC CICS` statements.

To Declare the ISAM Migrated Files:

1. Modify the Tuxedo envfile adding a new variable, if not already present, describing all the VSAM/ISAM files used in the programs previously defined.

For our Simple Application example the following line must be entered, (for simplicity, we have located the file in the same place as the ubbconfig, envfile and tuxconfig files but this is not mandatory).

```
DD_VSAMFILE=${HOME}/trf/config/tux/desc.vsam
```

2. If the file does not exist, physically create the `desc.vsam` file at the indicated location.
3. Modify the `desc.vsam` file by adding a new line describing the different information fields used by the accessor in a "csv" format for each accessor/file used.

For our Simple Application example, the following line is entered:

Listing 4-4 Simple Application ISAM File Declaration

```
#DDname;Accessor;DSNOrganization;Format;MaxRecordLength;KeyStart;KeyLength
ODCSF0;ASG_ ODCSF0;I;F;266;1;6
```

Where:

ODCSF0

Is the Data Description Name (logical name) used in the EXEC CICS Statements.

RM_ODCSF0

Is the name of the accessor program managing the access to the Oracle table resulting from the data conversion of the former VSAM File .

I

The Data Set Name organization is indexed

F

Fixed, all the records have the same fixed length format.

266

Maximum record length.

1

Key position in the file (1 means first column or first character).

6

Key length.

Modifying the CICS Runtime Tuxedo Servers

To manage CICS application transactions, in addition to the servers previously defined:

1. Implement the CICS Runtime Tuxedo Server `ARTSTRN`.

This server manages only basic CICS Runtime transactions, those that are the most often used: synchronous (not delayed) and simultaneous (not only one at a time).

2. Indicate to CICS Runtime to start only the transactions belonging to the `SIMPAPP` CICS Runtime Group name.

The following example of a `*SERVERS` section of the Tuxedo `ubbcconfig` file shows the configuration of a `ARTSTRN` server.

Listing 4-5 Simple Application CICS Runtime Server Tuxedo Configuration

```
*SERVERS
...
ARTSTRN    SRVGRP=GRP02
           SRVID=20
           CONV=Y
           MIN=1 MAX=1 RQADDR=QKIX110 REPLYQ=Y
           CLOPT=" -o /home2/work9/demo/Logs/TUX/sysout/stdout_strn -e
/home2/work9/demo/Logs/TUX/sysout/stderr_strn -r -- -s KIXR -l SIMPAPP "
...
```

Where

***SERVERS**

Tuxedo `ubbcconfig` Keyword indicating a Server Section definition.

SRVGRP

Is the Tuxedo Group Name to which `ARTSTRN` belongs.

SRVID

Is the identifier of a Tuxedo Server of `ARTSTRN`.

CONV=Y

Indicates that this server operates in a conversational mode.

MIN=1 and MAX=1

Indicates that only one instance of this server must be run.

REPLYQ=Y

Indicates that this server will respond.

RQADDR=QCNX015

Name of the Tuxedo queue used for the responses.

CLOPT

Is a quoted text string passed to the server containing its parameters.

-o

Indicates the file used for the standard output messages of the server.

-e

Indicates the file used for the error output messages of the server.

-r

Is a Tuxedo parameter used to provide statistical reports.

-s KIXR

Indicates the CICS Runtime name where the KIXR transaction is run.

-l SIMAPP

Indicates that only the transaction of the SIMAPP group are to be selected.

Modifying the CICS Runtime Tuxedo Servers Groups

To be started, the `ARTSTRN` server must be defined in a Tuxedo Server Group previously defined (and not commented) in the `ubbconfig` file.

In our example, `ARTSTRN` belong to the Tuxedo Server Group `GRP02` (`SRVGRP=GRP02`).

Listing 4-6 Simple Application CICS Runtime Tuxedo Servers Groups Example:

```
*GROUPS
...
GRP02
    GRPNO=12
    ENVFILE="/home2/work9/demo/config/tux/envfile"
    TMSNAME="TMS_ORA"
...
```

Where

***GROUPS**

Tuxedo ubbconfig Keyword indicating a Server Section Group section definition.

GRPNO=

Tuxedo Group.

ENVFILE=

Path of the Tuxedo envfile.

TMSNAME=

Name of the Tuxedo Transaction Manager Server executable.

Verifying the CICS Application Installation

Using the Tuxedo tadmin psr Commands

Enter the Tuxedo `tadmin psr` command to check that all of the CICS Runtime required servers (ARTTCPL, ARTCNX, and ARTSTRN) are running and that their messages conform to the Tuxedo documentation and this document.

Listing 4-7 tadmin psr Simple Application Installation Check

```
# tadmin
```

tmadmin - Copyright (c) 2007-2010 Oracle.

Portions * Copyright 1986-1997 RSA Data Security, Inc.

All Rights Reserved.

Distributed under license by Oracle.

Tuxedo is a registered trademark.

> psr

Prog Name	Queue Name	Grp Name	ID	RqDone	Load Done	Current Service
-----	-----	-----	--	-----	-----	-----
BBL	200933	KIXR	0	2	100	(IDLE)
ARTTCPL	00001.00101	TCP00	101	0	0	(IDLE)
ARTCNX	QCNX015	GRP01	15	2	100	(IDLE)
ARTSTRN	QKIX110	GRP02	20	6	300	(IDLE)

> quit

#

Using the Tuxedo tmadmin psc Commands

Another possible check can be made by entering the Tuxedo `tmadmin psc` command to display all the different Tuxedo Services running.

In addition to the CICS Runtime System transactions/services (`CSGM`, `CESN`, `CESF` ...), you can now see the transaction codes of your CICS Runtime application `SA00`, `SA01`, `SA02` and `SA03`

Listing 4-8 tmadmin psc Simple Application Installation Check

```
# tmadmin
```

```
tmadmin - Copyright (c) 2007-2010 Oracle.
```

```
Portions * Copyright 1986-1997 RSA Data Security, Inc.
```

Implementing CICS Applications

All Rights Reserved.

Distributed under license by Oracle.

Tuxedo is a registered trademark.

```
> psc
```

Service Name	Routine Name	Prog Name	Grp Name	ID	Machine	# Done	Status
authfail	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
CESF	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
CESN	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
CSGM	cnxsvc	ARTCNX	GRP01	15	KIXR	1	AVAIL
disconnect	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
connect	cnxsvc	ARTCNX	GRP01	15	KIXR	1	AVAIL
SA03	kixsvc	ARTSTRN	GRP02	20	KIXR	3	AVAIL
SA02	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL
SA01	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL
SA00	kixsvc	ARTSTRN	GRP02	20	KIXR	3	AVAIL

```
> quit
```

```
#
```

Using the CICS Runtime Application

Before using the CICS application, you have to populate the ISAM files accessed by your application. Then, access CICS Runtime with a 3270 Terminal or Emulator, with a UNIX x3270 command. It should be:

```
# x3270 ${HOSTNAME}:${TCPNETADDR}
```

Where:

\$(HOSTNAME)

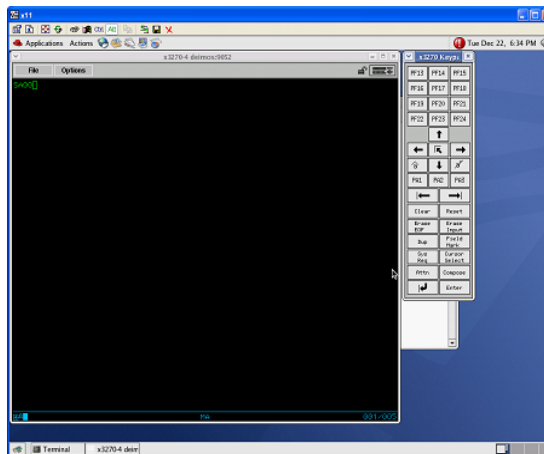
Is the System UNIX variable containing the name of the UNIX machine on which you are running CICS Runtime.

\$(TCPNETADDR)

Is the port number for your UNIX 3270 emulator set up by your Tuxedo Administrator at installation time in the ubbconfig file.

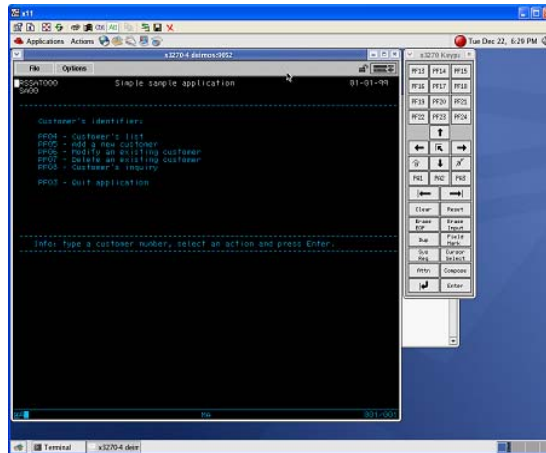
1. You will receive the Good Morning Message.
2. Clear it by pressing the `Clear` key of your 3270 emulator keypad.
3. Type the main transaction code `SA00` (of your CICS Runtime application) in the top left corner:

Figure 4-1 Simple Application transaction Code Entry



4. The main menu of the application is displayed:

Figure 4-2 Simple Application Main Menu



5. Navigate through the screens of the application to check that they are displayed without errors.

Presentation of Simple Application on COBOL-IT / BDB

Based on BDB with XA protocol, the CICS COBOL programs compiled by COBOL IT can access the indexed ISAM files which are converted from Mainframe VSAM files through the ART Workbench. The following sections describes the configurations should be done in ART CICS Runtime to enable this application.

Configuring ubbconfig File in CICS Runtime

Add the MRM parameter in the group entry of *GROUPS and *RMS section in Tuxedo ubbconfig file. See the following example:

Listing 4-9 Adding MRM Parameter in ubbconfig File Example

```
*GROUPS
GRP02
GRPNO=12
MRM=Y
```



```
*RMS  
MRMG_RM1  
RVGRP=GRP02  
RMID=15  
TMSNAME="TMS_BDB"  
OPENINFO="BERKELEY-DB:/home2/work10/data"
```

Where:

***GROUPS**

Tuxedo ubbconfig keyword indicating the definitions of Servers Groups.

GRPNO=

Indicates the Tuxedo Group.

MRM= Y

Indicates that this server group can support multiple resource managers.

***RMS**

Tuxedo ubbconfig keyword indicating the definitions of resource managers.

MRMG_RM1

Indicates the logical name of RMS entry.

RVGRP=

Indicates the name of the group associated with this RM.

RMID=

Indicates the unique ID number of this RM in the group. ID number must be between 1 and 31.

TMSNAME

Indicates the name of the transaction manager server associated with the group specified by SRVGRP.

OPENINFO=

Indicates the resource manager dependent information needed when opening resource manager for the associated group.

Building BDB TMS Server

Add the following text in `$TUXDIR/udataobj/RM` to build the BDB TMS server:

```
# BDB

BDB_HOME=/opt/cobol-it-64-bdb

BERKELEY-DB:db_xa_switch:-L/opt/cobol-it-64-bdb/lib -ldb-5.2

Build BDB TMS

buildtms -r -v BERKELEY-DB -o $APPDIR/TMS_BDB
```

Exporting Variables Before Booting Up ART Servers

Export the following variables explicitly before booting up the ART servers:

- `DD_VSAMFILE`
`export DD_RBDB02=${DATA}/MTWART.ES.SFI.RCIBDB02.BDB0122`
RBDB02 is the logical file name.
- `COB_ENABLE_XA`
`export COB_ENABLE_XA=1`

Implementing Synchronous CICS Transactions With a Limited Number of Parallel Instances

In some particular cases, the number of transactions bearing the same transaction code running simultaneously has to be limited, for performance constraints for example.

On z/OS, this limit cannot be defined in the transaction resource itself but is defined in a distinct resource named `TRANCLASS` (transaction class) that contains a specific `MAXACTIVE` parameter describing the maximum number of concurrent instances of the same transaction.

To link a transaction to a transaction class, to inherit its parameters, especially the `MAXACTIVE` parameter, the z/OS CICS transaction resource has a `TRANCLASS` field containing the name of the `TRANCLASS` resource.

This instance management is performed differently on UNIX with CICS Runtime. The maximum number of transactions running concurrently is defined by the number of servers offering the same transaction. This maximum number and the minimum number are indicated respectively in

the `MAX` and `MIN` parameters of the `ARTSTRN` definition in the `*SERVERS` section of the Tuxedo file `ubbconfig`.

It means that the `maxactive` parameter is not taken in account to manage these limits except in the following very particular case:

The Special Case of Transaction Classes With `MAXACTIVE=1`

The `MAXACTIVE=1` is really an exception in this management because it indicates that no concurrent transaction belonging to these kind of transaction classes can be run simultaneously.

To manage this very particular case of sequential transactions, a Tuxedo CICS Runtime feature must be configured

Modification of the `ubbconfig` File for Sequential Transactions

All of the transactions linked to transactions classes with a `MAXACTIVE` superior or equal to 2 are managed by the CICS Runtime Tuxedo Server `ARTSTRN` and do not required modifying anything else. For the transactions with a `MAXACTIVE` parameter set to 1, an CICS Runtime Tuxedo Server named `ARTSTR1` is dedicated to their specific management.

To activate this server, modify the `ubbconfig` file to add this server in the `*SERVERS` section:

Listing 4-10 Adding a `ARTSTR1` Server to `ubbconfig`

```
*SERVERS
...
ARTSTR1    SRVGRP=GRP02
           SRVID=200
           CONV=Y
           MIN=1 MAX=1
           CLOPT=" -o /home2/work9/demo/Logs/TUX/sysout/stdout_str1 -e
/home2/work9/demo/Logs/TUX/sysout/stderr_str1 -r -- -s KIXR -l SIMPAPP"
...
```

Where:

***SERVERS**

Tuxedo ubbconfig Keyword indicating a Server Section definition.

SRVGRP

Is the Tuxedo Group Name to which ARTSTR1 belongs.

SRVID

Is the identifier of a ARTSTR1 Tuxedo Server.

CONV=Y

Indicates that this server operates in a conversational mode.

MIN=1 and MAX=1

Are mandatory and indicate that only one instance of this server must run.

CLOPT

Is a quoted text string passed to the server containing the parameters:

-o

Indicates the file used for the standard output messages of the server.

-e

Indicates the file used for the error output messages of the server.

-r

Is a Tuxedo parameter used to produce statistical reports.

-s

KIXR indicates the CICS Runtime name where the KIXR transaction is run.

-l SIMAPP

Indicates that only the transaction of the SIMAPP group are to be selected.

Note: All of the CICS Runtime Transaction Servers (ARTSTRN, ARTSTR1, ARTATRN and ARTATR1) share the same CICS Runtime Transaction Group Servers, no modifications are required to the ubbconfig Server Group Section (*GROUPS).

Modifying the tranclasses.desc File

For ART CICS, concurrent transactions do not really need to be bound to transactions classes with MAXACTIVE parameters superior or equal to two because parallelism is the default behavior.

For sequential transactions, it is mandatory because it is the only way to declare these transactions to CICS Runtime. Declare specific transaction classes defined with a MAXACTIVE=1 parameter.

Like the other CICS Runtime resources, this one must belong to an CICS Runtime Group name. For each TRANCLASS, declare in a csv format:

1. The name of the transaction class (mandatory)
2. The CICS Runtime Group name (mandatory)
3. A brief description of the transaction class (optional, at least one blank)
4. The maximum number of the same transaction to RUN (MAXACTIVE).

Note: The MAXACTIVE parameter should be understood like a binary switch:

- MAXACTIVE=1 <=> Sequential transaction class (mandatory).
- MAXACTIVE>1 (all the values are at this step equivalent) <=> Concurrent transaction (optional).

Examples:

```
TRCLASS1;SIMPAPP ; Traclass with maxactive set to 1; 1
TRCLASS2;SIMPAPP ; Traclass with maxactive set to 2; 2
TRCLASS10;SIMPAPP ; Traclass with maxactive set to 10; 10
```

The first tranclass TRCLASS1 has its maxactive parameter equal to 1, indicating that all the transaction belonging to this tranclass must be managed sequentially by the ARTSTR1.

The two last tranclasses, TRCLASS2 and TRCLASS10, are in fact similar because their maxactive parameters are superior to 1 indicating that the transactions belonging to these tranclasses can run concurrently managed by the ARTSTRN server.

Note: These two last definitions are optional. Their absence has the same meaning.

Modifying the transactions.desc File

In addition to the first four mandatory fields of this csv format file (Transaction name, Group name, Description, Program name), you must add a twelfth field: TRANCLASS (Transaction Class name).

The TRANCLASS field must be separated from the Program field by eight semicolon characters (;) with at least one blank between each of them.

In our example, let us suppose that the CICS Runtime Simple Application must have the following MAXACTIVE limits:

- SA00: MAXACTIVE=0

- SA01: MAXACTIVE=1
- SA02: MAXACTIVE=2
- SA03: MAXACTIVE=10

Then these transactions must be linked to the following tranclasses that we have previously defined:

- SA00: none
- SA01: TRCLASS1
- SA02: TRCLASS2
- SA03: TRCLASS10

Once modified, the `transactions.desc` file will look like this:

Listing 4-11 Example transactions.desc File

```
#Transaction Name ;Group Name ; Description ;Program Name
SA00;SIMPAPP; Home Menu Screen of the Simple Application;RSSAT000
SA01;SIMPAPP; Customer Detailed Information Screen of the Simple ;
Application;RSSAT001; ; ; ; ; ; ; TRCLASS1
SA02;SIMPAPP; Customer Maintenance Screen of the Simple
Application;RSSAT002; ; ; ; ; ; ; TRCLASS2
SA03;SIMPAPP; Customer List of the Simple Application;RSSAT003; ; ; ; ; ;
; TRCLASS10
```

Notes:

- No modification is made to SA00 meaning that no transaction class is associated with this transaction code. It means that this transaction is not associated with a MAXACTIVE=1 parameter and so is not sequential.
- SA02 and SA03 are associated to transaction classes, respectively TRCLASS2 and TRCLASS10, defined with MAXACTIVE >= 2. Knowing that these transactions are not required, the result would be the exactly the same if SA02 and SA03 were defined like SA00 without transaction classes.

- SA01, which can run sequentially, is the only one where the transaction class field is mandatory. Verify that its associated transaction class, TRCLASS1, is really defined with a MAXACTIVE=1.

Checking the ARTSTR1 Configuration

Using the Tuxedo tadmin psr Commands

The ARTSTR1, is shown below:

Listing 4-12 Checking the ARTSTR1 Server with the tadmin psr Commands

```
# tadmin
tadmin - Copyright (c) 2007-2010 Oracle.
Portions * Copyright 1986-1997 RSA Data Security, Inc.
All Rights Reserved.
Distributed under license by Oracle.
Tuxedo is a registered trademark.

> psr
Prog Name      Queue Name  Grp Name      ID RqDone Load Done Current Service
-----
ARTSTR1        00012.00200 GRP02          200     0       0 ( IDLE )
BBL            200933      KIXR           0       3      150 ( IDLE )
ARTTCPL 00001.00101 TCP00          101     0       0 ( IDLE )
ARTCNX         QCNX015     GRP01          15     0       0 ( IDLE )
ARTSTRN        QKIX110     GRP02          20     0       0 ( IDLE )

> quit
#
```

Using the Tuxedo tadmin psc Commands

No new service or transaction should appear.

In our example where ARTSTRN was the only server running, we can see that nothing changed when ARTSTR1 is also activated.

Listing 4-13 Checking the ARTSTRN Server with the tadmin psc Commands

```
# tadmin
tadmin - Copyright (c) 2007-2010 Oracle.
Portions * Copyright 1986-1997 RSA Data Security, Inc.
All Rights Reserved.
Distributed under license by Oracle.
Tuxedo is a registered trademark.

> psc
```

Service Name	Routine Name	Prog Name	Grp Name	ID	Machine	# Done	Status
authfail	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
CESF	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
CESN	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
CSGM	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
disconnect	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
connect	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
SA03	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL
SA02	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL
SA01	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL
SA00	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL


```
> quit
#
```

Implementing Asynchronous CICS Non-Delayed Transactions

These transactions are launched by specific `CICS EXEC CICS START TRANSID` requests coded in the CICS programs that are not using `DELAY` or `TIME` parameters to delay their execution.

If at least one of your programs contains this kind of statement, install, and activate some new features of CICS Runtime Tuxedo Servers without changing any other settings.

Modifying the Tuxedo `ubbconfig` File to Manage Asynchronous Transactions

The file is modified in the same manner as for the `ARTSTRN` and the `ARTSTR1` servers, except the "s" (synchronous) character used to prefix the name of these servers should be replaced by the "a" (asynchronous) character.

Using Parallel Asynchronous Transactions

To use parallel asynchronous transactions, with a `MAXACTIVE` parameter strictly superior to one, the dedicated server is the `ARTATRN`. Please refer to the section describing the installation of the `ARTSTRN` server to install the `atrn_server`.

To check your settings you can use also the `tadmin psr` and `psc` commands.

For the Simple Application example we can see that:

- The `psr` command shows that a new server is running `ARTATRN`.
- The `psc` command shows that five new services are running, one is dedicated to the asynchronous transaction while each synchronous transaction (`SA00` to `SA03`) is duplicated (`ASYNC_SA00` to `ASYNC_SA03`) to allow them to run in an asynchronous mode.

Listing 4-14 `tadmin` Commands Showing Parallel Asynchronous Transactions

```
# tadmin
```

Implementing CICS Applications

tmadmin - Copyright (c) 2007-2010 Oracle.

Portions * Copyright 1986-1997 RSA Data Security, Inc.

All Rights Reserved.

Distributed under license by Oracle.

Tuxedo is a registered trademark.

> psr

Prog Name	Queue Name	Grp Name	ID	RqDone	Load Done	Current	Service
-----	-----	-----	--	-----	-----	-----	-----
ARTSTR1	00012.00200	GRP02	200	0	0	(IDLE)	
BBL	200933	KIXR	0	4	200	(IDLE)	
ARTTCPL	00001.00101	TCP00	101	0	0	(IDLE)	
ARTCNX	QCNX015	GRP01	15	0	0	(IDLE)	
ARTSTRN	QKIX110	GRP02	20	0	0	(IDLE)	
ARTATR	QKIXATR	GRP02	30	0	0	(IDLE)	

> psc

Service Name	Routine Name	Prog Name	Grp Name	ID	Machine	# Done	Status
-----	-----	-----	-----	--	-----	-----	-----
authfail	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
CESF	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
CESN	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
CSGM	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
disconnect	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
connect	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
SA03	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL
SA02	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL

```
SA01      kixsvc      ARTSTRN   GRP02     20      KIXR      0 AVAIL
SA00      kixsvc      ARTSTRN   GRP02     20      KIXR      0 AVAIL
ASYNC_QUEUE ASYNC_QUEUE ARTATRNL  GRP02     30      KIXR      0 AVAIL
ASYNC_SA03 atrsvc      ARTATRNL  GRP02     30      KIXR      0 AVAIL
ASYNC_SA02 atrsvc      ARTATRNL  GRP02     30      KIXR      0 AVAIL
ASYNC_SA01 atrsvc      ARTATRNL  GRP02     30      KIXR      0 AVAIL
ASYNC_SA00 atrsvc      ARTATRNL  GRP02     30      KIXR      0 AVAIL
```

```
> quit
```

```
{deimos:work9}-/home2/work9/demo/config/tux#{deimos:work9}-/home2/work9/de
mo/config/tux#
```

Using Non-Parallel Asynchronous Transactions

To use non-parallel asynchronous transactions, with a `MAXACTIVE` parameter exactly equal to one, the dedicated server is `ARTATR1`.

Please refer to the section describing the reasons and the installation of the `ARTSTR1` server to install the `ARTSTR1` server.

To check your setting, you can use also the Tuxedo `tmadmin psr` and `psc` commands

For the Simple Application example we can see that:

- The `psr` command shows that a new server is running `ARTATR1`.
- The `psc` command shows that no new services are running.

Listing 4-15 `tmadmin` Commands Showing non-parallel Asynchronous Transactions

```
# tmadmin
tmadmin - Copyright (c) 2007-2010 Oracle.
Portions * Copyright 1986-1997 RSA Data Security, Inc.
All Rights Reserved.
```

Implementing CICS Applications

Distributed under license by Oracle.

Tuxedo is a registered trademark.

> psr

Prog Name	Queue Name	Grp Name	ID	RqDone	Load Done	Current	Service
ARTATR1	00012.00300	GRP02	300	0	0	(IDLE)	
ARTSTR1	00012.00200	GRP02	200	0	0	(IDLE)	
BBL	200933	KIXR	0	4	200	(IDLE)	
ARTTCPL	00001.00101	TCP00	101	0	0	(IDLE)	
ARTCNX	QCNX015	GRP01	15	0	0	(IDLE)	
ARTSTRN	QKIX110	GRP02	20	0	0	(IDLE)	

> psc

Service Name	Routine Name	Prog Name	Grp Name	ID	Machine	# Done	Status
authfail	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
CESF	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
CESN	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
CSGM	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
disconnect	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
connect	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
SA03	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL
SA02	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL
SA01	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL
SA00	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL

```
> quit
#
```

Implementing Asynchronous CICS Delayed Transactions

These transactions are launched when `ASYNC_QSPACE` for `EXEC START` is set with option `INTERVAL` or `PROTECT`.

If at least one of your programs contains this kind of statement, you need to install and activate some new features of the CICS Runtime Tuxedo Servers without making any other changes to your other settings.

These new features are:

1. The creation of a Tuxedo /Q Queue Space named `ASYNC_QSPACE`.
2. The creation of a Tuxedo /Q Queue named `ASYNC_QUEUE` in `ASYNC_QSPACE`.
3. The activation of the `TMQUEUE` and `TMQFORWARD` servers dedicated to these asynchronous transactions.

Creating the Tuxedo /Q Features

CICS Runtime provides a UNIX script that creates all the Tuxedo /Q components: `mkqmconfig.sh`.

1. Before using the script, define and export in your UNIX `~/.profile` file:
 - The `QMCONFIG` variable `QMCONFIG`- containing the full directory path that stores the Tuxedo /Q Queue Space `ASYNC_QSPACE`.
 - The `KIX_QSPACE_IPCKEY` variable - containing the IPC Key for the Queue Space.

Examples of `~/.profile` variables and values:

```
export QMCONFIG=${HOME}/trf/config/tux/kixqspace
export KIX_QSPACE_IPCKEY=200955
```

2. Execute `mkqmconfig.sh` from the command line to create the Tuxedo /Q features.

Modifying the Tuxedo ubbconfig File to Manage the Tuxedo /Q Queue

1. The GQUEUE Server Group must be added to the ubbconfig file in the *GROUP section.

Listing 4-16 Simple Application Tuxedo Queue ubbconfig Example

```
*GROUPS
...
# /Q
GQUEUE      GRPNO=1000
             TMSNAME=TMS_QM TMSCOUNT=2

OPENINFO="TUXEDO/QM:/home2/work9/demo/config/tux/kixqspace:ASYNC_QSPACE"
...
```

Where:

***GROUPS**

Tuxedo ubbconfig Keyword indicating definitions of Servers Groups.

GRPNO=

Tuxedo Group.

TMSCOUNT=

Number of Tuxedo Transaction Manager Servers.

TMSNAME

Name of the Tuxedo Transaction Manager Server executable.

OPENINFO=

Indicates to the Tuxedo /Q Transaction Manager QM, the QSPACE name to manage and its UNIX absolute path.

2. Then, two servers, TMQUEUE and TMQFORWARD, must be added to the ubbconfig file in the *SERVERS section.

Listing 4-17 Simple Application ubbconfig TMQUEUE and TMQFORWARD Example

```

*SERVERS

...

# /Q

TMQUEUE      SRVGRP=GQUEUE

              SRVID=1010

              GRACE=0 RESTART=Y 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"

...

```

Where:

***SERVERS**

Tuxedo ubbconfig Keyword indicating a Server Section definition.

SRVGRP

Is the Tuxedo Group Name of which belongs ARTSTRN.

SRVID

Is the identifier of a Tuxedo Server of ARTSTRN.

MAXGEN=10

Specifies that the process can have up to 10 server restarts.

GRACE=0

Means there is no limit interval to contain the number of server restarts.

CONV=N

Indicates that this server operates in a non-conversational mode.

CLOPT

Is a quoted text string passed to the server containing its parameters.

Using the `tmadmin psr` and `psc` commands check that four new servers and two new services are running:

Listing 4-18 Simple Application TMQUEUE and TMQFORWARD tmadmin Example

```
# tmadmin
tmadmin - Copyright (c) 2007-2010 Oracle.
Portions * Copyright 1986-1997 RSA Data Security, Inc.
All Rights Reserved.
Distributed under license by Oracle.
Tuxedo is a registered trademark.

> psr
```

Prog Name	Queue Name	Grp Name	ID	RqDone	Load Done	Current Service
ARTATR1	00012.00300	GRP02	300	0	0	(IDLE)
ARTSTR1	00012.00200	GRP02	200	0	0	(IDLE)
BBL	200933	KIXR	0	4	200	(IDLE)
ARTTCPL	00001.00101	TCP00	101	0	0	(IDLE)
ARTCNX	QCNX015	GRP01	15	0	0	(IDLE)
TMS_QM	GQUEUE_TMS	GQUEUE	30001	0	0	(IDLE)
TMS_QM	GQUEUE_TMS	GQUEUE	30002	0	0	(IDLE)
TMQUEUE	01000.01010	GQUEUE	1010	0	0	(IDLE)
TMQFORWARD	01000.01020	GQUEUE	1020	0	0	(IDLE)
ARTSTRN	QKIX110	GRP02	20	0	0	(IDLE)

Implementing Asynchronous CICS Delayed Transactions

```

ARTATR      QKIXATR      GRP02          30          0          0 ( IDLE )

> psc
Service Name Routine Name Prog Name  Grp Name  ID    Machine  # Done Status
-----
TMS          TMS          TMS_QM     GQUEUE 30001    KIXR     0 AVAIL
TMS          TMS          TMS_QM     GQUEUE 30002    KIXR     0 AVAIL
ASYNC_QSPACE TMQUEUE     TMQUEUE     GQUEUE 1010    KIXR     0 AVAIL
authfail     cnxsvc      ARTCNX     GRP01   15      KIXR     0 AVAIL
CESF         cnxsvc      ARTCNX     GRP01   15      KIXR     0 AVAIL
CESN         cnxsvc      ARTCNX     GRP01   15      KIXR     0 AVAIL
CSGM         cnxsvc      ARTCNX     GRP01   15      KIXR     0 AVAIL
disconnect   cnxsvc      ARTCNX     GRP01   15      KIXR     0 AVAIL
connect      cnxsvc      ARTCNX     GRP01   15      KIXR     0 AVAIL
SA03         kixsvc      ARTSTRN    GRP02   20      KIXR     0 AVAIL
SA02         kixsvc      ARTSTRN    GRP02   20      KIXR     0 AVAIL
SA01         kixsvc      ARTSTRN    GRP02   20      KIXR     0 AVAIL
SA00         kixsvc      ARTSTRN    GRP02   20      KIXR     0 AVAIL
ASYNC_QUEUE ASYNC_QUEUE ARTATR     GRP02   30      KIXR     0 AVAIL
ASYNC_SA03   atrsvc      ARTATR     GRP02   30      KIXR     0 AVAIL
ASYNC_SA02   atrsvc      ARTATR     GRP02   30      KIXR     0 AVAIL
ASYNC_SA01   atrsvc      ARTATR     GRP02   30      KIXR     0 AVAIL
ASYNC_SA00   atrsvc      ARTATR     GRP02   30      KIXR     0 AVAIL

> quit
#

```

Implementing CICS Application Using Temporary Storage (TS) Queues

These transactions use CICS programs containing EXEC CICS requests relative to CICS Temporary Storage Queues.

The statements used are EXEC CICS WRITEQ TS ... END-EXEC, EXEC CICS READQ TS ... END-EXEC, EXEC CICS DELETEQ TS ... END-EXEC.

If at least one of your programs contains one of these statements, install and activate the new features of CICS Runtime without changing your other settings.

To manage TS Queues, activate the ARTTSQ CICS Runtime Tuxedo Server.

- To activate this server, add this server to the *SERVERS section of the Tuxedo ubbconfig file:

Listing 4-19 Activating the ARTTSQ in the ubbconfig File

```
*SERVERS
...
ARTTSQ      SRVGRP=GRP02
            SRVID=40
            MIN=1 MAX=1
            CLOPT="-o /home2/work9/demo/Logs/TUX/sysout/stdout_tsq -e
/home2/work9/demo/Logs/TUX/sysout/stderr_tsq -r -- -s KIXR -l SIMPAPP"
...
```

Where:

***SERVERS**

Tuxedo ubbconfig Keyword indicating a Server Section definition.

SRVGRP

Is the Tuxedo Group Name to which ARTTSQ belongs.

SRVID

Is the identifier of a Tuxedo Server of ARTTSQ.

MIN=1 and MAX=1

Indicates that only one instance of this server must be run.

CLOPT

Is a quoted text string passed to the server containing its parameters:

-o

Indicates the following file is used for the standard output messages of the server.

-e

Indicates the following file is used for the error output messages of the servers.

-r

Is a Tuxedo parameter used to have statistical reports.

-s KIXR

Indicates the CICS Runtime name where the transaction runs is KIXR.

-l SIMAPP

Indicates that only the components of the SIMAPP group are to be selected at start up.

Use the Tuxedo `tmadmin psr` and `psc` commands to check that the server is running and that six new services are published:

Listing 4-20 Checking ARTTSQ Server and Services are Running

```
# tmadmin
tmadmin - Copyright (c) 2007-2010 Oracle.
Portions * Copyright 1986-1997 RSA Data Security, Inc.
All Rights Reserved.
Distributed under license by Oracle.
Tuxedo is a registered trademark.

> psr
Prog Name      Queue Name  Grp Name      ID RqDone Load Done Current Service
```

Implementing CICS Applications

```

-----
ARTATR1      00012.00300 GRP02      300      0      0 ( IDLE )
ARTSTR1      00012.00200 GRP02      200      0      0 ( IDLE )
BBL          200933      KIXR          0      3      150 ( IDLE )
ARTTCPL 00001.00101 TCP00      101      0      0 ( IDLE )
ARTCNX       QCNX015      GRP01        15      0      0 ( IDLE )
ARTSTRN      QKIX110      GRP02        20      0      0 ( IDLE )
ARTTSQ       00012.00040 GRP02        40      0      0 ( IDLE )

```

> psc

Service Name	Routine Name	Prog Name	Grp Name	ID	Machine	# Done	Status
authfail	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
CESF	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
CESN	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
CSGM	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
disconnect	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
connect	cnxsvc	ARTCNX	GRP01	15	KIXR	0	AVAIL
SA03	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL
SA02	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL
SA01	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL
SA00	kixsvc	ARTSTRN	GRP02	20	KIXR	0	AVAIL
TSM00004_TSQ	tsqsvc	ARTTSQ	GRP02	40	KIXR	0	AVAIL
TSM00003_TSQ	tsqsvc	ARTTSQ	GRP02	40	KIXR	0	AVAIL
TSM00002_TSQ	tsqsvc	ARTTSQ	GRP02	40	KIXR	0	AVAIL
TSM00001_TSQ	tsqsvc	ARTTSQ	GRP02	40	KIXR	0	AVAIL
TSM00000_TSQ	tsqsvc	ARTTSQ	GRP02	40	KIXR	0	AVAIL

```

TSQUEUE      tsqsvc      ARTTSQ      GRP02      40      KIXR      0 AVAIL

> quit

{deimos:work9}~/home2/work9/demo/config/tux#

```

Implementing Unrecoverable TS Queues

For unrecoverable TS Queues, no integrity is guaranteed by CICS Runtime concerning their content. For example, if an abend occurs at any point during a CICS transaction, work done on this TS is not rolled-back to the last consistency point.

TS Queues are stored in a sequential file in a dedicated directory defined in the `KIX_TS_DIR` UNIX environment variable. This variable is defined and then exported from the `~/.profile` UNIX System File:

```
KIX_TS_DIR=${HOME}/trf/KIXTSDIR
```

Modify the Tuxedo `ubconfig` file to activate the new `ARTTSQ` server dedicated to their management.

Implementing Recoverable TS Queues

For these TS Queues, CICS Runtime guarantees the integrity of their content. For example, if an abend occurs at any point during a CICS transaction, they are rolled-back to the last consistency point, if all is in order, their content is committed to become a new consistency point. These TS Queues are stored in Oracle Tables to benefit from the RDBMS integrity management.

Concerning the TS Queue, there is an enhanced behavior for reading a recoverable TS Queue.

On source CICS z/OS, CICS enqueueing is not invoked for `READQ` TS commands, thereby making it possible for one task to read a temporary storage queue record while another is updating the same record. To avoid this, use explicit enqueueing on the temporary storage queues so that concurrent executing tasks can read and change queues with the same temporary storage identifier.

This behavior also allows one transaction to see or read a record freshly written in a recoverable TS Queue, even before it is committed, and after its rollback.

On target we don't have this limitation, but in particular:

- A reading transaction is not able to see a record that is just added and not yet committed.
- A reading transaction is not able to see a modification to the record that is not yet committed.

To Use Recoverable TS Queues

To use recoverable TS Queues you need to define an Oracle Table to contain the TS Queues. CICS Runtime provides a UNIX script to create all these tables: `crtstable_Oracle`.

1. Before using the script define and export from your UNIX `~/.profile` file

- The `ORA_USER` variable containing the user ID used to connect to Oracle.
- The `ORA_PASSWD` variable containing the associated password.

Examples of `~/.profile` variables and values:

```
export ORA_USER="Oracle_User_1"
export ORA_PASSWD="Oracle_Pswd_1"
```

2. Once the variables have been set, execute the `crtstable_Oracle` script.

3. Then, modify the Tuxedo `ubbconfig` file to modify the Server Group used by ARTTSQ to establish the connection to Oracle in the `*GROUPS` section.

Listing 4-21 Example of the `*GROUP` Section of the Tuxedo `ubbconfig` File Concerning the Server Group `GRP02` used by the ARTTSQ Server.

```
*GROUPS
...
GRP02
    GRPNO=12
    ENVFILE="/home2/work9/demo/config/tux/envfile"
    TMSNAME="TMS_ORA"
    OPENINFO="Oracle_XA:Oracle_XA+Acc=P/work9/work9+SesTm=600+LogDir=/home2/work9/demo/Logs/TUX/xa+DbgFl=0x20"
...
```

Where:

***GROUPS**

Tuxedo ubbconfig Keyword indicating definitions of Servers Groups.

GRPNO=

Tuxedo Group number.

TMSNAME=

Name of the Tuxedo Transaction Manager Server executable.

OPENINFO=

Parameters send to the Oracle_XA Manager.

4. Use the Tuxedo psr and psc commands to check that Oracle is available; three new servers and three new services should be indicated:

Listing 4-22 Simple Application Check For Recoverable TS Queues

```
# tadmin
tadmin - Copyright (c) 2007-2010 Oracle.
Portions * Copyright 1986-1997 RSA Data Security, Inc.
All Rights Reserved.
Distributed under license by Oracle.
Tuxedo is a registered trademark.

> psr
```

Prog Name	Queue Name	Grp Name	ID	RqDone	Load Done	Current Service
ARTATR1	00012.00300	GRP02	300	0	0	(IDLE)
ARTSTR1	00012.00200	GRP02	200	0	0	(IDLE)
BBL	200933	KIXR	0	4	200	(IDLE)
ARTTCPL	00001.00101	TCP00	101	0	0	(IDLE)
TMS_ORA	GRP02_TMS	GRP02	30001	0	0	(IDLE)

Implementing CICS Applications

```

TMS_ORA          GRP02_TMS   GRP02           30002           0           0 ( IDLE )
TMS_ORA          GRP02_TMS   GRP02           30003           0           0 ( IDLE )
ARTCNX          QCNX015     GRP01            15            0           0 ( IDLE )
ARTSTRN         QKIX110     GRP02            20            0           0 ( IDLE )
ARTTSQ          00012.00040 GRP02            40            0           0 ( IDLE )

```

> psc

```

Service Name Routine Name Prog Name Grp Name ID Machine # Done Status
-----
TMS          TMS          TMS_ORA   GRP02   30001   KIXR    0 AVAIL
TMS          TMS          TMS_ORA   GRP02   30002   KIXR    0 AVAIL
TMS          TMS          TMS_ORA   GRP02   30003   KIXR    0 AVAIL
authfail    cnxsvc      ARTCNX    GRP01    15     KIXR    0 AVAIL
CESF        cnxsvc      ARTCNX    GRP01    15     KIXR    0 AVAIL
CESN        cnxsvc      ARTCNX    GRP01    15     KIXR    0 AVAIL
CSGM        cnxsvc      ARTCNX    GRP01    15     KIXR    0 AVAIL
disconnect  cnxsvc      ARTCNX    GRP01    15     KIXR    0 AVAIL
connect     cnxsvc      ARTCNX    GRP01    15     KIXR    0 AVAIL
SA03        kixsvc      ARTSTRN   GRP02    20     KIXR    0 AVAIL
SA02        kixsvc      ARTSTRN   GRP02    20     KIXR    0 AVAIL
SA01        kixsvc      ARTSTRN   GRP02    20     KIXR    0 AVAIL
SA00        kixsvc      ARTSTRN   GRP02    20     KIXR    0 AVAIL
TSM00004_TSQ tsqsvc     ARTTSQ    GRP02    40     KIXR    0 AVAIL
TSM00003_TSQ tsqsvc     ARTTSQ    GRP02    40     KIXR    0 AVAIL
TSM00002_TSQ tsqsvc     ARTTSQ    GRP02    40     KIXR    0 AVAIL
TSM00001_TSQ tsqsvc     ARTTSQ    GRP02    40     KIXR    0 AVAIL
TSM00000_TSQ tsqsvc     ARTTSQ    GRP02    40     KIXR    0 AVAIL

```



```

TSQUEUE      tsqsvc      ARTTSQ      GRP02      40      KIXR      0 AVAIL

> quit
#

```

Managing TD Queue Intrapartitions

Presentation of the Mechanism on Source Platform

Transient Data Control

The CICS transient data control facility provides a generalized queuing facility. Data can be queued (stored) for subsequent internal or external processing. Selected data, specified in the application program can be routed to or from predefined symbolic transient data queues: either intrapartition or extrapartition.

Transient data queues are intrapartition if they are associated with a facility allocated to the CICS region and extrapartition if the data is directed to a destination that is external to the CICS region. Transient data queues must be defined and installed before the first reference by an application program.

You can:

- Write data to a transient data queue (`WRITEQ TD` command).
- Read data from a transient data queue (`READQ TD` command).
- Delete an intrapartition transient data queue (`DELETEQ TD` command).

Note: In this document we concentrate exclusively on intrapartition TD queues.

Intrapartition Transient Data Queues

Intrapartition refers to data on direct-access storage devices for use with one or more programs running as separate tasks. Data directed to or from these internal queues is referred to as intrapartition data; it must consist of variable-length records.

When data is written to the queue by a user task, the queue can be used subsequently as input data by other tasks within the CICS region. All access is sequential, governed by read and write

pointers. Once a record has been read, it cannot be read subsequently by another task. Intrapartition data may ultimately be transmitted upon request to the terminal or retrieved sequentially from the output dataset.

Typical uses of intrapartition data include:

- Message switching.
- Broadcasting.
- Database access.
- Routing of output to several terminals (for example, for order distribution).
- Queuing of data (for example, for assignment of order numbers or priority by arrival).
- Data collection (for example, for batched input from 2780 Data Transmission Terminals)

There are three types of intrapartition transient data queues:

Non-recoverable

Non-recoverable intrapartition transient data queues are recovered only on a warm start of CICS. If a unit of work (UOW) updates a non-recoverable intrapartition queue and subsequently backs out the updates, the updates made to the queue are not backed out.

Physically recoverable

Physically recoverable intrapartition transient data queues are recovered on warm and emergency restarts. If a UOW updates a physically recoverable intrapartition queue and subsequently backs out the updates, the updates made to the queue are not backed out.

Logically recoverable

Logically recoverable intrapartition transient data queues are recovered on warm and emergency restarts. If a UOW updates a logically recoverable intrapartition queue and subsequently backs out the changes it has made, the changes made to the queue are also backed out. On a warm or an emergency restart, the committed state of a logically recoverable intrapartition queue is recovered. In-flight UOWs are ignored.

Automatic Transaction Initiation (ATI)

For intrapartition queues, CICS provides the option of automatic transaction initiation (ATI).

A basis for ATI is established by the system programmer by specifying a non-zero trigger level for a particular intrapartition destination. When the number of entries (created by WRITEQ TD commands issued by one or more programs) in the queue reaches the specified trigger level, a transaction specified in the definition of the queue is automatically initiated. Control is passed to

a program that processes the data in the queue; the program must issue repetitive READQ TD commands to deplete the queue.

When the queue has been emptied, a new ATI cycle begins. That is, a new task is scheduled for initiation when the specified trigger level is again reached, whether or not execution of the earlier task has ended. The exact point at which a new ATI cycle begins depends on whether or not the queue is defined as logically recoverable. If the queue is defined with a RECOVSTATUS of No or Physical, the new ATI cycle begins when the queue is read to QZERO. But if the queue is defined with a recoverability attribute of Logical, the new ATI cycle begins only after the task terminates after having read the queue to QZERO.

If an automatically initiated task does not empty the queue, access to the queue is not inhibited. The task may be normally or abnormally ended before the queue is emptied (that is, before a QZERO condition occurs in response to a READQ TD command). If the contents of the queue are to be sent to a terminal, and the previous task completed normally, the fact that QZERO has not been reached means that trigger processing has not been reset and the same task is reinitiated. A subsequent WRITEQ TD command does not trigger a new task if trigger processing has not been reset.

Presentation of the Mechanism on Target Platform

Tuxedo /Q

Tuxedo /Q offers a robust and versatile queuing system with the same capabilities as TD queues and more.

Queues can be defined as recoverable or not, and triggering with a few different options is also available. The management of errors is much more sophisticated, and will simplify error management in case of ATI transaction failures on target.

Architecture Design

Table 4-5 Source to Target Mapping

Source element	Target correspondence
TD Queue intrapartition	Tuxedo /Q Queue.
Associated transaction (TRANID)	Associated transaction offered by an ATR server.

Table 4-5 Source to Target Mapping

Source element	Target correspondence
Trigger level	Trigger level.
Recoverability: No Physical Logical	Same levels available as on target, but with different configuration principles.

The CICS verbs READQ TD, WRITEQ TD and DELETEQ TD (applied to intrapartition queues), now read, write or delete from a Tuxedo /Q queue. (tpenqueue and tpdequeue) in terms of tuxedo vocabulary.

If the Queue is logically recoverable, these actions are done in the current UOW, else they are done atomically, independently of the current UOW.

For information, inside CICS Runtime, this is done by adding the TPNOTRAN flag to operations on non-logically recoverable queues.

Triggering

In case of triggering, like in native CICS, a transaction will be automatically triggered, this transaction having to read the corresponding queue and process accordingly the messages.

In CICS Runtime these asynchronous transactions are offered and processed by a dedicated server type ARTATR, with either of its two variants ARTATR1 and ARTATR2.

These servers process all asynchronous transactions, more precisely, transactions submitted by START TRANSID, or by automatic Transaction Invocation related to td queue intrapartition.

In this case a specific CICS Runtime client, TDI_TRIGGER, is used to launch the corresponding asynchronous transaction, when the trigger level is reached.

Runtime CICS Configuration of TD Queue Intrapartition

CICS Runtime Resource Declaration

Every CICS-like resource in CICS Runtime, is declared using a dedicated configuration file stored in directory `$(KIXCONFIG)`.

TD Queue extrapartition and TD Queue intrapartition resource declaration share very few arguments, and are semantically very different objects, even if using the same API for read and write operations.

This is the reason why, in CICS Runtime, we have separated TD Queue extrapartition resource configuration and TD Queue intrapartition resource configuration into two different resource files.

Intrapartition queues are declared in the file `tdqintra.desc`, described in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.

The important attributes are:

TDQUEUE(name)

The queue name, exactly identical to the queue name in the source configuration, This name must be the same as the name of the queue in the Tuxedo /Q configuration.

RECOVSTATUS(status)

Only the status NO or LOGICAL, are accepted, the difference between the two modes impacts the treatment of WRITEQ TD and READQ TD, more precisely LOGICAL making them part of the current UOW, while NO makes them atomic operations independent of the current UOW.

The difference between NO or PHYSICAL, is not defined in the resource configuration file but will be implemented using native tuxedo /Q configuration parameters, mapping to persistent /Q or non persistent.

TRANSID and TRIGGERLEVEL

In the current release are documentary only in `tdqintra.desc`, it is their value in /Q configuration which is taken in account.

QSPACENAME

New argument needed for /Q: defining into which QSPACE the current /Q is stored. This argument is mandatory and must match the QSPACENAME into which the actual /Q queue is physically stored.

/Q Configuration for TD Queue Intrapartition in CICS Runtime

For detailed and accurate information on `qmadmin` and [/Q configuration Using the ATMI /Q Component](#) in the Tuxedo documentation.

The script `mk_td_qm_config.sh` distributed with CICS Runtime provides an example of qspace creation and then of queue creation and configuration into /Q, to be used for TD intrapartition queues.

This script uses three environment variables, which must be set according to your environment:

- `KIX_TD_QSPACE_DEVICE`: must contain the filename of the physical file containing the /Q database for TD queues.
- `KIX_TD_QSPACE_NAME`: contains the name of the logical QSPACE to create, which will contain the queues.
- `KIX_TD_QSPACE_IPCKEY`: a specific key which must be unique on the machine for the IPC used by the instance of /Q.

The creation of the device (`KIX_TD_QSPACE_DEVICE`) and of the QSPACE are very standard, we will not detail them.

The interesting part is related to queue configuration.

A `qopen QspaceName` command, to open the qspace which will contain the queues must be made before the creation of any queue. The `QspaceName` must match the `QSPACENAME` in the resource declaration of these queue(s).

Below is an example of an interactive queue creation using `qmadmin`, where the questions asked by `qmadmin` are in normal font, while the entries typed in by the user are in bold.

Listing 4-23 `qopen` Dialog

```
qopen TD_QSPACE
qcreate
Queue name: TEST
Queue order (priority, time, expiration, fifo, lifo): fifo
Out-of-ordering enqueueing (top, msgid, [default=none]): none
Retries [default=0]: 5
Retry delay in seconds [default=0]: 0
High limit for queue capacity warning (b for bytes used, B for blocks used,
% for percent used, m for messages [default=100%]): 5m
Reset (low) limit for queue capacity warning [default=0%]: 0m
Queue capacity command: "TDI_TRIGGER -t S049"
```

In a script an exact equivalent to this manual entry would be:

Listing 4-24 qopen Script

```
qopen TD_QSPACE
qcreate TEST fifo none 3 0 5m 0m "TDI_TRIGGER -t S049"
```

qopen Parameters

TD_QSPACE

The `QspaceName` must match the `QSPACENAME` in the resource declaration of these queue(s).

Queue name

The name of the queue must match exactly the name provided in the resource declaration.

Queue order

The default dequeuing order when reading the queue, the setting corresponding to TD intra native behavior is: `fifo`.

Out-of-ordering enqueueing

Not meaningful unless some application is using native /Q interface to write into these queue; for Runtime CICS only usage to set it to is: `none`

Retries

Defines the number of times a message will be put back on the queue in case of abort of the UOW having read this queue, to avoid resubmitting again and again an ATI transaction which fails because of a bad message, set this number to a reasonable number.

When this number is reached, or at the first abort if you set it to zero, the message will be removed from this queue and put onto the error queue for further analysis.

Retry delay in seconds

If retries is not null, defines a delay before putting a record back on its queue, in case of rollback, the recommended value with Runtime CICS is the default value 0.

High limit for queue capacity warning

This is the much more flexible equivalent of the trigger level of TD queues. For a setting compatible with TD queues, set it to the trigger level and express it in number of

messages. For example: 0m to suspend triggering, or 5m for a trigger level of 5 messages in the queue.

Reset (low) limit for queue capacity warning

This is the down level to be reached before resetting the trigger for the upper limit, for compatibility with TD queue behavior, it should be set to 0, (QZERO) which is the reset value for TD queues in CICS.

Queue capacity command:

This is the command to be launched when the trigger level is reached, in CICS Runtime it should be set to: `TDI_TRIGGER -t TRID`. Where TRID is the Transaction identifier of the transaction to trigger which should match the TRANSID of the resource configuration.

Tip: ATR servers when processing an ATI, know whether the transaction reached QZERO or not, and whether it was a success or a rollback. So if QZERO is not reached, they resubmit the transaction in the same manner as on the source platform. But now it is the number of retries which will replace the ATIFACILITY parameter and will govern the fact that a rolledback TD queue record will be resubmitted or not. It is a progress compared with the source is that now the administrator can decide the number of resubmissions, and get the faulty messages on an error queue.

Implementing Distributed Program Link (DPL)

For several reasons, on z/OS, the Distributed Program Link function enables a local CICS program (the client program) to call another CICS program (the server program) in a remote CICS region via `EXEC CICS LINK` statements. CICS Runtime supports this feature used in multi-CICS architecture like MRO.

To Detect That DPL Is Needed

Unless you wish to use the DPL in a UNIX written application, check the technical specificities of the z/OS application

1. Check on z/OS, using the CEDA system transaction, if at least one remote program is defined in the z/OS CICS CSD file. Such programs have some of their fields of the `REMOTE ATTRIBUTES` section filed:

Listing 4-25 Checking for Remote Programs

```

DEF PROGR

OVERTYPE TO MODIFY                                CICS RELEASE = 0610

CEDA DEFine PROGRam(                               )

  PROGRam      ==>
  Group        ==>
  DDescription ==>

. . . .

REMOTE ATTRIBUTES

DYNAMIC      ==> NO                               No ! Yes

REMOTESYSTEM ==> XXXX

REMOTENAME   ==> YYYYYYYY

TRANSID     ==> ZZZZ

EXECUTIONSET ==> Dplsubset                       Fullapi ! Dplsubset

```

Where (CICS default values are underlined):

DYNAMIC(YES|NO)

The following parameters cannot be overridden in the CICS LINK API. This field is only relevant for DPL use when it is set to NO and the three following fields are empty.

REMOTESYSTEM(name)

Remote CICS region name. An empty field is not relevant with `DYNAMIC(YES)`

REMOTENAME(name)

Remote server program name. An empty field is not relevant with `DYNAMIC(YES)` because the default is the client program name (`PROGRam ==>`).

TRANSID(name)

Remote mirror transaction. An empty field is not relevant with `DYNAMIC(YES)` because the default is the mirror system transaction CSMI.

EXECUTIONSET(FULLAPI|DPLSUBSET)

The DPL cannot use the full CICS API but only a subset. The DPLSUBSET parameter indicates explicit usage of a DPL subset of the CICS API, but note that this subset may also be sufficient to execute LINK in a non-DPL context without errors. On the other hand, this field may contain FULLAPI in a DPL context but does not ensure that no "Invalid Request errors" will follow if non-DPL API are used.

As described above, in some cases, the Remote Attributes declaration may not exist or can be incomplete. The reason is that these fields establish only some of the default values, some of the previous parameters in bold in the example are not provided in the EXEC CICS LINK API.

2. Then check in the programs, inside the EXEC CICS LINK API:
 - If the names of the programs called in this order match the names of programs defined in the CSD with remote attributes partially or fully informed.
 - If these statement contain at least one of the optional remote parameters shown in italics in the following CICS LINK API (the others fields are not relevant for DPL).

Listing 4-26 CICS LINK API For DPL

```
EXEC CICS LINK PROGRAM(...)
    COMMAREA(...)
    LENGTH(...)
    DATALENGTH(...)
    RETCODE(...)
    SYSID(XXXX) : Remote CICS region name
    SYNCONRETURN : Used for remote CICS syncpoint or rollback
    TRANSID(XXXX) : Remote mirror transaction instead of the CSMI default
    INPUTMSG(...)
    INPUTMSGLEN(...)
END-EXEC
```

Modifying the Tuxedo ubbconfig File to Manage the DPL

If at least one of your programs use the DPL, install and activate the ARTDPL server without changing your other settings.

To activate this server, modify your ubbconfig file to add this server to the *SERVERS section of the Tuxedo ubbconfig file. This server belongs to the same Server Group as the Transactions Servers (ARTSTRN, ARTSTR1, ARTATRN, ARTATR1).

Listing 4-27 ubbconfig File Example of a *SERVERS Section Describing the ARTDPL Server.

```
*SERVERS
...
ARTDPL      SRVGRP=GRP02
            SRVID=500
            CONV=N
            MIN=1 MAX=1 RQADDR=QKIXDPL REPLYQ=Y
            CLOPT=" -o /home2/work9/demo/Logs/TUX/sysout/stdout_dpl -e
/home2/work9/demo/Logs/TUX/sysout/stderr_dpl -r -- -s KIXD -l SIMPAPP"
...
```

Where:

***SERVERS**

Tuxedo ubbconfig Keyword indicating a Server Section definition.

SRVGRP

Is the Tuxedo Group Name to which ARTSTRN belongs.

SRVID

Is the identifier of a Tuxedo Server of ARTSTRN.

CONV=N

Indicates that this server operates in a non-conversational mode.

MIN=1 and MAX=1

Indicates that only one instance of this server must be run.

REPLYQ=Y

Indicates that this server will respond.

RQADDR=QKIXDPL

Name of the Tuxedo queue used for the responses.

CLOPT

Is a quoted text string passed to the server containing its parameters:

- o
Indicates the following file is used for the standard output messages of the server.
- e
Indicates the following file is used for the error output messages of the server.
- r
Is a Tuxedo parameter used to provide statistical reports.
- s KIXD
Indicates the CICS Runtime name where the KIXD transaction is run.
- l SIMAPP
Indicates that only the components of the SIMPDPL group are to be selected at start up.

Use the Tuxedo `tmadmin psr` and `psc` commands to check that this server is running and that no new service is published:

Listing 4-28 tmadmin Commands to Check ARTDPL Server

```
# tmadmin
tmadmin - Copyright (c) 2007-2010 Oracle.
Portions * Copyright 1986-1997 RSA Data Security, Inc.
All Rights Reserved.
Distributed under license by Oracle.
Tuxedo is a registered trademark.
```

Implementing Distributed Program Link (DPL)

> psr

Prog Name	Queue Name	Grp Name	ID	RqDone	Load	Done	Current	Service
-----	-----	-----	--	-----	-----	-----	-----	-----
ARTDPL	QKIXDPL	GRP02	500	0		0	(IDLE)	
ARTATR1	00012.00300	GRP02	300	0		0	(IDLE)	
ARTSTR1	00012.00200	GRP02	200	0		0	(IDLE)	
BBL	200933	KIXR	0	5		250	(IDLE)	
TMS_QM	GQUEUE_TMS	GQUEUE	30001	0		0	(IDLE)	
TMS_ORA	GRP02_TMS	GRP02	30001	0		0	(IDLE)	
ARTTCPL	00001.00101	TCP00	101	0		0	(IDLE)	
TMS_QM	GQUEUE_TMS	GQUEUE	30002	0		0	(IDLE)	
TMS_ORA	GRP02_TMS	GRP02	30002	0		0	(IDLE)	
TMS_ORA	GRP02_TMS	GRP02	30003	0		0	(IDLE)	
TMQUEUE	01000.01010	GQUEUE	1010	0		0	(IDLE)	
ARTCNX	QCNX015	GRP01	15	0		0	(IDLE)	
TMQFORWARD	01000.01020	GQUEUE	1020	0		0	(IDLE)	
ARTSTRN	QKIX110	GRP02	20	0		0	(IDLE)	
ARTATRN	QKIXATR	GRP02	30	0		0	(IDLE)	
ARTTSQ	00012.00040	GRP02	40	0		0	(IDLE)	

> psc

Service Name	Routine Name	Prog Name	Grp Name	ID	Machine	# Done	Status
-----	-----	-----	-----	--	-----	-----	-----
TMS	TMS	TMS_QM	GQUEUE	30001	KIXR	0	AVAIL
TMS	TMS	TMS_ORA	GRP02	30001	KIXR	0	AVAIL
TMS	TMS	TMS_QM	GQUEUE	30002	KIXR	0	AVAIL
TMS	TMS	TMS_ORA	GRP02	30002	KIXR	0	AVAIL

Implementing CICS Applications

TMS	TMS	TMS_ORA	GRP02	30003	KIXR	0 AVAIL
ASYNC_QSPACE	TMQUEUE	TMQUEUE	GQUEUE	1010	KIXR	0 AVAIL
authfail	cnxsvc	ARTCNX	GRP01	15	KIXR	0 AVAIL
CESF	cnxsvc	ARTCNX	GRP01	15	KIXR	0 AVAIL
CESN	cnxsvc	ARTCNX	GRP01	15	KIXR	0 AVAIL
CSGM	cnxsvc	ARTCNX	GRP01	15	KIXR	0 AVAIL
disconnect	cnxsvc	ARTCNX	GRP01	15	KIXR	0 AVAIL
connect	cnxsvc	ARTCNX	GRP01	15	KIXR	0 AVAIL
SA03	kixsvc	ARTSTRN	GRP02	20	KIXR	0 AVAIL
SA02	kixsvc	ARTSTRN	GRP02	20	KIXR	0 AVAIL
SA01	kixsvc	ARTSTRN	GRP02	20	KIXR	0 AVAIL
SA00	kixsvc	ARTSTRN	GRP02	20	KIXR	0 AVAIL
ASYNC_QUEUE	ASYNC_QUEUE	ARTATR	GRP02	30	KIXR	0 AVAIL
ASYNC_SA03	atrsvc	ARTATR	GRP02	30	KIXR	0 AVAIL
ASYNC_SA02	atrsvc	ARTATR	GRP02	30	KIXR	0 AVAIL
ASYNC_SA01	atrsvc	ARTATR	GRP02	30	KIXR	0 AVAIL
ASYNC_SA00	atrsvc	ARTATR	GRP02	30	KIXR	0 AVAIL
TSQUEUE	tsqsvc	ARTTSQ	GRP02	40	KIXR	0 AVAIL

> quit

#

Declaring Remote Programs in CICS Runtime

To allow an application to use distributed programs called in EXEC CICS LINK statements, these programs must be declared to CICS Runtime.

1. To declare REMOTE programs which can only use the DPL Subset of the CICS API:

- In the `programs.desc` file, set `REMOTESYSTEM` (the 7th field of the csv format dataset), to remote `SYSID` name (`KIXD` in sample of [Listing 4-27](#)).

The default is `local` (empty field), meaning that local programs are declared because they can use the FULL CICS API.

In our Simple Application example, if we suppose that `RSSAT000`, `RSSAT001` are remote and `RSSAT002` and `RSSAT003` are local, then the `programs.desc` file is set to:

Listing 4-29 Simple Application `programs.desc` Configuration of Remote Programs

```
#PROGRAM;GROUP;DESCRIPTION;LANGUAGE;EXECKEY;STATUS;REMOTESYSTEM;REMOTENAME
RSSAT000;SIMPAPP;Home Menu Program of Simple Application;COBOL;
;ENABLE;KIXD

RSSAT001;SIMPAPP;Customer Detailed Inf Program of Simple Application;COBOL;
;ENABLE;KIXD

RSSAT002;SIMPAPP;Customer Maintenance Program of the Simple
Application;COBOL; ;ENABLE

RSSAT003;SIMPAPP;Customer List of the Simple Application;COBOL; ;ENABLE
```

2. Shutdown and reboot Tuxedo.
3. Using the Tuxedo `tmadmin psr` and `psc` commands, check that new services for DPL programs are published and managed by `ARTDPL: KIXD_RSSAT0001` and `KIXD_RSSAT0003`.

Note: To avoid problems with homonyms, these distributed services have their names composed of the Tuxedo `DOMAINID` defined in the `ubbconfig` and the name of the program they manage.

Listing 4-30 Using `tmadmin` Commands to Check DPL Services

```
{deimos:work9}-/home2/work9/demo/Logs/TUX/sysout# tmadmin
tmadmin - Copyright (c) 2007-2010 Oracle.
Portions * Copyright 1986-1997 RSA Data Security, Inc.
All Rights Reserved.
```

Implementing CICS Applications

Distributed under license by Oracle.

Tuxedo is a registered trademark.

> psr

Prog Name	Queue Name	Grp Name	ID	RqDone	Load Done	Current	Service
-----	-----	-----	--	-----	-----	-----	-----
ARTDPL	QKIXDPL	GRP02	500	0	0	(IDLE)	
ARTATR1	00012.00300	GRP02	300	0	0	(IDLE)	
ARTSTR1	00012.00200	GRP02	200	0	0	(IDLE)	
BBL	200933	KIXR	0	5	250	(IDLE)	
TMS_QM	GQUEUE_TMS	GQUEUE	30001	0	0	(IDLE)	
TMS_ORA	GRP02_TMS	GRP02	30001	0	0	(IDLE)	
ARTTCPL	00001.00101	TCP00	101	0	0	(IDLE)	
TMS_QM	GQUEUE_TMS	GQUEUE	30002	0	0	(IDLE)	
TMS_ORA	GRP02_TMS	GRP02	30002	0	0	(IDLE)	
TMS_ORA	GRP02_TMS	GRP02	30003	0	0	(IDLE)	
TMQUEUE	01000.01010	GQUEUE	1010	0	0	(IDLE)	
ARTCNX	QCNX015	GRP01	15	0	0	(IDLE)	
TMQFORWARD	01000.01020	GQUEUE	1020	0	0	(IDLE)	
ARTSTRN	QKIX110	GRP02	20	0	0	(IDLE)	
ARTATR1	QKIXATR	GRP02	30	0	0	(IDLE)	
ARTTSQ	00012.00040	GRP02	40	0	0	(IDLE)	

> psc

Service Name	Routine Name	Prog Name	Grp Name	ID	Machine	# Done	Status
-----	-----	-----	-----	--	-----	-----	-----
KIXD_RSSAT0+	dplsvc	ARTDPL	GRP02	500	KIXR	0	AVAIL

Implementing Distributed Program Link (DPL)

KIXD_RSSAT0+	dplsvc	ARTDPL	GRP02	500	KIXR	0 AVAIL
TMS	TMS	TMS_QM	GQUEUE	30001	KIXR	0 AVAIL
TMS	TMS	TMS_ORA	GRP02	30001	KIXR	0 AVAIL
TMS	TMS	TMS_QM	GQUEUE	30002	KIXR	0 AVAIL
TMS	TMS	TMS_ORA	GRP02	30002	KIXR	0 AVAIL
TMS	TMS	TMS_ORA	GRP02	30003	KIXR	0 AVAIL
ASYNQ_QSPACE	TMQUEUE	TMQUEUE	GQUEUE	1010	KIXR	0 AVAIL
authfail	cnxsvc	ARTCNX	GRP01	15	KIXR	0 AVAIL
CESF	cnxsvc	ARTCNX	GRP01	15	KIXR	0 AVAIL
CESN	cnxsvc	ARTCNX	GRP01	15	KIXR	0 AVAIL
CSGM	cnxsvc	ARTCNX	GRP01	15	KIXR	0 AVAIL
disconnect	cnxsvc	ARTCNX	GRP01	15	KIXR	0 AVAIL
connect	cnxsvc	ARTCNX	GRP01	15	KIXR	0 AVAIL
SA03	kixsvc	ARTSTRN	GRP02	20	KIXR	0 AVAIL
SA01	kixsvc	ARTSTRN	GRP02	20	KIXR	0 AVAIL
SA00	kixsvc	ARTSTRN	GRP02	20	KIXR	0 AVAIL
ASYNQ_QUEUE	ASYNQ_QUEUE	ARTATR	GRP02	30	KIXR	0 AVAIL
ASYNQ_SA03	atrsvc	ARTATR	GRP02	30	KIXR	0 AVAIL
ASYNQ_SA01	atrsvc	ARTATR	GRP02	30	KIXR	0 AVAIL
ASYNQ_SA00	atrsvc	ARTATR	GRP02	30	KIXR	0 AVAIL
TSM00004_TSQ	tsqsvc	ARTTSQ	GRP02	40	KIXR	0 AVAIL
TSM00003_TSQ	tsqsvc	ARTTSQ	GRP02	40	KIXR	0 AVAIL
TSM00002_TSQ	tsqsvc	ARTTSQ	GRP02	40	KIXR	0 AVAIL
TSM00001_TSQ	tsqsvc	ARTTSQ	GRP02	40	KIXR	0 AVAIL
TSM00000_TSQ	tsqsvc	ARTTSQ	GRP02	40	KIXR	0 AVAIL
TSQUEUE	tsqsvc	ARTTSQ	GRP02	40	KIXR	0 AVAIL

```
> quit  
# .
```

To see full details on the truncated values displayed, you can use the Tuxedo verbose command.

To reduce the scope of the services listed to only those managed by ARTDPL (SRVID=500), use the Tuxedo psc command followed with the `-i srvid` parameter to restrict the display to a particular server id.

In our example, the srvid of the ARTDPL server is 500 as displayed just above.

Listing 4-31 Using tadmin Commands to Check Specific DPL Service in Verbose Mode

```
# tadmin  
tadmin - Copyright (c) 2007-2010 Oracle.  
Portions * Copyright 1986-1997 RSA Data Security, Inc.  
All Rights Reserved.  
Distributed under license by Oracle.  
Tuxedo is a registered trademark.  
  
> verbose  
Verbose now on.  
  
> psc -i 500  
Service Name: KIXD_RSSAT003  
Service Type: USER  
Routine Name: dplsvc  
    Prog Name: /home2/work9/KIXEDO/bin/ARTDPL  
Queue Name: QKIXDPL  
Process ID: 1327244, Machine ID: KIXR
```

```
      Group ID: GRP02, Server ID: 500
      Current Load: 50
      Current Priority: 50
      Current Trantime: 30
      Current Blocktime: 0
      Current BUFTYPECONV: 0
      Requests Done: 0
      Current status: AVAILABLE

      Service Name: KIXD_RSSAT001
      Service Type: USER
      Routine Name: dplsvc
      Prog Name: /home2/work9/KIXEDO/bin/ARTDPL
      Queue Name: QKIXDPL
      Process ID: 1327244, Machine ID: KIXR
      Group ID: GRP02, Server ID: 500
      Current Load: 50
      Current Priority: 50
      Current Trantime: 30
      Current Blocktime: 0
      Current BUFTYPECONV: 0
      Requests Done: 0
      Current status: AVAILABLE

> quit
#
```

Implementing CICS Common Work Area (CWA)

On z/OS, the CWA is a common storage area defined in memory for a CICS region that programs can use to save and exchange data between themselves as long as this CICS region is running.

This area is addressed thru a pointer delivered by the CICS statement `EXEC CICS ADDRESS CWA`. If you find this CICS statement in your application, you have to implement this feature in CICS Runtime.

Listing 4-32 COBOL Example of CWA Usage

```
LINKAGE SECTION.  
01  COMMON-WORK-AREA.  
    03  APPL-1-ID          PIC X(4).  
    03  APPL-1-PTR        USAGE IS POINTER.  
    03  APPL-2-ID          PIC X(4).  
    03  APPL-2-PTR        USAGE IS POINTER.  
  
PROCEDURE DIVISION.  
  
    . . .  
  
    END-EXEC.  
  
* Set up addressability to the CWA  
    EXEC CICS ADDRESS  
        CWA(ADDRESS OF COMMON-WORK-AREA)  
  
    END-EXEC.
```

After the `CICS ADDRESS CWA`, the address of the COBOL group named `COMMON-WORK-AREA` is set to the address of the CWA allocated by CICS, meaning that `COMMON-WORK-AREA` maps and refines this memory area. The total amount of this shared memory is fixed and defined at CICS start up.

To Replicate CICS ADDRESS CWA Functionality in CICS Runtime

1. Contact your z/OS CICS Administrator to know the size of memory implemented. (For your information this value is defined with the parameter `WRKAREA` of the `DFHSP`. The default value is 512 bytes and the size can vary from 0 to 3584 bytes). Another way is to calculate the biggest size of the data record contained in the programs addressing the CWA.
2. Modify your `~/profile` UNIX system file to export a new CICS Runtime variable, `KIX_CWA_SIZE`, and set it to the value found in the `WRKAREA` of the `DFHSP`. If this variable is not declared, note that the default value is 0 and the authorized interval from 0 to 32760 bytes.

Example:

```
KIX_CWA_SIZE=512
```

3. Modify your `~/profile` UNIX system file to export a new CICS Runtime variable, `KIX_CWA_IPCKEY`, and valorize it to a Unix IPC key to define the cross memory segment used as CWA.

Example:

```
KIX_CWA_IPCKEY=200944
```

4. Restart Tuxedo to take all these changes into account.

Implementing a CICS Transaction Work Area (TWA)

On z/OS, the TWA is a common storage area defined in memory for a CICS region that programs can use to save and exchange data between themselves during the execution time of one CICS transaction. In other words, this TWA can only be accessed by the programs participating in the transaction. This area is addressed thru a pointer delivered by the CICS statement `EXEC CICS ADDRESS TWA`. If you find an `EXEC CICS ADDRESS TWA` statement in your application, you have to implement this feature in CICS Runtime.

Listing 4-33 A COBOL Example of Use of the CWA

```
LINKAGE SECTION.
01 TRANSACTION-WORK-AREA.
   03 APPL-1-ID           PIC X(4).
   03 APPL-1-PTR         USAGE IS POINTER.
   03 APPL-2-ID           PIC X(4).
```

Implementing CICS Applications

```
03  APPL-2-PTR          USAGE IS POINTER.

PROCEDURE DIVISION.

. . .

END-EXEC.

* Set up addressability to the TWA

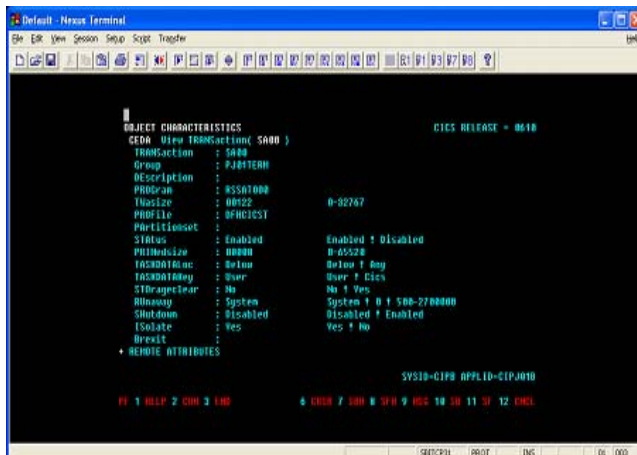
EXEC CICS ADDRESS

          TWA (ADDRESS OF TRANSACTION-WORK-AREA)

END-EXEC.
```

After the CICS ADDRESS TWA, the address of the COBOL group named TRANSACTION-WORK-AREA is set to the address of the TWA allocated by CICS, meaning that TRANSACTION -WORK-AREA maps and refines this memory area. The total amount of this shared memory is defined for each transaction in the z/OS CSD configuration file in the field TWAsize. The next screen shows the result of a z/OS CEDA system transaction where the TWAsize parameter is set to 122 for the SA00 transaction code:

Figure 4-3 z/OS ceda System Transaction Example



To replicate this functionality in CICS Runtime:

1. Modify the CICS Runtime `transactions.desc` file to report the needed amount of TWA memory (`TWAsize>0`).
2. For each transaction using programs with `CICS ADDRESS TWA` statements, modify the `transactions.desc` file to declare its `TWAsize` in the sixteenth field of this csv format file.

Table 4-6 TWA Size Values Associated to Each Transaction Code of the Simple Application

Transaction	TWA Size
SA00	0
SA01	100
SA02	200
SA03	300

Listing 4-34 Configuration of TWA in the transactions.desc File

```
#Transaction;Group;Description;Program; ; ; ; ; ;Status; ; ; ;Tranclass
;TWA Size
SA00;SIMPAPP;pg for simpapp;RSSAT000; ; ; ; ; ;ENABLED
SA01;SIMPAPP;pg for simpapp;RSSAT001; ; ; ; ; ;ENABLED; ; ; ; ;100
SA02;SIMPAPP;pg for simpapp;RSSAT002; ; ; ; ; ;ENABLED; ; ; ; ;200
SA03;SIMPAPP;pg for simpapp;RSSAT003; ; ; ; ; ;ENABLED; ; ; ; ;300
```

Note: Nothing is indicated for the SA00 transaction that had a TWA size equal to zero.

3. Restart the CICS Runtime Tuxedo servers, the modifications can be seen in the different `stderr` files of the servers involved in the transaction management (`ARTSTRN`, `ARTSTR1`, `ARTATRN` and `ARTATR1`)

Listing 4-35 stderr_strn TWA Example

```
|-----|
| TRANSACTIONS loaded : < 4> |
```

```

|-----|-----|-----|-----|-----| | | | |
|---|---|---|---|---|---|---|---|---|
|      |      |      |      |      |      |
|T|      |      |      |      |      |
|TRAN|  GROUP  |      |      |      |      |      |      |      |
|TASK|R|  TRAN  |  TWA |MAX|      |      |      |      |
|      |      |      |      |      |      |      |      |
|A|  CLASS  |  SIZ |ACT|      |      |      |      |
|      |      |      |      |      |      |      |      |
|C|      |      |  IVE|      |      |      |      |
|-----|-----|-----|-----|-----| | | | |
|---|---|---|---|---|---|---|---|---|
|SA00|SIMPAPP  |RSSAT000      |      |      |      |      |
|USER |Y|      |00000|999|      |      |      |      |
|SA01|SIMPAPP  |RSSAT001      |      |      |      |      |
|USER |Y|      |00100|999|      |      |      |      |
|SA02|SIMPAPP  |RSSAT002      |      |      |      |      |
|USER |Y|      |00200|999|      |      |      |      |
|SA03|SIMPAPP  |RSSAT003      |      |      |      |      |
|USER |Y|      |00300|999|      |      |      |      |

```

Implementing CICS Transaction Trigger Monitor (ARTCKTI)

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, retrieves trigger messages when a trigger event occurs, and then forwards the trigger messages to the target transaction.

Work Flow

ARTCKTI is a standalone Oracle Tuxedo server. The ARTCKTI server behaves as follows:

1. Monitor one or multiple WebSphere MQ initiation queues.

One server instance can only monitor WebSphere MQ initiation queues within the same WebSphere MQ queue manager. The queues in different WebSphere MQ queue managers should be monitored by separate ARTCKTI server instances.

2. When trigger message has arrived, the ARTCKTI server retrieves the message.
3. Retrieve the transaction ID from the trigger message.
4. Transfer the trigger message from structure MQTMC to MQTMC2.

Since MQTMC has many fields, it is always too complicated to send the structure as the parameter of EXEC CICS START call. MQTMC2 is used in CKTI to pass the structure as data to the START request for the trigger monitor.

5. Invoke the target transaction, and send the MQTMC2 data.

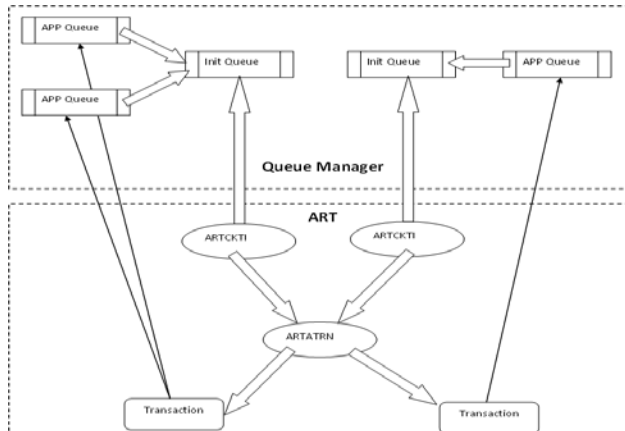
Since CICS CKTI transaction starts the target transaction with asynchronized call (EXEC CICS START), the ARTCKTI server also starts the target transaction with asynchronized call (Tuxedo tpcall).

6. User transaction retrieves the trigger message by CICS RETRIEVE, and performs operations on the WebSphere MQ application queue.

If the user transaction does not retrieve the message or the triggered transaction is not available, WebSphere MQ no longer sends trigger message in this condition. A new trigger message is issued until the WebSphere MQ initiation queue is reopened or a new trigger condition is met.

[Figure 4-4](#) illustrates the behavior.

Figure 4-4 WebSphere MQ Trigger Condition



Command Configuration

ARTCKTI 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 at the WebSphere MQ initiation queue.
- **-s retry_interval**: specifies the retry interval for ARTCKTI to reconnect to WebSphere MQ queue manager or reopen WebSphere MQ initiation queue upon failure.
- **-m queue_manager_name**: specifies the name of the WebSphere MQ queue manager to be monitored.
- **-q queue1,queue2,.....**: specifies the name of the WebSphere MQ initiation queue to be monitored.

CICS Runtime Logs

Tuxedo System Log

Like other Tuxedo applications, CICS Runtime is managed by Tuxedo that records certain events and problems in a dedicated system log.

This log is the standard Tuxedo User Log (ULOG) whose name is contained in the system variable `ULOGPFX` of the Tuxedo `ubbconfig` file.

Example:

```
ULOGPFX="/home2/work9/demo/Logs/TUX/log/ULOG"
```

This file is the only means to understand problems that may occur at the start up of CICS Runtime Tuxedo servers, just before they produce their own messages.

Once the CICS Runtime servers are running, they issue messages about their activity (warning, errors ...) to this log.

The ULOG log provides a good overview of the behavior and activity of CICS Runtime, as shown in the following extract:

Listing 4-36 ULOG Example

```
./ .metadata/.plugins/org.eclipse.ui.workbench/workbench.xml:<persistable
path="/trf/Logs/TUX/log/ULOG.120909"/>

./KIXEDO/dev/obj/tuxtrace.lst:          "@(#) VERSION: 1.0 Mar 22 2001:
ulog stderr stdout trace\".

./KIXEDO/dev/obj/tuxtrace.lst:          02 LigneFormateeUlog pic x(1000).

./KIXEDO/dev/obj/tuxtrace.lst:          call "sprintf" using
LigneFormateeUlog,

./KIXEDO/dev/obj/tuxtrace.lst:          LigneFormateeUlog(length
of LigneFormateeUlog:1)

./KIXEDO/dev/obj/tuxtrace.lst:          perform until
LigneFormateeUlog(nbr:1) = low-value

./KIXEDO/dev/obj/tuxtrace.lst:          inspect LigneFormateeUlog
converting x"0a" to "#"

./KIXEDO/dev/obj/tuxtrace.lst:          call "userlog" using pslw
LigneFormateeUlog

./KIXEDO/dev/obj/tuxtrace.lst:* 000607 LIGNEFORMATEEULOG . . . . .
00005967 00001000 WS E AlphNum G

./KIXEDO/dev/obj/tuxtrace.lst:* LIGNEFORMATEEULOG Alphnumeric
1000

./KIXEDO/dev/scripts/cleanlog:# Suppress TUXEDO ULOG
```

Implementing CICS Applications

```
./KIXEDO/dev/scripts/cleanlog:rm -f $TUXLOG/log/ULOG.*
./KIXEDO/dev/scripts/mkubbconfig.sh:  ULOGPFX="${TUXLOG}/log/ULOG"
./KIXEDO/dev/src/config.c:          userlog("undefined TP_USER_TRACE variable
default 'ULOG' assumed");
./KIXEDO/dev/src/config.c:          if ((strcmp(varEnv, "SID") == 0) ||
(strcmp(varEnv, "ULOG") == 0)) {
./KIXEDO/dev/src/config.c:          userlog("invalid USER_TRACE variable
value '%s' : 'ULOG' assumed", varEnv);
./KIXEDO/dev/src/tuxtrace.cbl:      "@(#) VERSION: 1.0 Mar 22 2001:
ulog stderr stdout trace\".
./KIXEDO/dev/src/tuxtrace.cbl:      02 LigneFormateeUlog  pic x(1000).
./KIXEDO/dev/src/tuxtrace.cbl:      call "sprintf" using
LigneFormateeUlog,
./KIXEDO/dev/src/tuxtrace.cbl:      LigneFormateeUlog(length
of LigneFormateeUlog:1)
./KIXEDO/dev/src/tuxtrace.cbl:      perform until
LigneFormateeUlog(nbr:1) = low-value
./KIXEDO/dev/src/tuxtrace.cbl:      inspect LigneFormateeUlog
converting x"0a" to "#"
./KIXEDO/dev/src/tuxtrace.cbl:      call "userlog" using pslw
LigneFormateeUlog
{deimos:work9}-/home2/work9#
{deimos:work9}-/home2/work9# find . -name ULOG*
./demo/Logs/TUX/log/ULOG.012210
./ULOG.012210
{deimos:work9}-/home2/work9# vi ./demo/Logs/TUX/log/ULOG.012210
{deimos:work9}-/home2/work9# vi ./demo/Logs/TUX/log/ULOG.012210
145532.deimos!ARTATRN.1134804.1.0: vsam file parameters : '001'
145532.deimos!ARTATR1.954446.1.0: 01-22-2010: Tuxedo Version 11.1.1.1.0,
64-bit
```

```
145532.deimos!ARTATR1.954446.1.0: LIBTUX_CAT:262: INFO: Standard main
starting
145532.deimos!ARTATR1.954446.1.0: undefined TP_USER_TRACE variable default
'ULOG' assumed
145532.deimos!ARTATR1.954446.1.0: undefined KIX_CWA_SIZE variable : Default
assumed (32768)
145532.deimos!ARTATR1.954446.1.0: undefined KIX_CBL_TRAP_ERROR - Default is
<Y>
145532.deimos!ARTATR1.954446.1.0: ConfigRead >> DONE
145533.deimos!ARTATR1.954446.1.0: vsam file parameters : '001'
145533.deimos!ARTTSQ.888940.1.0: 01-22-2010: Tuxedo Version 11.1.1.1.0,
64-bit
145533.deimos!ARTTSQ.888940.1.0: LIBTUX_CAT:262: INFO: Standard main
starting
145533.deimos!ARTTSQ.888940.1.0: undefined TP_USER_TRACE variable default
'ULOG' assumed
145533.deimos!ARTTSQ.888940.1.0: undefined KIX_CWA_SIZE variable : Default
assumed (32768)
145533.deimos!ARTTSQ.888940.1.0: undefined KIX_CBL_TRAP_ERROR - Default is
<Y>
145533.deimos!ARTTSQ.888940.1.0: ConfigRead >> DONE
145533.deimos!ARTDPL.872680.1.0: 01-22-2010: Tuxedo Version 11.1.1.1.0,
64-bit
145533.deimos!ARTDPL.872680.1.0: LIBTUX_CAT:262: INFO: Standard main
starting
145533.deimos!ARTDPL.872680.1.0: undefined TP_USER_TRACE variable default
'ULOG' assumed
145533.deimos!ARTDPL.872680.1.0: undefined KIX_CWA_SIZE variable : Default
assumed (32768)
145533.deimos!ARTDPL.872680.1.0: undefined KIX_CBL_TRAP_ERROR - Default is
<Y>
```

```
145533.deimos!ARTDPL.872680.1.0: ConfigRead >> DONE
145533.deimos!ARTDPL.872680.1.0: vsam file parameters : '001'
145533.deimos!TMS_QM.336010.1.0: 01-22-2010: Tuxedo Version 11.1.1.1.0,
64-bit
145533.deimos!TMS_QM.336010.1.0: LIBTUX_CAT:262: INFO: Standard main
starting
145534.deimos!TMS_QM.1831090.1.0: 01-22-2010: Tuxedo Version 11.1.1.1.0,
64-bit
145534.deimos!TMS_QM.1831090.1.0: LIBTUX_CAT:262: INFO: Standard main
starting
145534.deimos!TMQUEUE.1679402.1.0: 01-22-2010: Tuxedo Version 11.1.1.1.0,
64-bit
145534.deimos!TMQUEUE.1679402.1.0: LIBTUX_CAT:262: INFO: Standard main
starting
145534.deimos!TMQFORWARD.348390.1.0: 01-22-2010: Tuxedo Version 11.1.1.1.0,
64-bit
145534.deimos!TMQFORWARD.348390.1.0: LIBTUX_CAT:262: INFO: Standard main
starting
```

The CICS Runtime Server Logs

When declaring a service in the Tuxedo `ubbconfig` file, each server has CLOPT options defined including two files:

- `-o` option for stdout (normal messages)

The name of this file is `stdout_<server name>` without the ART prefix.

For example: the `ARTSTRN` server has a standard output named `stdout_strn`.

- `-e` option for stderr (error messages)

The name of this file is `stderr_<server name>` without the ART prefix.

For example: the `ARTSTRN` server has an error output named `stderr_strn`.

The different stdout and stderr message files for each CICS Runtime server are:

Table 4-7 Message Files by Server

Server name	-o standard output file	-e standard error file
ARTTCPL	stdout_tcp	stderr_tcp
ARTCNX	stdout_cnx	stderr_cnx
ARTSTRN	stdout_strn	stderr_strn
ARTSTR1	stdout_str1	stderr_str1
ARTATRn	stdout_atrn	stderr_atrn
ARTATR1	stdout_atr1	stderr_atr1
ARTTSQ	stdout_tsq	stderr_tsq
ARTDPL	stdout_dpl	stderr_dpl

Note: In the stderr file of a server all the configuration files mounted are described. The stderr file contains not only the error messages concerning problems encountered when the servers are booted but also information about the different resources loaded. Specifically you will find:

- The groups of resources installed depending on the `-l` list parameter of each CICS Runtime server.
- The resources successfully installed and available for use (remember that an installed resource may be disabled for use) depending on the valorization of each `.desc` configuration file.

Listing 4-37 Example of the stdout_strn Just After Start Up for a ARTSTRN Server

```
Groups loaded: <0001>
```

```
|-----|
|  GROUP  |
|-----|
|SIMPAPP  |
|-----|
```

Implementing CICS Applications

ARTSTRN: Read config done

```

|-----|
| TRANCLASS loaded : < 2> |
|-----|
|          TRANCLASS          |  GROUP  |MAXACTIVE|
|-----|-----|-----|
|TRCLASS1                      |SIMPAPP |    001 |
|TRCLASS2                      |SIMPAPP |    002 |
|-----|
|-----|
| PROGRAMS loaded : < 4> |
|-----|
|          PROGRAM          |  GROUP  |LANGUAGE|EXEC|  STATUS |
|          |                |          |    |    |        |
|-----|-----|-----|----|-----|
|RSSAT000                      |SIMPAPP |COBOL  |USER|ENABLED |
|RSSAT001                      |SIMPAPP |COBOL  |USER|ENABLED |
|RSSAT002                      |SIMPAPP |COBOL  |USER|ENABLED |
|RSSAT003                      |SIMPAPP |COBOL  |USER|ENABLED |
|-----|
|-----|
| TRANSACTIONS loaded : < 4> |
|-----|-----|----|---|---|---|-----|
|-|-----|-|-----|-----|---|
|          |          |          |          |          |
|T|          |          |          |          |          |
|TRAN|  GROUP  |          PROGRAM          |ALIA|M|O|PRI|E|E|  STATUS
|TASK |R|  TRAN  | TWA |MAX|

```



```

|      |      |      |      |      |      |      |      |      |      | |
|A| CLASS | SIZ |ACT|      |      |      |      |      |      |      |
|      |      |      |      |      |      |      |      |      |      |
|C|      |      | IVE|      |      |      |      |      |      |
|-----|-----|-----|-----|-----|-----|-----|-----| | |
|---|---|---|---|---|---|---|---|---|---|
|SA00|SIMPAPP  |RSSAT000      |      |      |      |      |      |      |
|USER |Y|      |00000|999|      |      |      |      |      |
|SA01|SIMPAPP  |RSSAT001      |      |      |      |      |      |      |
|USER |Y|      |00000|999|      |      |      |      |      |
|SA02|SIMPAPP  |RSSAT002      |      |      |      |      |      |      |
|USER |Y|      |00000|999|      |      |      |      |      |
|SA03|SIMPAPP  |RSSAT003      |      |      |      |      |      |      |
|USER |Y|      |00000|999|      |      |      |      |      |
|-----|-----|-----|-----|-----|-----|-----|
-----|

```

Warning: zero TSQMODEL loaded!!

FILES<FILE> lineNo(1) skipping Record: Group not to load

FILES<FIC3> lineNo(4) skipping Record: Group not to load

We can note in this example that

- One group (SIMPAPP) is selected with the -1 option
- Four configurations files are used: transactions, tranclasses, programs and tsqmodels.
- Information on the successful loading of these resources (Warning: zero TSQMODEL loaded).
- The detail of the resources loaded and their explicit characteristics (name, group, description ...) even default/implicit values were used in the .desc file leaving the fields filed with space(s).

Disabling and Enabling Programs

Sometimes, problems are encountered in a program that significantly impacts your system and the program must be eliminated urgently by prohibiting end-users from running it. In the immediate, this helps temporarily to stabilize the system giving time to analyze and solve the dysfunction.

As on z/OS, CICS Runtime allows to disable a program. A program is disabled by modifying the CICS Runtime configuration file `programs.desc`. This file contains a dedicated field, the `STATUS` field, to indicate if a program is `DISABLED` or `ENABLED` (status by default).

To Disable Programs

To switch your transaction from enabled to disabled, you have to modify the seventh field of this csv file, to change the previous value from an implicit (" " space(s)) or an explicit `ENABLED` status to the explicit `DISABLED` status.

After shutting down and booting the CICS Runtime Tuxedo servers, your modifications of one or more programs will be taken in account.

If you disable a program, when somebody wants to use it, the error messages displayed depend on the way that the application handles CICS errors.

Listing 4-38 Example Simple Application SA02 COBOL Program Set to `DISABLED` in `programs.desc`

```
#PROGRAM;GROUP;DESCRIPTION;LANGUAGE; ; ;STATUS
RSSAT000;SIMPAPP; Home Menu Program of the Simple Application ;COBOL
RSSAT001;SIMPAPP; Customer Detailed Information Program of the Simple
Application ;COBOL; ; ;ENABLED;
RSSAT002;SIMPAPP; Customer Maintenance Program of the Simple
Application;COBOL; ; ;DISABLED;
RSSAT003;SIMPAPP; Customer List of the Simple Application ;COBOL
```

To Enable Programs

To enable a program, you have only to do the opposite, changing the STATUS field from DISABLED to ENABLED or " " (at least one space).

After shutting down and booting the CICS Runtime Tuxedo servers, your modifications of one or more programs will be taken in account.

Checking the Change in Program Status

If you consult the logs of the different transactions servers or the CICS Runtime you will note the modification of the modified status in the `stderr_*` logs.

Just after the start up of this server, the logs shows (in italics) that this program is disabled.

Listing 4-39 Log Report Showing Program Status

```

Groups loaded: <0001>

|-----|
|  GROUP  |
|-----|
|SIMPAPP  |
|-----|

ARTSTRN: Read config done

|-----|
|  TRANCLASS loaded : <  2>  |
|-----|
|          TRANCLASS          |  GROUP  |MAXACTIVE|
|-----|-----|-----|
|TRCLASS1                      |SIMPAPP |    001 |
|TRCLASS2                      |SIMPAPP |    002 |
|-----|-----|-----|
|-----|

```

Implementing CICS Applications

```

| PROGRAMS loaded : < 4>
|-----|
|          PROGRAM          |  GROUP  | LANGUAGE|EXEC|  STATUS  |
|                          |         |         |KEY |         |
|-----|-----|-----|----|-----|
|RSSAT000                  |SIMPAPP | COBOL  |USER|ENABLED  |
|RSSAT001                  |SIMPAPP | COBOL  |USER|ENABLED  |
|RSSAT002                  |SIMPAPP | COBOL  |USER|DISABLED |
|RSSAT003                  |SIMPAPP | COBOL  |USER|ENABLED  |
|-----|
|-----|
| TRANSACTIONS loaded : < 4>
|-----|-----|-----|----|-----| |
|---|---|---|---|---|---|
|          |          |          |          |          |
|T|          |          |          |          |          |
|TRAN|  GROUP  |          PROGRAM  |ALIA|M|O|PRI|E|E|  STATUS
|TASK |R|  TRAN  | TWA |MAX| | |
|          |          |          |          |          |
|A|  CLASS  | SIZ |ACT|          |          |          |
|          |          |          |          |          |
|C|          |          |IVE|          |          |          |
|-----|-----|-----|----|-----|
-|-----|-|-----|-----|----|
|SA00|SIMPAPP  |RSSAT000          |          |N|N|001|N|N|ENABLED
|USER |Y|          |00000|999|
|SA01|SIMPAPP  |RSSAT001          |          |N|N|001|N|N|ENABLED
|USER |Y|          |00000|999|
|SA02|SIMPAPP  |RSSAT002          |          |N|N|001|N|N|ENABLED
|USER |Y|          |00000|999|

```

```
|SA03|SIMPAPP   |RSSAT003           |      |N|N|001|N|N|ENABLED
|USER |Y|         |00000|999|
```

```
|-----|
|-----|
```

Warning: zero TSQMODEL loaded!!

Removing and Adding Applications for CICS Runtime

Sometimes, you want to delete an application from a given machine either to definitely delete all its components or to move them to another machine. If all the resources used by your application were defined in one or more resource groups dedicated to your application, you have only to suppress these groups from CICS Runtime and eventually install them elsewhere.

Each CICS Runtime Tuxedo Server reads a list of groups to be selected and installed at start up, contained in its CLOPT options after the `-l` parameter. To remove or add group(s) from an application, you have only to remove or add these groups from this list for each CICS Runtime Tuxedo server.

Listing 4-40 Example of Application in ARTSTRN Server

```
ARTSTRN   SRVGRP=GRP02
          SRVID=20
          CONV=Y
          MIN=1 MAX=1 RQADDR=QKIX110 REPLYQ=Y
          CLOPT=" -o /home2/work9/demo/Logs/TUX/sysout/stdout_strn
          -e /home2/work9/demo/Logs/TUX/sysout/stderr_strn -r -- -
          s KIXR -l SIMPAPP"
```

If you want to add one or more groups, you have to concatenate these new groups to those previously defined, separating them with a ":" character.

Listing 4-41 Example of Adding group1 and group2 in ARTSTRN Server

```
ARTSTRN    SRVGRP=GRP02

           SRVID=20

           CONV=Y

           MIN=1 MAX=1 RQADDR=QKIX110 REPLYQ=Y

           CLOPT="-o /home2/work9/demo/Logs/TUX/sysout/stdout_strn
           -e /home2/work9/demo/Logs/TUX/sysout/stderr_strn -r -- -
           s KIXR -1 SIMPAPP:GROUP1:GROUP2"
```

If you want to remove groups, you remove them from the -1 lists when they are present, leaving only one : character between the remaining groups.

Listing 4-42 Example of Removing group1 in ARTSTRN Server

```
ARTSTRN    SRVGRP=GRP02

           SRVID=20

           CONV=Y

           MIN=1 MAX=1 RQADDR=QKIX110 REPLYQ=Y

           CLOPT="-o /home2/work9/demo/Logs/TUX/sysout/stdout_strn
           -e /home2/work9/demo/Logs/TUX/sysout/stderr_strn -r -- -
           s KIXR -1 SIMPAPP:GROUP2"
```

Notes:

- When the groups are removed, all the resources of these groups are only logically suppressed. If you want also to suppress them physically, you have to delete all the lines of the CICS Runtime resource configuration files containing the group names.
- When the groups are added, all the resources of these groups must be present in the different CICS Runtime resource configuration files under the group names. To

avoid future problems, do not omit to declare resources in a group because they are already declared in groups from other applications.

- When groups are added or removed, be careful to indicate the same list of groups in each CICS Runtime server.

Implementing CICS Applications

Reference

Cross Reference of .desc Configuration Files Used by CICS Runtime Servers

The following table lists the configuration `.desc` files used per each CICS Runtime server. The value of 1 at the intersection of a server and a file means that they are linked.

Table 5-1 Resources Configuration ".desc" File

Servers	FILES	PROGRAMS	TRANCLASSES	TRANSACTIONS	TSQMODEL	Total
ARTATR1	1	1	1	1	1	5
ARTATRN	1	1	1	1	1	5
ARTDPL					1	1
ARTSTR1	1	1	1	1	1	5
ARTSTRN	1	1	1	1	1	5
ARTTSQ					1	1
Total	4	4	4	4	6	22

Reference

Oracle Tuxedo Application Runtime for CICS CSD Converter

This chapter contains the following topics:

- [Overview](#)
- [Resource Definition Online \(RDO\) Mapping](#)

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 currently defined in the CICS system definition file (CSD).

The `tcxcsdconv` tool (located in the `$KIXDIR/tools` directory), maps the CSD file to resource descriptive files (including transaction, transaction class, program, files, TS Queue, ENQ, TD Queue extra partition, TD Queue intra partition, mapset, and typeterm).

This tool is used to set the target CSD file in argument, and the translated resource configuration files resides in current directory by default. You can also specify other target directories to store the configuration files.

Resource Definition Online (RDO) Mapping

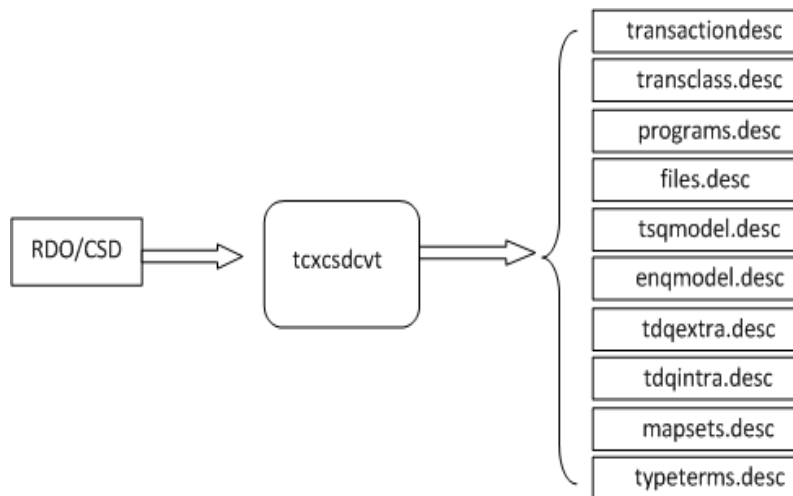
Resource Definition Online (RDO) Mapping consists of two parts:

1. Files conversion from RDO/CSD on z/OS to resource configuration files of all types on universal platform, such as transactions, programs, mapsets and etc.

2. For each type, tool `tcxcscvt` reads the value of all fields, and then generates a record in the corresponding resource configuration file. For more information, see "CICS Runtime Configuration Files" of CICSREF.

Figure 6-1 depicts the file data stream model.

Figure 6-1 File Data Stream Model



Tables 1~10 describe detailed correspondence between RDO/CSD and target resource configuration files, which have ".desc" as suffix mentioned above. These mappings include:

- [TRANCLASS Mapping](#)
- [PROGRAM Mapping](#)
- [FILE Mapping](#)
- [Journaling Attributes in FILE Mapping](#)
- [TSQUEUE MODEL Mapping](#)
- [ENQMODEL Mapping](#)
- [TDQUEUE Extra Partition Mapping](#)
- [TDQUEUE Intra Partition Mapping](#)
- [MAPSET Mapping](#)

- [TYPETERM Mapping](#)

Note: Since some field names are new options added in CICS Runtime Configuration Files, they are not defined or supported by RDO/CSD. To mark these attributes "----" is used.

Table 6-1 TRANCLASS Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
TRANCLASS	TRANCLASS	Name of the transaction class.
GROUP	GROUP	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	DESCRIPTION	A small textual comment zone for description of the resource.
MAXACTIVE	MAXACTIVE	Defines the degree of parallelism of execution.

Table 6-2 PROGRAM Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
PROGRAM	PROGRAM	Name of the program.
GROUP	GROUP	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	DESCRIPTION	A small textual comment zone for description of the resource.
LANGUAGE	LANGUAGE	The language of the program, required to know how to communicate with it.
EXECKEY	EXECKEY	Reserved for future use. Concerns memory protection of CICS shared structures.
STATUS	STATUS	Specifies the program status.

Table 6-2 PROGRAM Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
REMOTESYSTEM	REMOTESYSTEM	Specifies that the program is not offered locally but in a DPL server.
REMOTENAME	REMOTENAME	Specifies for a DPL program the name of the program on the distant site.

Table 6-3 FILE Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
FILE	FILE	Name of the file; logical name of the file used in EXEC CICS related to this file.
GROUP	GROUP	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	DESCRIPTION	A small textual comment zone for description of the resource.
DISPOSITION	DISPOSITION	Specifies the disposition of this file.
DSNAME	DSNAME	Specifies the data set name to be used for this file.
JOURNAL	JOURNAL	Specifies whether you want automatic journaling for this file.
KEYLENGTH	KEYLENGTH	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).
OPENTIME	OPENTIME	Specifies when the file is opened.
READINTEG	READINTEG	Specifies the level of read integrity required for files defined with RLSACCESS (YES).

Table 6-3 FILE Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
RECORDSIZE	RECORDSIZE	Specifies the maximum length in bytes of records in a remote file or a coupling facility data table.
REMOTENAME	REMOTENAME	Specifies the name of the file on the remote system.
REMOTESYSTEM	REMOTESYSTEM	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	STATUS	Specifies the initial status of the file following a CICS initialization.

Table 6-4 Journaling Attributes in FILE Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
JNLADD	JNLADD	Specifies if you want the add operations recorded on the journal nominated by the JOURNAL attribute.
JNLREAD	JNLREAD	Specifies the read operations you want recorded on the journal nominated by the JOURNAL attribute.
JNLSYNCREAD	JNLSYNCREAD	Specifies whether you want the automatic journaling records, written for READ operations to the journal, to be synchronous.

Table 6-4 Journaling Attributes in FILE Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
JNLSYNCWRITE	JNLSYNCWRITE	Specifies whether you want the automatic journaling records, written for WRITE operations to the journal, to be synchronous.
JNLUPDATE	JNLUPDATE	Specifies whether you want REWRITE and DELETE operations recorded on the journal.

Table 6-5 TSQUEUE MODEL Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
TSMODEL	TSMODEL	Name of the TS Queue model.
GROUP	GROUP	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	DESCRIPTION	A small textual zone for description of the resource.
LOCATION	LOCATION	Specifies the kind of storage to use: file or memory.
PREFIX XPREFIX	PREFIX XPREFIX	Specifies the character string that is to be used as the prefix for this model.
RECOVERY	RECOVERY	Specifies whether or not queues matching this model are to be recoverable.
POOLNAME	POOLNAME	Specifies the 8-character name of the shared TS pool definition that you want to use with this TSMODEL definition.

Table 6-5 TSQUEUE MODEL Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
REMOTE_SYSTEM	REMOTESYSTEM	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	REMOTEPREFIX XREMOTEPREFIX	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
SECURITY	SECURITY	Specifies whether security checking is to be performed for queues matching this model.

Table 6-6 ENQMODEL Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
ENQMODEL	ENQMODEL	Name of the ENQ model.
GROUP	GROUP	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	DESCRIPTION	A small textual zone for description of the resource.
ENQNAME	ENQNAME	Specifies the 1 to 255-character resource name.

Table 6-6 ENQMODEL Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
ENQSCOPE	ENQSCOPE	If omitted or specified as blanks, matching enqueue models will have a local scope, else they will have a global scope
STATUS	STATUS	E = Enabled; D = Disabled.

Table 6-7 TDQUEUE Extra Partition Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
TDQUEUE	TDQUEUE	Specifies the 1- to 4-character name of a transient data queue.
GROUP	GROUP	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	DESCRIPTION	A small textual zone for description of the resource.
DDNAME	DDNAME	Specifies a 1-to 8-character value that may refer to a data set defined in the startup JCL.
DISPOSITION	DISPOSITION	Specifies the disposition of the data set (MOD; OLD; SHR).
ERRORPTION	ERRORPTION	(UNSUPPORTED) Specifies the action to be taken if an I/O error occurs.
OPENTIME	OPENTIME	(UNSUPPORTED) Specifies the initial status of the data set.
RECORDFORMAT	RECORDFORMAT	Specifies the record format of the data set.
PRINTCONTROL	PRINTCONTROL	(UNSUPPORTED) Specifies the control characters to be used.

Table 6-7 TDQUEUE Extra Partition Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
RECORDSIZE	RECORDSIZE	Specifies the record length in bytes.
TYPEFILE	TYPEFILE	Specifies the type of data set the queue is to be associated with an input or output dataset.
DSNAME	DSNAME	Specifies the name of the file that is to be used to store records written to this extra partition queue.
SYSOUTCLASS	SYSOUTCLASS	(UNSUPPORTED) Specify the class of the SYSOUT data set.
TRT	----	Allow integrators and customers to make their own specific implementation of extra-partition queues.

Table 6-8 TDQUEUE Intra Partition Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
TDQUEUE	TDQUEUE	Specifies the 1- to 4-character name of a transient data queue.
GROUP	GROUP	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	DESCRIPTION	A small textual zone for description of the resource.
RECOVSTATUS	RECOVSTATUS	Specifies if the queue is logically recoverable or not.
TRANSID	TRANSID	Specifies the name of the transaction that is to be automatically initiated when the trigger level is reached.

Table 6-8 TDQUEUE Intra Partition Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
TRIGGERLEVEL	TRIGGERLEVEL	Specifies the number of records to be accumulated before a task is automatically initiated to process them.
USERID	USERID	Specifies the userid you want CICS to use for security checking when verifying the trigger-level transaction specified in the TRANSID field.
WAIT	WAIT	(INACTIVE field) Accepted only in the resource loading.
WAITACTION	WAITACTION	(INACTIVE field) Accepted only in the resource loading.
QSPACE	----	Specify the name of the tuxedo /Q QSPACE into which this queue is physically stored.
TRT	----	Allow integrators and customers to make their own specific implementation of intra-partition queues.

Table 6-9 MAPSET Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
NAME	MAPSET	Name of the MAPSET.
GROUP	GROUP	Installation group name.
DESCRIPTION	DESCRIPTION	A general description of the MAPSET resource.
RESIDENT	RESIDENT	YES=preload. NO=load on first use.
swastatus	STATUS	Sets the status of the resource, to specify if it is available.

Table 6-9 MAPSET Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
Usage	USAGE	Specifies the caching scheme to be used once the MAPSET is loaded.
FILENAME	----	Specifies the physical (binary) file name of the mapset, which is generated by the <code>tcxmapgen</code> utility (refer to section).

Table 6-10 TYPETERM Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
NAME	TYPETERM	Name of the <code>typeterm</code> .
GROUP	GROUP	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	DESCRIPTION	A small textual zone for description of the resource.
color	COLOR	Designates extended color attributes.
defscreencolumn	DEFSCREEN (rows, columns)	Number of columns of the default screen size.
defscreenrow	DEFSCREEN (rows, columns)	Number of rows of the default screen size.
hilight	HILIGHT	Indicates whether a terminal supports the highlight feature.
logonmsg	LOGONMSG	Indicates whether the "Good Morning" (CSGM) transaction is automatically started on the terminal.
outline	OUTLINE	Indicates whether the terminal supports field outlining.
swastatus	STATUS	Specifies the resource status (whether available).

Table 6-10 TYPETERM Mapping

Field Name in ART FOR CICS	Resource Attribute in RDO/CSD	Description
uctran	UCTRAN	Specify whether translate lowercase alphabetic characters to uppercase, or only translate the transaction ID from lowercase to uppercase, or not translate any characters.
userarealen	USERAREALEN	The terminal control table user area (TCTUA) area size for the terminal.
INTERCODE	----	Specifies which encoding type of inbound data is used.
EXTERCODE	----	Specifies which encoding type of outbound data is used.
SOSI	SOSI	Specifies whether mixed EBCDIC and double-byte character set (DBCS) is supported.
PROGSYMBOLS	PROGSYMBOLS	Specifies whether the programmed symbol (PS) facility is supported.

UDB Linking

Installation Time UDB Linking

The file `makefile_intg` is provided as an example to help you generate a runtime linked with UDB, you can adapt it for other uses (partial linking with UDB, linking with DB2 connect, etc.).

CICS Runtime servers can be linked with an Oracle database or a UDB (LUW) database using a makefile named `makefile_intg`. This makefile detects the database that you are using thru the following environment variables: `DB2DIR` or `ORACLE_HOME`.

- If `DB2DIR` exists, the makefile links the servers with UDB libraries instead of the Oracle libraries, binds or rebinds the impacted servers and builds the `TMS_UDB`.
- remark1 for ARTTSQ: ARTTSQ server uses a specific module for DB2 or Oracle access, so in the makefile there is not only the database library changes, but also the linking of `kix_tsrecov_UDB` or `kix_tsrecov_Oracle` to build the ARTTSQ server.
- Remark 2 for ARTTSQ, if you have recoverable TS Queues, and are using DB2 then you need to create the TS queue table for UBD. This should be done by using the script `crtstable_UDB` instead of the script `crtstable_Oracle`
- If `ORACLE_HOME` exists, the makefile links or relinks the servers with Oracle libraries and builds the `TMS_ORA`.
- If both variables (`DB2DIR` and `ORACLE_HOME`) exist or neither of these variables exist, the makefile stops and publishes a clear error message.

You can also change the `DATABASE` variable in the `makefile_intg` file to force the UDB or Oracle linking, as shown in the following example:

Listing A-1 Setting the Database Variable

```
#DATABASE=$( DATABASE_U )$( DATABASE_O )  
DATABASE=UDB
```

Rebuilding Servers for UDB

The servers delivered are built to be used with Oracle, to rebuild these servers for UDB:

1. Check that the `DB2DIR` variable is set .
2. Check that the `ORACLE_HOME` variable is **not** set (or change the makefile).
3. Open the `tools` directory.
4. Run `make -f makefile_intg all`

Note: For UDB linking, make sure that you have the following line in the Tuxedo RM file:

```
UDB_XA:db2xa_switch_std:-L${DB2DIR}/lib64 -ldb2
```


Rebuilding ART Servers for CICS

You need to rebuild the ART CICS servers in case of updates on one of the following components:

- Major OS version
- Tuxedo
- RDBMS: Oracle or UDB
- WebSphere MQ
- Cobol compiler: Micro Focus or Cobol IT
- C++ compiler

Rebuilding the ART CICS Servers

Use the provided tool “`makefile_intg`” and do the following:

1. Open the `tools` directory of your ART installation.
2. Check that the variables used in the `makefile_intg` are set in your environment.
3. Run `make -f makefile_intg all`

Rebuilding ART Servers for CICS

External CICS Interface (EXCI)

Overview

The external CICS interface is an application programming interface which enables a non-CICS program running in MVS to:

- Allocate and open sessions to a CICS system, and issue DPL requests on these sessions.
- Call a server program running in a CICS environment, pass and receive data by means of a communications area.

The external CICS interface provides two forms of programming interface:

- The `EXCI CALL` interface, which consists of six commands that allow you to:
 - Allocate and open sessions to a CICS system from non-CICS programs running under the MVS.
 - Issue DPL requests on these sessions from the non-CICS programs.
 - Close and release the sessions on completion of the DPL requests.
- The `EXEC CICS` interface, which provides a single composite command (`LINK PROGRAM() APPLID()`) that performs all six commands of the `EXCI CALL` interface in one invocation.

EXCI in Oracle Tuxedo Application Runtime

Figure C-1 EXCI in ART



Each EXCI ART CICS program must be defined as a DPL service in the `resource/program.desc` file. The seventh column must contain the CICS SYSID, and the service is advertised with the name: `<SYSID>_<PROGRAM>`.

If all DPL requests are done with `SYNCONRETURN` and not under the RRMS control, no Resource Manager is required in the Tuxedo Client. The initialization is done during the first EXCI request process and the Tuxedo session is terminated at the end of the client process.

If RRMS is used or one DPL request is done without `SYNCONRETURN`, the Tuxedo Client process must be built with one Resource Manager. At the initialization, the Resource Manager is opened and the transaction is begun at the beginning of the Client process. If one of these steps is not successful, the Client process aborts. At the normal end of the client process, the transaction is committed if it was not explicitly done by the client program (`RSSCMIT`). At the abnormal end of the client process, the transaction is rolled back. In each of these cases the Resource Manager is closed and the Tuxedo session is terminated.

Supported EXCI Interface

The EXCI precompiler option must be used for CICS client batch program.

The precompiler supports `EXCI CALL` or only one CICS command, `EXEC CICS LINK` with the next described options.

In case of `EXEC CICS LINK`, the `RETCODE` command option is mandatory with `EXCI` but forbidden with `NOEXCI`, and the `APPLID` option is `EXCI` specific. Without `EXCI` the `SYSID` option can be used.

With the `EXCI` precompiler option neither `DFHEIBLK` nor `DFHCOMMAREA` is generated as `PROCEDURE DIVISION USING` parameter.

The `EXCI` precompiler option is set by inserting a COBOL comment line containing from the seventh column:

```
*KIX--OPTION EXCI
```

before `IDENTIFICATION DIVISION` line.

Precompiler Controls

- `PROGRAM()` and `RETCODE()` are required for the `LINK` command in `EXCI`.
- `SYSID` is not recognized in `EXCI`.
- `COMMAREA` must be present if `LENTGH` or `DATALENGTH` is present.
- `EXCI CICS LINK` is the only recognized command in `EXCI`.

Access Authorization

The Tuxedo configuration `SECURITY` level drives the access authorization.

The `MT_EXCIAPPPROFILE` environment variable provides the application profile file name generated by the `genappprofile ARTKIX` tool. The default file name is `$HOME/.EXCIappProfile`.

In DPL service, the `EXEC CICS ASSIGN USERID()` command returns:

- `$USER` environment variable in the `EXEC` interface
- `USERID` value passed to the DPL request function of the `CALL` interface

See `ART Batch Runtime` documentation for more details.

ART-KIX Implementation

The programs linked via the EXCI interface are advertised by the DPL `KIX` server. They are named as `<program>_<sysid>`, where `<program>` is the linked program name (option `PROGRAM(<program>)` of `EXCI EXEC` interface), and `<sysid>` is the CICS system ID.

The EXCI interface uses the `<applid>` CICS application ID to address the appropriate CICS region. The relationship between `<applid>` and `<sysid>` is made via a specific DPL server service named `<applid>_info`.

The `-a` user parameter value of the DPL server command line (`CLOPT`) is used as `<applid>` value for the `_info` service.

If the `<applid>` is omitted by the client (without `APPLID(<applid>)` `EXCI EXEC` interface option), the `default_info` service is called. This service is advertised by the first DPL booted server.

The `_info` service returns the `<sysid>` associated to the server by the `-s` user parameter of the server command line.

ART Restrictions

Common EXCI Interfaces ART Restrictions

- The `TRANSID` has no meaning. There is no control on it. It is only passed to the DPL service in the `EIBTRNID` field in `DFHEIBLK` structure.
- The `COMMAREA` length is limited to 32763 bytes.

EXCI CALL Interface ART Restrictions

- Only `VERSION-1` is supported.
- The initial user `USER-NAME` is only used to generate a user-token without any control.
- The DPL `UOWID` is kept for compatibility only, and is not set and tested.
- The `PIPE-TYPE` has no meaning. The recognized values for `PIPE-TYPE` are only `X'00'` and (`X'C3'` or `X'D8'`) (`X'C3'` and `X'D8'` are the possible ASCII values for `X'80'` EBCDIC depending code-page). On other value the response code is set to 12 and the reason code to 498.

- The recognized values for `SYNC-TYPE` are only `X'00'` and (`X'C3'` or `X'D8'`) (`X'C3'` and `X'D8'` are the possible ASCII values for `X'80'` EBCDIC depending code-page). On other value the response code is set to 12 and the reason code to 499.

EXCI EXEC Interface ART Restrictions

The `DFHXCRM` replaceable-module is not treated.

SRRCMIT/SRRBACK Functions

The `SRRCMIT` and `SRRBACK` functions are available. `ATRCMIT` and `ATRBACK` functions are not supported.

`SRRCMIT` and `SRRBACK` functions must be coded as:

```
01  SRR-RETCODE                PIC 9(8) COMP-5.  
CALL "SRRCMIT" USING SRR-RETCODE  
CALL "SRRBACK" USING SRR-RETCODE
```

External CICS Interface (EXCI)