Oracle Tuxedo Application Runtime for CICS

User Guide

11*g* Release 1 (11.1.1.1.0)

March 2010

Oracle Tuxedo Application Runtime for CICS User Guide, 11*g* Release 1 (11.1.1.1.0)

Contents:

1. Introduction to CICS Runtime

2. Overview of the CICS Runtime

3. Initial Configuration of the CICS Runtime

4. Implementing CICS Applications

5. Reference

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.

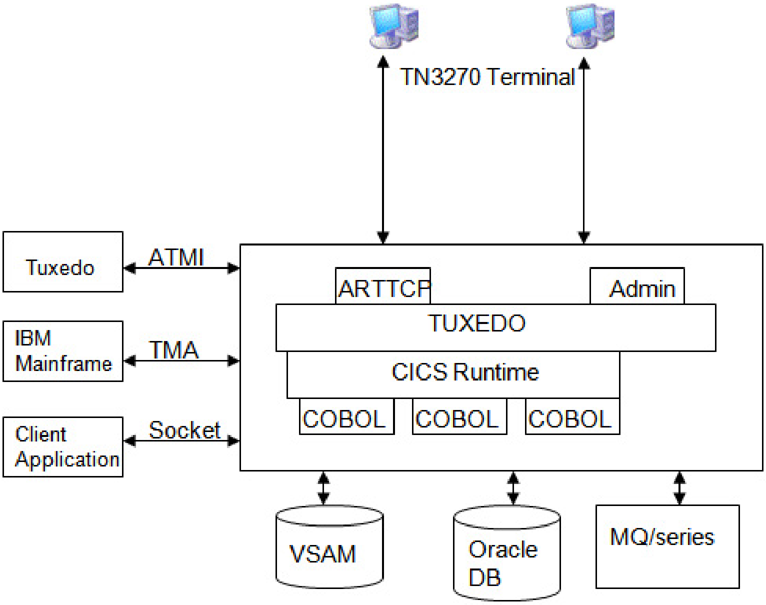
CICS is middleware that implements the control and integrity of shared resources, providing developers with APIs (EXEC CICS … END-EXEC statements) to dialog with CICS inside programs mainly developed on z/OS in COBOL, PL1 and Assembler languages.

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 Oracle Tuxedo features.

On a UNIX platform, Oracle 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, Oracle 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 Oracle Tuxedo and the converted CICS applications.

Figure 2-1 illustrates the CICS Runtime global architecture.

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 as shown in Listing 2-1.

Listing 2-1 z/OS CICS calls

```
*EXEC CICS
*     RECEIVE MAP    ('RTSAM10')
*             MAPSET ('RTSAM10')
*             INTO   (RTSAM10I)
*END-EXEC.
MOVE '  è? ???? ??? 00203   ' 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 as shown in Listing 2-2.

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
```

CICS Runtime Oracle Tuxedo Servers

The CICS Runtime Oracle 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.

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.
- TS Queue servers `ARTTSQ`, `TMQUEUE` and `TMQFORWARD`: manage the use of CICS Temporary Storage Queues - files managed by CICS thru specific commands.

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

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 anther 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 thru dedicated CICS statements like `EXEC CICS SEND` or `RECEIVE MAP` inside COBOL programs.

- Typeterms (`typeterms.desc` file)

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

CHAPTER 3

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 …).

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:

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".

#
```

```
# <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 $APPDIR
#      directory.)
#
# Copyright ï¿½1998, BEA Systems, Inc., all rights reserved.
#-----------------------------------------------------------------------------
-
# TUXEDO environment
APPDIR=${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 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.
- 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 servers 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.

LMID=

Name of the CICS where CSGM is running.

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 files

1. 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

description="IBM 327x family terminal"

hilight=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.

hilight=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 tmadmin 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 you have started only the mandatory servers `ARTTCPL` and `ARTCNX`, 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 tmadmin psc commands

You can also check that the required Tuxedo services are running using the `tmadmin 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 `tmadmin psc` **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.


> 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}`.

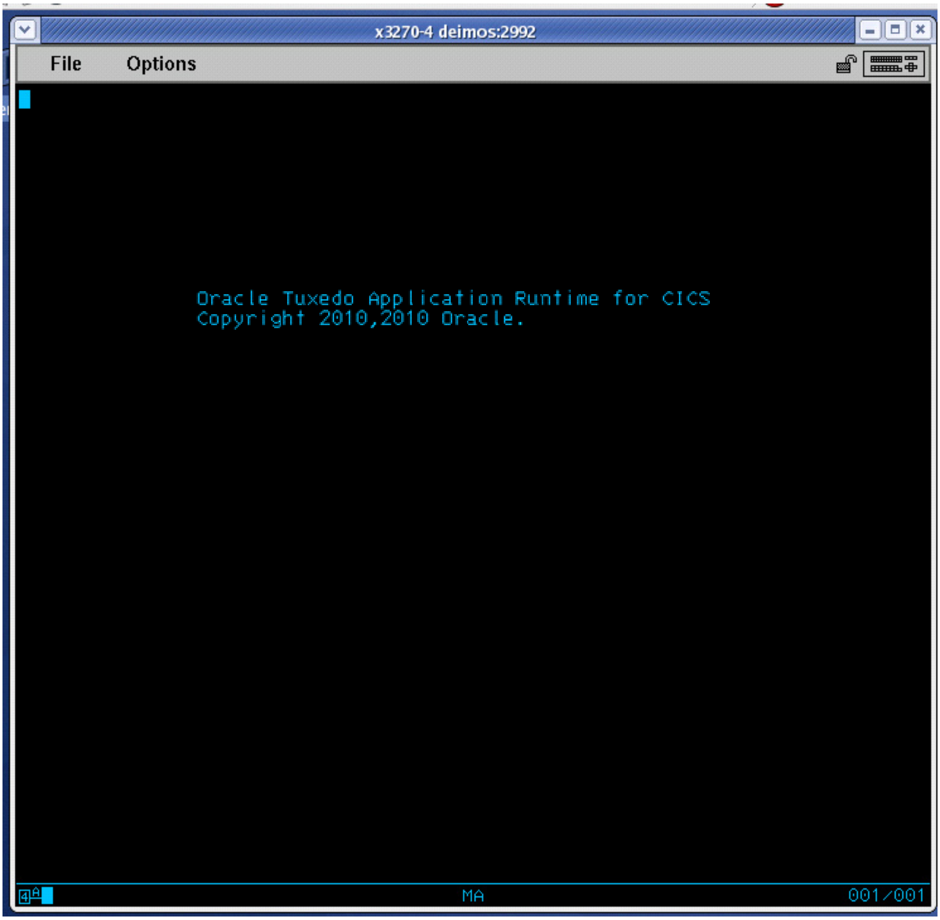
Where:

${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 for your UNIX 3270 emulator set up by your Tuxedo Administrator at installation time in the ubbconfig file.

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

Figure 3-1



```
Oracle Tuxedo Application Runtime for CICS
Copyright 2010,2010 Oracle.
```

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

Implementing CICS Applications

To illustrate the implementation of CICS applications, we will use the Simple Application application which is delivered with the CICS Runtime product.

Initially we will install an application using basic CICS features like standard transactions, COBOL programs, BMS mapsets and indexed files. Then we will progressively described the installation of others features.

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 Mapsets 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” on page 5-1 for a full description of which configuration files are used with each server.

To declare 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

To declare 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
```

To declare 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.

To declare 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.`

To declare 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.

To modify 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 `ubbconfig` 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 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=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.

To modify 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 tmadmin psr commands

Enter the Tuxedo `tmadmin 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 tmadmin psr Simple Application installation check

```
# 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      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.

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.

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 4-9 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 ARTC 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  ; Tranclass with maxactive set to 1; 1

TRCLASS2;SIMPAPP  ; Tranclass with maxactive set to 2; 2

TRCLAS10;SIMPAPP  ; Tranclass with maxactive set to 10; 10
```

The first transclass `TRCLASS1` has is maxactive parameter equal to 1, indicating that all the transaction belonging to this transclass 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: TRCLAS10

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

Listing 4-10 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 tmadmin psr commands

The `ARTSTR1`, is shown below:

Listing 4-11 Checking the ARTSTR1 server with the tmadmin psr commands

---

```
# 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
```

```
---------     ----------  --------     -- ------ --------- ---------------
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 tmadmin 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 4-12 Checking the ARTSTRN server with the tmadmin psc commands

```
# 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.

> 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 specifics `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 Severs 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 tmadmin 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 4-13 tmadmin commands showing parallel asynchronous transactions

```
# 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

---------      ----------  --------      -- ------ --------- --------------

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 )

ARTATRN       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  ARTATRN   GRP02     30       KIXR        0 AVAIL
ASYNC_SA03   atrsvc       ARTATRN   GRP02     30       KIXR        0 AVAIL
ASYNC_SA02   atrsvc       ARTATRN   GRP02     30       KIXR        0 AVAIL
ASYNC_SA01   atrsvc       ARTATRN   GRP02     30       KIXR        0 AVAIL
ASYNC_SA00   atrsvc       ARTATRN   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 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 4-14 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.
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 by specifics CICS `EXEC CICS START TRANSID` requests coded in the CICS programs using DELAY or TIME parameters to delay their execution.

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 Severs 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 4-15 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 4-16 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 which the server belongs to.

SRVID
Is the identifier of a Tuxedo Server.

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 4-17 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 )
ARTATRN       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   ARTATRN   GRP02     30       KIXR        0 AVAIL

ASYNC_SA03    atrsvc        ARTATRN   GRP02     30       KIXR        0 AVAIL

ASYNC_SA02    atrsvc        ARTATRN   GRP02     30       KIXR        0 AVAIL

ASYNC_SA01    atrsvc        ARTATRN   GRP02     30       KIXR        0 AVAIL

ASYNC_SA00    atrsvc        ARTATRN   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 4-18 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 4-19 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
---------     ----------  --------      -- ------ --------- --------------
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, transactions are not rolled-back to the last consistency point.

TS Queues are stored in a Micro Focus 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 ubbconfig 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.

To use Recoverable TS Queues

Yo 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.sh.`

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 `crstable.sh` 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 4-20 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/wo
rk9/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 4-21 Simple Application check for recoverable TS Queues

```
# 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 )

TMS_ORA        GRP02_TMS   GRP02      30001      0         0 (  IDLE )
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
#
```

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 ATRIBUTES` section filed:

Listing 4-22 Checking for remote programs

```
  DEF PROGR
  OVERTYPE TO MODIFY                                          CICS RELEASE = 0610
   CEDA  DEFine PROGram(         )
    PROGram      ==>
    Group        ==>
    DEscription  ==>
....
REMOTE ATTRIBUTES
 DYnamic      ==> No                 No ! Yes
```

```
REMOTESystem ==> XXXX
REMOTEName   ==> YYYYYYYY
Transid      ==> ZZZZ
EXECUtionset ==> Dplsubset          Fullapi ! Dplsubset
```

Where (CICS default values are underlined):

DYNAMIC(YES|<u>NO</u>)
: 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(<u>FULLAPI</u>|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 4-23 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 4-24 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 `ARTDPL` belongs.

SRVID
Is the identifier of a Tuxedo Server of `ARTDPL.`

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 4-25 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.

> 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
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  ARTATRN   GRP02     30       KIXR       0 AVAIL
ASYNC_SA03   atrsvc       ARTATRN   GRP02     30       KIXR       0 AVAIL
```

```
ASYNC_SA02    atrsvc        ARTATRN   GRP02     30        KIXR        0 AVAIL
ASYNC_SA01    atrsvc        ARTATRN   GRP02     30        KIXR        0 AVAIL
ASYNC_SA00    atrsvc        ARTATRN   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 `EXECUTIONSET` (the fifth field of the csv format dataset), to `DPL`.

   The default is `FULL`, 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 4-26 Simple Application programs.desc configuration of remote programs

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

Note: Nothing is declared for `RSSAT003`, meaning that the `EXECUTIONSET` field is set to `FULL` implying that this program is local.

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 4-27 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.

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 )
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
------------ ------------ ---------  --------  --    -------  ------ ------
KIXD_RSSAT0+ dplsvc       ARTDPL    GRP02    500        KIXR       0 AVAIL
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
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
```

```
SA01         kixsvc       ARTSTRN   GRP02     20       KIXR        0 AVAIL
SA00         kixsvc       ARTSTRN   GRP02     20       KIXR        0 AVAIL
ASYNC_QUEUE  ASYNC_QUEUE  ARTATRN   GRP02     30       KIXR        0 AVAIL
ASYNC_SA03   atrsvc       ARTATRN   GRP02     30       KIXR        0 AVAIL
ASYNC_SA01   atrsvc       ARTATRN   GRP02     30       KIXR        0 AVAIL
ASYNC_SA00   atrsvc       ARTATRN   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 4-28 Using tmadmin commands to check specific DPL service in verbose mode

```
# 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.

> 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 4-29 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 DFHSIT. 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 `DFHSIT`. 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 4-30 A COBOL example of use of the TWA

```
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).
    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 4-5 TWA size values associated to each transaction code of the Simple Application

Transaction	TWA Size
SA00	0
SA01	100
SA02	200
SA03	300

Listing 4-31 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 4-32 stderr_strn TWA example

```
|---------------------------------|
| TRANSACTIONS loaded : <   4>    |
|----------------------------------------------|----|-|-|---|-|-|---------
-|-----|-|--------|-----|---|
|    |         |                          |    |C|C|   |R|R|         |
|T|        |     |   |
|TRAN|  GROUP   |           PROGRAM          |ALIA|M|O|PRI|E|E|  STATUS
|TASK |R|  TRAN   | TWA |MAX|
|    |         |                          |    |D|N|   |S|S|         |DATA
|A|  CLASS | SIZ |ACT|
|    |         |                          |    |S|F|   |S|T|         |KEY
|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|        |00100|999|
|SA02|SIMPAPP   |RSSAT002                      |    |N|N|001|N|N|ENABLED
|USER |Y|        |00200|999|
|SA03|SIMPAPP   |RSSAT003                      |    |N|N|001|N|N|ENABLED
|USER |Y|        |00300|999|
```

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 in 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, ss shown in the following extract:

Listing 4-33 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         Alphanumeric
1000

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

./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 Servers 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 4-6 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 4-34 Example of the stdout_strn just after start up for a ARTSTRN server

```
Groups loaded: <0001>

|----------|

|  GROUP   |

|----------|

|SIMPAPP   |

|----------|

ARTSTRN: Read config done

|---------------------------------------------------|

| TRANCLASS loaded : <   2>                         |

|---------------------------------------------------|

|           TRANCLASS          |  GROUP   |MAXACTIVE|

|------------------------------|----------|---------|

|TRCLASS1                      |SIMPAPP   |      001|

|TRCLASS2                      |SIMPAPP   |      002|

|---------------------------------------------------|

|--------------------------------------------------|

| PROGRAMS loaded : <   4>                         |

|-------------------------------------------------------------------|
|            PROGRAM           |  GROUP   |LANGUAGE|EXEC|  STATUS  |
|                              |          |        |KEY |          |
|------------------------------|----------|--------|----|----------|
|RSSAT000                      |SIMPAPP   |COBOL   |USER|ENABLED   |
|RSSAT001                      |SIMPAPP   |COBOL   |USER|ENABLED   |
|RSSAT002                      |SIMPAPP   |COBOL   |USER|ENABLED   |
|RSSAT003                      |SIMPAPP   |COBOL   |USER|ENABLED   |
|------------------------------------------------------------------|
|---------------------------------|
| TRANSACTIONS loaded : <   4>    |
|----------------------------------------------|----|-|-|---|-|-|---------
-|-----|-|--------|-----|---|
|    |         |                          |    |C|C|   |R|R|         |
|T|        |     |   |
|TRAN|  GROUP   |            PROGRAM          |ALIA|M|O|PRI|E|E|  STATUS
|TASK |R|  TRAN  | TWA |MAX|
|    |         |                          |    |D|N|   |S|S|        |DATA
|A|  CLASS | SIZ |ACT|
|    |         |                          |    |S|F|   |S|T|        |KEY
|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!!
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 `-l` 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 4-35 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 4-36 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|
|---------------------------------------------------|
|--------------------------------------------------|
| 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>    |
|----------------------------------------------|----|-|-|---|-|-|---------
-|-----|-|--------|-----|---|
|    |         |                           |   |C|C|   |R|R|         |
|T|        |     |   |
|TRAN|  GROUP   |            PROGRAM           |ALIA|M|O|PRI|E|E|  STATUS
|TASK |R|  TRAN   | TWA |MAX|

|    |          |                            |    |D|N|   |S|S|         |DATA
|A|  CLASS | SIZ |ACT|

|    |          |                            |    |S|F|   |S|T|         |KEY
|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. So, to remove or add group(s) from an application, you have only to remove or add theses groups from this list for each CICS Runtime Tuxedo server.

Listing 4-37 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 4-38 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 -l SIMPAPP:GROUP1:GROUP2"
```

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

Listing 4-39 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 -l 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 theses 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.

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 " files

Servers	FILES	PROGRAMS	TRANCLASSES	TRANSACTIONS	TSQMODEL	Total
stderr_atr1	1	1	1	1	1	5
stderr_atrn	1	1	1	1	1	5
stderr_dpl					1	1
stderr_str1	1	1	1	1	1	5
stderr_strn	1	1	1	1	1	5
stderr_tsq					1	1
Total	4	4	4	4	6	22