Oracle Tuxedo Application Runtime for IMS

Users Guide

12*c* Release 2 (12.2.2)

April 2016

ORACLE

Oracle Tuxedo Application Runtime for IMS Users Guide, 12*c* Release 2 (12.2.2)

Oracle Tuxedo Application Runtime for IMS Users Guide

Oracle Tuxedo Application Runtime for IMS Users Guide

This chapter contains the following topics:

Overview

Oracle Tuxedo Application Runtime for IMS Users (Tuxedo ART for IMS), is used to emulate the same functions on open platforms (for example IBM IMS/TM on OS/390 or MVS/ESA environment).

This guide provides explanations and instructions for configuring and using Tuxedo ART for IMS when developing and running On-Line Transaction Processing (OLTP) applications on UNIX/Linux platforms.

This guide helps you to:

- Configure IMS Runtime software.
- Declare components to IMS Runtime.
- Run an IMS Application.

Introduction to IMS on z/OS

IMS consists of three components:

- the Transaction Manager (TM) component
- the Database Manager (DB) component, and
- a set of system services that provide common services to the other two components.

The IMS Transaction Manager provides you with access to applications running under IMS. You can be at a terminal or a workstation, or other application programs, either on the same OS/390 system, on other OS/390 systems, or on other non-OS/390 platforms. The IMS database manager component supports databases using the IMS hierarchical database model. It provides access to these databases from applications running under the IMS Transaction Manager. IMS has control regions and dependent regions (for example, message processing region, batch message processing (BMP) region, etc.).

The IMS control region provides the central point for an IMS subsystem. It provides the interface to the SNA network for the Transaction Manager functions and the Transaction Manager OTMA interface for access to non-SNA networks. It provides the interface to OS/390 for the operation of the IMS subsystem. It controls and dispatches the application programs running in the various dependent regions. The IMS MPP region is used to process messages input to the IMS

Transaction Manager component (that is, online programs). The address space does not automatically load an application program, but waits until work becomes available. The BMP region consists of two sub-types, Message Driven BMP (also called transaction oriented BMP) which reads and processes messages from the IMS message queue, and Non-message BMP (batch oriented) which do not process IMS messages but have access to the IMS DB.

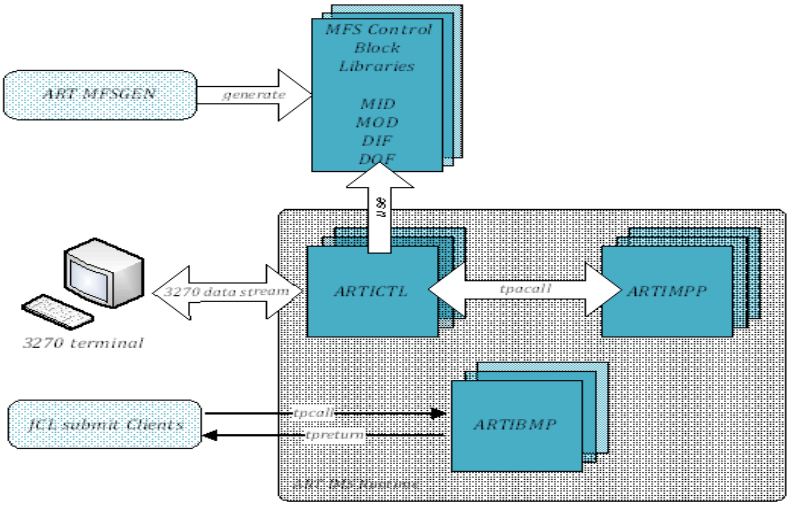
From IMS on z/OS to Oracle Tuxedo Application Runtime for IMS Users on UNIX

To emulate the functionality of IMS on z/OS, Tuxedo ART for IMS provides a group of servers running under Oracle Tuxedo control (including mandatory servers and optional servers). Mandatory servers include `ARTICTL`, `ARTIMPP` and `ARTIBMP`. Optional servers include `ARTIADM` and `ARTITERM`. In addition, Tuxedo ART for IMS provides a group of DLLs and utilities to assist the servers. The following is an IMS on z/OS to Tuxedo ART for IMS mapping list:

- IMS language utility `DFSUPAA0`: `MFSGEN` utility
- BMP utility `DFSRRC00`: Oracle Tuxedo client `DFSRRC00`
- Control region: Oracle Tuxedo `ARTICTL` server
- MPP region: Oracle Tuxedo server `ARTIMPP`, the container which runs MPP
- BMP region: Oracle Tuxedo server `ARTIBMP`, the container which runs BMP
- DLI address space: No dedicated address space for DLI in ART, provides a library which implements the DL/I APIs and runs in `ARTIMPP/ARTIBMP` address spaces
- Transaction: service

The overall Tuxedo ART for IMS architecture is shown in the following figure.

Figure 1 Tuxedo ART for IMS Architecture



Tuxedo ART for IMS consists of:

- ART MFS Language Utility - `MFSGEN`
- ART MFS Control Blocks Libraries
- ART Message Format Service (`ARTICTL`)
- ART Message Processing Region (`ARTIMPP`)
- ART Batch Message Processing Region (`ARTIBMP`)
- ART JCL Submission Clients – `DFSRRC00`
- A Mainframe IMS/DB support solution that includes an open system DB plug-in and a z/OS ODBA Proxy

Tuxedo ART for IMS receives a work request. The request is entered at a remote terminal. It is usually made up of a transaction code which identifies to IMS the kind of work to be done and the data that is used. They all should follow the MFS control block definitions parsed from original files on the Mainframe via the ART MFS Language Utility for `ARTICTL`.

`ARTICTL` handles the remote terminal connection and transfers the 3270 data stream from EBCDIC to ASCII. It then parses the information from the data stream according to DIF, saving the message segments into the memory according to MID for `ARTIMPP`.

ARTIMPP initiates and controls a specific program (in COBOL) which uses the request data to perform the remote operator request. It also prepares some data for the remote operator in response to the work request (for example, acknowledgment of work done, answer to a query, etc.).

Finally, the data prepared by the program is transmitted back to the terminal that originally requested the work, this process just reversing the work flow above.

Unlike ARTIMPP, ARTIBMP is not activated by the remote terminal, but by an Oracle Tuxedo client specific to ARTIBMP (for example, DFSRRC00).

Oracle Tuxedo Application Runtime for IMS Users Servers

The Tuxedo ART for IMS servers are:

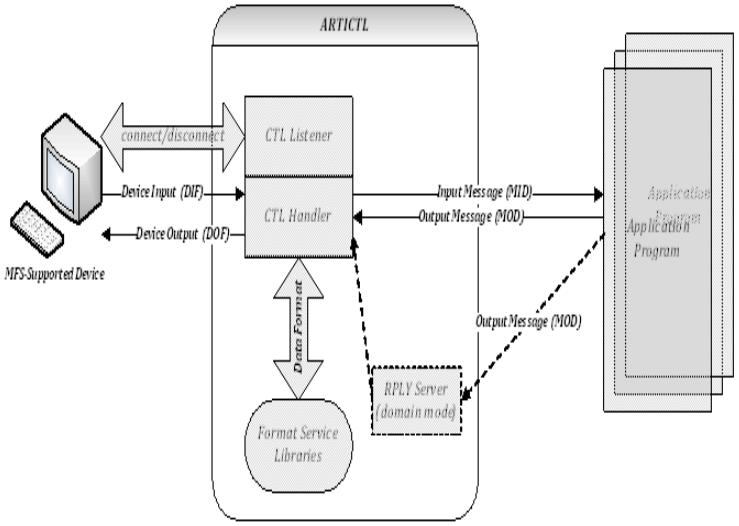
- ARTICTL
- ARTIMPP
- ARTIMPP_ORA
- ARTIBMP
- ARTIBMPT
- ARTIBMP_ORA
- ARTIADM
- ARTITERM
- ARTIGW

For more information, see Server Configurations in the Oracle Tuxedo Application Runtime for IMS Reference Guide.

ARTICTL

Figure 2 shows how ARTICTL can make an application program device-independent by formatting input data from the device or remote program for presentation to IMS, and formatting the application program data for presentation to the output device or remote program.

Figure 2 Device Independent Application Program



MS/MFS consists of three components:

- Terminal Control Listener (CTLL)
- Terminal Control Handler (CTLH)
- Format Service Libraries (LIBMFS)

Before describing the functions of each part in detail, another offline utility ART MFS language utility, `MFSGEN` must be mentioned. `MFSGEN` is used to transfer four types of MFS control blocks on Mainframes to binary files on open platforms so that the files can be read by Format Service Libraries. `MFSGEN` is also an important Tuxedo ART for IMS feature.

There are four types of MFS control blocks that you must specify to format input and output for the application program and the terminal or remote program:

- Message Output Descriptors (MODs): define the message layout MFS receives from the application program.
- Device Output Formats (DOFs): describe how MFS formats messages for each of the devices the program communicates with.
- Device Input Formats (DIFs): describe the formats of messages MFS receives from each of the devices the program communicates with.

- Message Input Descriptors (MIDs): describe how MFS further formats messages so that the application program can process them.

Note: Throughout this document, the term “message descriptors” refers to both MIDs and MODs. The term “device formats” refers to both DIFs and DOFs. Each of MOD, DOF, DIF and MID deals with a specific message. There must be a MOD and DOF for each unique message a program sends, and a DIF and MID for each unique message a program receives.

Terminal Control Listener

The Terminal Control Listener (CTLL) process is a standard Oracle Tuxedo server that runs in the ARTICTL subsystem. It starts when ARTICTL is initiated.

The CTLL performs the role of a terminal listener server. It listens at a public address that connects the application with a 3270. For each incoming connection request, it transmits this connection to one of its handler processes. Detailed functions are as follows:

- The CTLL process establishes a network port that connects to the 3270 terminal emulators. The port is “well-known”, that is, its address is available to any end user of a terminal emulator.
- The CTLL process spawns CTLH processes which listen on the well-known port for incoming connection requests. The CTLL process manages the number of CTLH processes dynamically, based on system load.
- The CTLL communicates with the CTLH processes utilizing various inter-process communication (IPC) mechanisms, such as shared memory and dedicated TCP/IP connections. Each CTLH can handle numerous terminal emulator clients and the CTLL tracks the number of connections each handler is servicing.

Terminal Control Handler

Terminal Control handler (CTLH) process manages multiple connections including terminal I/O, user authentication, calling the requested transactions on behalf of the user and etc. Each time a user requests a transaction, CTLH transmits (via `tpacall()`) this transaction request to the Messaging Processing Region (ARTIMPP). Detailed functions are as below:

Terminal I/O

When a connection request arrives from a terminal emulator, it is accepted and serviced by one of the CTLH processes. These processes manage inputs and outputs for the ARTICTL subsystem. When a terminal connects to the ARTICTL subsystem, the ARTCTLH acts as an Oracle Tuxedo

client on behalf of the terminal. When you enter a transaction ID from a terminal, the ARTCTLH converts the transaction ID into an Oracle Tuxedo service identifier and invokes `tpacall()`, Oracle Tuxedo then routes the transaction along with the terminal 3270 data stream to the MPP server to perform the transaction.

Session Management

The CTLH performs the session management. When you connect to ARTICTL subsystem via terminal, CTLH establishes a new user session for the connection and handles all subsequent screen I/O for the terminal. As a performance enhancement, each CTLH process can manage multiple sessions simultaneously. When you disconnect the emulator from the port, the CTLH terminates the session.

3270 Data Stream Transform

The CTLH performs the transform between ASCII and EBCDIC. After or before CTLH receives or sends the 3270 data stream from or to the terminal, it performs the data conversion between ASCII and EBCDIC.

IMS Message Formatting

The CTLH performs the message formatting by invoking Message Format Service Libraries (LIBMFS). When the CTLH receives data from a terminal, it splits the data stream and extracts the useful information according to the DIF control block, then composes the message to be used by application program according to the MID control block. When the CTLH receives the message returned from an application program, it splits the message and extracts the useful segments according to the MOD control block. It then composes the data stream to send to terminal according to the DOF control block.

There are three types of message formatting defined in each MID/MOD MSG statement (OPT=1/2/3). Each type of message formatting defines a different MSG definition handled by ARTICTL and ARTIMPP (while omitting some fields or segments of MSG for performance). For the benefits and details of each MSG types please refer to IMS/TM Programming.

Application programs do not need to be concerned about the MSG type used between ARTICTL and **ARTIMPP**; ARTIMPP justifies the MSGs in its I/O area before transmitting to application program. ARTICTL and ARTIMPP do not support OPT, they handle the MSG using common type (do not omit any fields or segments of MSG) which could be recognized by application programs.

Terminal Type Support

ARTICTL subsystem supports six terminal types as follows:

3270 Size(24, 80)

3270,2 Size(24, 80)

3270 - A2 Size(24, 80)

When the ARTICTL subsystem boots up, the CTLH performs TN3270E protocol negotiation, and the type and identity of a terminal are determined through the negotiation (such as "IBM-3278-2-E"), then the CTLH handles the 3270 data stream corresponding to the type of the terminal.

Control Blocks Libraries Management

The Message Format Service library (LIBMFS) is delivered as libraries linked by CTLH and run in each CTLH process, it also handles the control blocks management. It searches the correct Control Blocks and loads them into the cache when they are requested, and maintains them in the cache. When the request terminates, Control Blocks are cleared from the cache.

IMS System Commands Support

IMS System Commands, such as /FORMAT, /SIGN and /EXIT, are also handled in TCP handler.

IMS Security Support

Tuxedo ART for IMS supports four types of Oracle Tuxedo security mechanisms: none security, APP_PW, user-level authentication (USER_AUTH) and Access Control List (ACL or Mandatory ACL).

Since there is no application password in Mainframe IMS, we remove the application password in Tuxedo ART for IMS security.

Note: When using tmloadcf, do not fill any application password characters. Leave it NULL by pressing Enter.

- Adding user list and access control list via tpgrpadd, tpusradd and tpacladd.

  The first screen when the terminal connects to Tuxedo ART for IMS runtime is IMS Sign-on screen.. The username (username should not be null) must be filled if no security configured.

If authentication:

- Succeeds, a success screen is returned to the terminal.
- Fails, return to the Sign-on screen until authentication succeeds

If authorization fails, it will display some error message in terminal, But you still could clear the terminal and do any other transaction.

- Using system command "`/Sign off`" to sign off the current user.

  When the user signs off from Tuxedo ART for IMS runtime, the sign on screen appears, which the users can use to re-sign directly; the users could also clear the screen and then use system command "`/Sign`", "`/Sign on`", "`/Sign` on username password" to re-sign the Tuxedo ART for IMS runtime.

The maximal length of Username and Password is 8 (Chars) just as Mainframe.

For more information, see Using Security in ATMI Applications, in the Oracle Tuxedo Users Guide.

`ARTICTL` provides access to 3270 terminals through TCP/IP and message formatting services. `ARTICTL` formats screen based on the user input, receives input from 3270 terminal, converts the messages received from 3270 terminal to Oracle Tuxedo requests, sends the requests to `ARTIMPP` for processing, receives responses from `ARTIMPP` formats the response, and sends back to the originating terminal.

Tuxedo ART for IMS also support encryption and certificate authentication over network links between `ARTICTL` Handler and 3270 terminal. Currently the supported SSL versions for client include: SSL 2.0, SSL 3.0, SSL 3.1(TLS 1.0).

ARTIMPP

`ARTIMPP` in normal mode (without `-p` option in `CLOPT`) is an Oracle Tuxedo server designed to act as a service container. It advertises a set of services based on configuration file while initializing; calls corresponding COBOL/C application program while receiving a request to a service it advertised; and sends back the response to the requester, normally `ARTICTL` server. Service is the counterpart on UNIX to transaction code on Mainframe.

`ARTIMPP` in persistent mode (with `-p` option in `CLOPT`), monitors the /Q for every persistent TP transaction (transaction that is defined in `imsresource.desc`). Once there is a message in one /Q for one persistent transaction, it gets the message from the /Q and calls corresponding COBOL/C application program and then sends back a response to the requester.

Dynamic Service Advertising

`ARTIMPP`, in normal mode, dynamically advertises a set of services based on a set of configuration files while starting. All the services (transaction codes) to be contained in `ARTIMPP` servers are defined in imstrans.desc, each transaction code defined in it corresponds to a service with same

name to be advertised. imsapps.desc defines all the COBOL/C application programs to be called by ARTIMPP. Each $appname.psb defines the alternate PCBs required by that application. For information, see Oracle Tuxedo Application Runtime for IMS Users Configuration. If the configuration files are changed, ARTIMPP can only accept the changes while restarting. In addition, ARTIMPP only supports applications with type of TP.

Dynamic COBOL Program Invocation

Each service (transaction code) advertised by ARTIMPP has a defined COBOL application name to handle the service. While ARTIMPP receives a request to a service, ARTIMPP finds the COBOL application name corresponding to the requested service, and calls the corresponding COBOL program. Each COBOL application is compiled to a .gnt file and put in a directory in COBOL search order. For Micro Focus COBOL environment, the program search order is defined by environment variable $COBPATH. For COBOL-IT COBOL environment, the program search order is defined by $COB_LIBRARY_PATH.

Transaction Class Differentiating

Each Instance of ARTIMPP server can specify which classes of transaction codes are to be advertised by it. This mechanism can be used to adjust the deployment.

Implicit Transaction Commitment

If the COBOL application program called for a service does not explicitly commit or roll back the transaction, ARTIMPP server commits the transaction implicitly.

Program Switching

ARTIMPP supports that a request is forwarded by one transaction code to another transaction code, i.e., program switch. Program switch can be accomplished from a non-conversation transaction code to another non-conversation transaction code, from a conversation transaction code to another conversation or non-conversation transaction code. For the program switch between one conversation code and another conversation code, deferred/immediate switch are supported. Deferred program switch means the originating transaction code returns to the terminal with another transaction code (switch target) contained in SPA, when the terminal sends message again, the message will be routed to the switch target. Immediate program switch means the originating forwards a message to another transaction code, which responds to terminal. For program switch between non-conversation transaction codes, only immediate switch is supported. For the program switch from a response mode transaction code to a non-response transaction code, ARTIMPP does not have limitation on it, but users should be careful of designing

the program switch like this since response-mode transaction requires a response but non-response mode transaction may not respond to the terminal.

For immediate program switch via ALT PCB, if the target transaction is persistent transaction(transaction that is defined in imsresource.desc), the message for the target transaction will be put into /Q. ARTIMPP in persistent mode will handle this transaction. If the target transaction is non persistent transaction, ARTIMPP will call the transaction service and ARTIMPP in normal mode will handle this transaction.

Note: All the transaction codes in the switch chain must be accessible by the end user who executes the first transaction code in the chain if ACL is enabled as privilege control mechanism, otherwise the result is unpredictable, one of the possible results is the terminal may not respond.

ARTIMPP_ORA

ARTIMPP_ORA has all the functionalities of ARTIMPP. It can support an Oracle database used as an external resource manager (RM). It requires some libraries provided by Oracle database. To use ARTIMPP_ORA with an Oracle database, the RM section must be configured correctly in the UBBCONFIG file.

ARTIBMP

ARTIBMP is an Oracle Tuxedo server that advertises a fixed service ARTIBMP_SVC so that BMP clients can request this service to call specified BMP program written in COBOL.

Dynamic COBOL Program Invocation

ARTIBMP_SVC retrieves the specified BMP program name and associated PSB name from the message passed from the ARTIBMP client, verifies that the requested program is a valid batch program (configured in imsapps.desc and with type of BATCH) and the specified PSB is valid too, calls the program, and returns the result or completion notification to the client synchronously.

ARTIBMPT

ARTIBMPT is a transaction-oriented BMP server. When DFSRRC00 is called with IN assigned, the transaction code is passed to ARTIBMPT with other parameters. ARTIBMPT processes the transaction by calling a specified BMP program written in COBOL/C. Only BATCH applications are supported. ARTIBMPT can only process transaction oriented BMP applications. A transaction

oriented BMP application is a application that is defined in $MBR of parameters list and also defined in imsapps.desc with TYPE=BATCH.

Note: Currently, ARTIBMPT does not support client terminal messages. The transaction oriented BMP application/transaction must be a persistent transaction; this requires that the transaction must be defined in imsresources.desc.

ARTIBMP_ORA

ARTIBMP_ORA has all the functionalities of ARTIBMP. It can support an Oracle database used as an external resource manager (RM). It requires some libraries provided by Oracle database. To use ARTIBMP_ORA with an Oracle database, the RM section must be configured correctly in the UBBCONFIG file.

ARTIADM

In MP mode, ARTIADM can be booted up based on your choice, it downloads the configuration files from the master to the slave node. It is an Oracle Tuxedo server, and each node just needs to be deployed at most one ARTIADM. If your want to boot ARTIADM, it should be booted prior to ARTICTL and the ART_IMS_CONFIG environment variable should be set on each node.

ARTITERM

In cross-domain mode, if ARTICTL and ARTIMPP are not in the same domain, ARTITERM is used to pass the response from ARTIMPP back to ARTICTL. That is, ARTITERM acts as an intermediate from ARTIMPP to ARTICTL.

ARTIGW

ARTIGW is an Oracle Tuxedo server that acts as a bridge between non-terminal clients and the ARTIMPP server. Its main duties are as follows:

- Advertises separated services to handle the requests from non-terminal Oracle Tuxedo clients and requests from MQ applications.
- Message Mapping.

  For MQ application request messages, it converts an MQ message to a format that can be used by ARTIMPP. For reply messages, it converts an ARTIMPP reply message to an MQ message.

For non-terminal Oracle Tuxedo client request messages, ARTIGW plays a role in transferring client FML32 buffers to a message format that the program needs and sends the message to MPP. It then decodes the MPP message and then sends a standard FML32 buffer to the client.

- Session Management.

  Session management is used to associate the asynchronous ARTIMPP reply with the original ARTIGW client requests.

Oracle Tuxedo Application Runtime for IMS Users Deployment

Note: Tuxedo ART for IMS can only be deployed across homogeneous machines since COBOL application programs are called by ARTIMPP and ARTIBMP servers. Messages passed between Tuxedo ART for IMS servers are filled by COBOL programs, but Tuxedo ART for IMS servers are not aware of the copybook defined in the COBOL programs for messages.

There are three kinds of deployment environment for Tuxedo ART for IMS: SHM, MP and Domain. SHM means all the Tuxedo ART for IMS servers are deployed on one single machine. MP means Tuxedo ART for IMS servers are deployed across multiple machines belonging to a single Tuxedo domain. Domain means Tuxedo ART for IMS servers are deployed across multiple Tuxedo domains.

In SHM mode, ARTICTL and ARTIMPP are required; ARTIBMP is also required if the you need to run BATCH programs. In MP mode (besides the servers required in SHM mode), ARTIADM is also required. In MP mode, one single machine can contain any combination of ARTICTL and ARTIMPP/ARTIBMP. In domain mode, besides the servers required in MP mode, ARTITERM is also required in each domain where ARTICTL lives. The deployment of domain mode will be described in specific.

Note: In domain mode, ARTIADM for MP is not necessary.

ARTITERM exports a service whose name is composed with the domain id configured in the UBBCONFIG file for the domain and a hard-coded string "RPLYSVC" (i.e., ${DOMAINID}_RPLYSVC). The service name mentioned above should be configured in the DMCONFIG file *DM_REMOTE_SERVICES section of every remote domain.

In addition, to make sure a correct service name is exported by each domain where ARTITERM lives and make sure no conflicting among such services, the domain id field must be configured and kept unique in the UBBCONFIG for every domain.

For example, there are three domains to be deployed, the domain IDs configured in their UBBCONFIG file are DOM1, DOM2 and DOM3 respectively. ARTITERM servers exist in DOM1 and DOM2. According to the above deployment rule, the DMCONFIG file for DOM3 should declare DOM1_RPLYSVC and DOM2_RPLYSVC; the DMCONFIG file for DOM2 should declare DOM1_RPLYSVC, the DMCONFIG file for DOM1 should declare DOM2_RPLYSVC.

Oracle Tuxedo Application Runtime for IMS Users Configuration

General Limitations

DL/I call CHKP is used to send out the messages built and commit the changes made since last check point in IMS. DL/I call ROLB is to abort all the changes made and all the messages built but not sent since last check point. In Tuxedo ART for IMS, ARTIMPP or ARTIBMP need to treat the interval between two check points as a transaction. And possibly resource managers may be added in future.

Note: The NO_XA option cannot be configured in each domain where ARTIMPP or ARTIBMP lives.

Environment Variables

To enable Tuxedo ART for IMS, you must set the following environment variables before starting the servers:

- **IMSDIR**
- **ART_IMS_DB**
- **ART_IMS_CONFIG**
- **ART_IMS_FMT**
- **COBPATH**
- **COB_LIBRARY_PATH**

You should set IMSDIR to point to the installation root of Tuxedo ART for IMS product, set ART_IMS_CONFIG to specify the location of configuration files, set ART_IMS_FMT to specify the location of control block files, set ART_IMS_DB to specify the location of GSAM files, and set COBPATH to specify the location of COBOL .gnt files.

To enable Multi-Byte Character Set (MBCS) support, you must set following environment variables for `ARTICTL` before starting the servers:

- `INTERCODE`
- `EXTERCODE`

You should set `INTERCODE` to encoding type used in open platform, set `EXTERCODE` to `EBCDIC` encoding type used in z/OS platform.

For more information, see Oracle Tuxedo Application Runtime for IMS Reference Guide.

Configuration Files

All the configuration files in this section are case insensitive for key and non-literal values, for example `bool (yes|no)` and `enum`. Literal values and their cases are kept. Comment line should be prefixed with "*".

The configuration files are as follows:

- `imstrans.desc`: Defines IMS transaction codes
- `imsapps.desc`: Defines IMS applications
- `imsresource.desc`: Define persistent transactions
- `imsdbs.desc`: Defines IMS databases
- `$appname.psb`: Defines PSB

The general format for configuration files is shown in Listing 1.

Listing 1 General Configuration File Format

```
[section name]
Field1=value1
Field2=value2
….
[section name]
….
[section name]
…
```

For more information, see the Oracle Tuxedo Application Runtime for IMS Reference Guide.

Security Configuration

Tuxedo ART for IMS supports three types of Tuxedo security mechanism: application password (APP_PW), user-level authentication (USER_AUTH) and ACL/Mandatory ACL.

Authentication Configuration

In Oracle Tuxedo, each type of security mechanism requires that every user provide an application password as part of the process of joining the Oracle Tuxedo ATMI application, but In Tuxedo ART for IMS, it has been removed in order to keep the same behavior as IMS resides on z/OS. User should keep application password as NULL. For more information, see Using Security in ATMI Applications, in the Oracle Tuxedo Users Guide.

The USER_AUTH and ACL/Mandatory ACL security mechanism requires that each user must provide a valid username and password to join the Tuxedo ART for IMS runtime. The per-user password must match the password associated with the user name stored in a file named tpusr. Client name is not used. The checking of per-user password against the password and user name in tpusr is carried out by the Oracle Tuxedo authentication service AUTHSVC, which is provided by the Oracle Tuxedo authentication server AUTHSVR.

For more information, see Using Security in ATMI Applications, in the Oracle Tuxedo Users Guide.

SSL Configuration

Tuxedo ART for IMS uses the following existing UBBCONFIG file parametersto configure information about SSL identification strings and the location of SSL certificate encryption passwords:

- SEC_PRINCIPAL_NAME
- SEC_PRINCIPAL_LOCATION
- SEC_PRINCIPAL_PASSVAR

For more information, see Using Security in ATMI Applications, in the Oracle Tuxedo Users Guide

SIGN Command

Tuxedo ART for IMS supports three types of SIGN Commands:

- /SIGN
  - /SIGN
  - /SIGN USER-ID PASSWORD
- /SIGN ON USER-ID PASSWORD
- /SIGN OFF

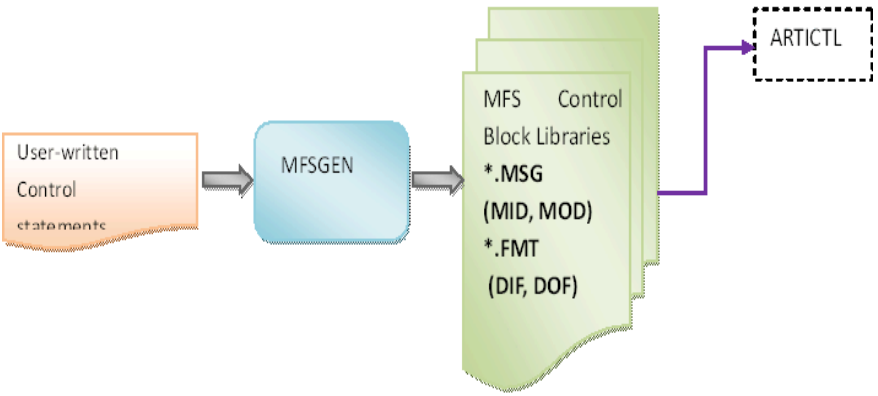
Oracle Tuxedo Application Runtime for IMS Users Utilities

Tuxedo ART for IMS uses the following utilities:

- chgcobol.sh
- DFSRRC00
- MFSGEN
- odbastop
- prepro-ims.pl
- RUNPROXY
- STOPROXY

For more information, see Oracle Tuxedo Application Runtime for IMS Reference Guide.

Figure 3 MFSGEN Utility



chgcobol.sh

Used to switch the COBOL runtime between Micro Focus COBOL and COBOL-IT if both exist. It can also show the current COBOL runtime being used without any arguments. For more information, see Oracle Tuxedo Application Runtime for IMS Reference Guide.

DFSRRC00

Used to activate the `ARTIBMP` server which is always waiting for the input from `DFSRRC00`. The parameter of `DFSRRC00` is a string, which should be passed from the script converted by workbench from JCL.

MFSGEN

This utility is used for ART MFS development. It converts user-written control statements to MFS binary control blocks.

Figure 3 depicts the `MFSGEN` workflow. For more information, see Oracle Tuxedo Application Runtime for IMS Reference Guide.

odbastop

An open system tool used to stop ODBA proxy on z/OS.

prepro-ims.pl

Used to transfer C programs (on z/OS) to proper C program format that can be invoked by Tuxedo ART for IMS. For more information, see Oracle Tuxedo Application Runtime for IMS Reference Guide.

RUNPROXY

A z/OS command used to start ODBA proxy on z/OS.

STOPROXY

A z/OS command used to stop ODBA proxy on z/OS.

Migrating COBOL Applications from IMS on z/OS to Oracle Tuxedo Application Runtime for IMS Users on UNIX

To migrate the COBOL applications running under the control of IMS on z/OS, you must do the following steps:

1. Convert the COBOL source code using ART Workbench
2. Customize configurations files of Tuxedo ART for IMS according to IMS Macros on z/OS,
3. Define and start an Oracle Tuxedo application consisting of Tuxedo ART for IMS servers, and executing desired COBOL applications by submitting a transaction code on 3270 terminal or run a JOB by ART Batch runtime.

Converting COBOL Programs

The COBOL source programs from IMS on z/OS should be firstly converted by ART Workbench. After converting, the converted source program can be compiled with Micro Focus COBOL or COBOL-IT compiler to.`gnt` files. For the details of converting COBOL source programs, please refer to the documents of ART Workbench. For the details of compiling COBOL source programs, please refer to the documents of Micro Focus COBOL or COBOL-IT.

For example, two COBOL source programs are converted and compiled.

`DFSIVAP1.cbl` (compiled to `DFSIVAP1.gnt`) is a MPP program

`DFSIVAP2.cbl` (compiled to `DFSIVAP2.gnt`) is a BMP program

The `.gnt` files should be put under `$COBPATH` (Micro Focus COBOL) or `$COB_LIBRARY_PATH` (COBOL-IT).

Implementing COBOL Program Debugging in IMS Runtime

Use Case 1:

Two users want to debug two MF BMP COBOL programs respectively.

User A wants to debug `COBOL Program1` and user B wants to debug `COBOL Program2`, do the following steps.

1. Add COBOL debug information to the `imsdebug.desc` configuration file as follows.

```
[cobol]
      USER=A
      APPNAME=program1
      DEBUGID=myDebugID1
[cobol]
      USER=B
      APPNAME=program2
      DEBUGID=myDebugID2
```

2. Start up ART IMS servers.

   User A and B starts their own ART IMS servers with the same Linux account which starts up the anim utility. Only the `DEBUGID` which the anim utility specifies is debugged.

3. Input anim command lines.

   User A inputs the following command line from the terminal: `anim %XmyDebugID1`.

   User B inputs the following command line fromthe terminal: anim %XmyDebugID2.

4. Start up the COBOL program and debug it.

Use Case 2:

A user wants to debug all CIT COBOL programs in one transaction.

User C wants to debug `COBOL Program1` and `Program2` in transaction `tranA`, do the following steps:

1. Add COBOL debug information to the `imsdebug.desc` configuration file as follows.

```
[cobol]
      USER=C
```

```
TRANNAME=tranA
DEBUGID = 111
```

2. Start up ART IMS servers and start transaction.
3. Input `deet` command lines.

   User C inputs the following command line from the terminal: `Deet -p 111,` and then debug.

   Note: You must detach when debugging is complete.

Customizing Configuration Files

To run a COBOL application migrated from IMS on z/OS in Tuxedo ART for IMS, some critical configuration files have to customized based on the IMS Macros related to the COBOL application.

To run an MPP program, there must be a transaction code corresponding to the program as shown in Listing 1.

Listing 1 imstrans.desc Example

```
[imstran]
name=TRAN1
response=no
edit=ULC
appname=DFSIVAP1
class=1
```

The transaction code `TRAN1` corresponds to application `DFSIVAP1`.

Each COBOL application must have its specific definition to identify it is an MPP or BMP (BMPT) program (TP or BATCH). Listing 2 shows an `imsapps.desc` example that defines two applications.

Listing 2 imsapps.desc Defining Two Applications

```
[imsapp]
name=DFSIVAP1
type=TP

[imsapp]
name=DFSIVAP2
type=BATCH
```

To run any applications in IMS, one PSB is required. In Tuxedo ART for IMS, PSB macro for an application is mapped to a .psb configuration file. For MPP program, the prefix of its .psb file should be the application name (i.e., $appname.psb; for BMP and BMPT programs, the prefix of its .psb file can be any name that complies to the naming rule of IMS applications. The .psb files for DFSIVAP1 and DFSIVAP2 are shown as below. One I/O PCB plus the PCBs defined in .psb are passed to the COBOL program as its parameters when the program is invoked.

Listing 3 DFSIVAP1 and DFSIVAP2 .psb Examples

```
DFSIVAP1.psb
[imspcb]
modify=yes
express=no

[imspcb]
modify=yes
express=no
```

From DFSIVAP1.psb, we can conclude that application DFSIVAP1 requires one I/O PCB and two alternate PCBs as its arguments.

Listing 4 DFSIVAP1 with two Alternate PCBs

```
DFSIVAPX.psb
[imspcb]
type=GSAM
name=DFSIVD5I
procopt=G

[imspcb]
type=GSAM
name=DFSIVD5O
procopt=LS

[imspcb]
type=GSAM
name=TSTIVD5O
procopt=LS
```

From `DFSIVAPX.psb`, we can conclude that application `DFSIVAP2` requires one I/O PCB and three GSAM PCBs.

For more information, see Oracle Tuxedo Application Runtime for IMS Reference Guide.

Defining an Oracle Tuxedo Application

To run a IMS COBOL application in Tuxedo ART for IMS, following servers must be started in a Tuxedo application:

`ARTICTL` - The server responsible for connection with 3270 terminals

`ARTIMPP` - The server responsible for executing MPP applications

`ARTIBMP` - The server responsible for executing BMP applications

For more information, see `UBBCONFIG(5)` in Section 5 - File Formats, Data Descriptions, MIBs and System Processes Reference in the Oracle Tuxedo documentation, and Tuxedo ART for IMS server parameters in the Oracle Tuxedo Application Runtime for IMS Reference Guide.

Running an MPP COBOL Program

To run the MPP program `DFSIVAP1`, open a 3270 terminal and connect to `ARTICTL` server with the hostname and port defined in `UBBCONFIG` file, then format the screen and input the transaction code `TRAN1`, `DFSIVAP1.gnt` is invoked by ARTIMPP server.

Running a BMP COBOL Program

To run the BMP program `DFSIVAP2`, the user need to convert the corresponding JCL on z/OS to shell script by ART Workbench so that it can be run by ART Batch runtime as a JOB. The JOB will invokes a utility `DFSRRC00`, which starts a specific advertised by `ARTIBMP` server, which in turn invokes the requested COBOL application. For information, see ART Workbench documentation.

Migrating C Applications from IMS on z/OS to Oracle Tuxedo Application Runtime for IMS Users on UNIX

To migrate the C applications running under the control of IMS on z/OS, you must do the following five steps:

1. Convert the C source code using `prepro-ims.pl`.
2. Customize configurations files of Tuxedo ART for IMS according to IMS Macros on z/OS,
3. Define and start an Oracle Tuxedo application consisting of Tuxedo ART for IMS servers, and executing desired C applications by submitting a transaction code on 3270 terminal or run a JOB by ART Batch runtime.

    a. Download C source files and dependency header file into local directory on UNIX.

    For example, two C source programs and ims.h are downloaded.

    `testmpp1.c` is a MPP program

    `testbmp1.c` is a BMP program

    `ims.h` is the header file supported in IBM IMS.

    b. Convert source code

```
prepro-ims.pl -i source-file -o dest-file [-m yourmakefile]
```

The ims.h from mainframe can also be processed by `prepro-ims.pl`. Lines beginning with "`??=`" or "`#pragma`" are commented out.

c. Set configuration file

To run an MPP program, there must be a transaction code corresponding to the program as shown in Listing 5.

Listing 5 imstrans.desc Example

```
[imstran]
name=TRAN3
response=no
edit=ULC
appname=TESTMPP1
class=1
```

Configure LANG in file imsapps.desc, LANG indicates the program to run is COBOL or C type as following:

Listing 6 imsapps.desc Defining Two Applications

```
[imsapp]
name=TESTMPP1
type=TP
LANG=C

[imsapp]
name=TESTBMP1
type=BATCH
LANG=C
```

To run any applications in IMS, one PSB is required. In Tuxedo ART for IMS, PSB macro for an application is mapped to a `.psb` configuration file. For MPP program, the prefix of

its `.psb` file should be the application name (i.e., `$appname.psb`; for BMP program), the prefix of its `.psb` file can be any name that complies to the naming rule of IMS applications. The `.psb` files for `TESTMPP1` and `TESTBMP1` are shown as below. One I/O PCB plus the PCBs defined in `.psb` are passed to the C program as its parameters when the program is invoked.

Listing 7 TESTMPP1.psb Examples

```
TESTMPP1.psb
[imspcb]
modify=yes
express=no

[imspcb]
modify=yes
express=no
```

Listing 8 TESTBMP1.psb with two Alternate PCBs

```
TESTBMP1.psb
[imspcb]
type=GSAM
name=DFSIVD6I
procopt=G

[imspcb]
type=GSAM
name=DFSIVD6O
procopt=LS

[imspcb]
type=GSAM
name=TSTIVD6O
procopt=LS
```

4. Compile source code.

   Modify makefile and run `gmake`

   Modify makefile by setting
   `ZOSINC=-I/path/containing/ims/header/from/mainframe`

   Modify makefile for that one program requiring more than one source files.

   e.g:

   `testmpp.c` will be compiled to `libartimstestmpp.so`

   `testbmp.c` will be `compiled to libartimstestbmp.so`

   The `APPNAME` in configuration file should be the same as application name converted by preprocessor.

   That is, `<filename> corresponds libartims<filename>.so`.

5. Set library search path.

   The environment variable `LD_LIBRARY_PATH` should be redefined by adding the directory containing these library files in the first of its original list.

   However, `LD_LIBRARY_PATH` is only for Linux and Solaris. On AIX, `LIBPATH` should be used instead.

   For example, `libartimstestmpp.so` and `libartimstestbmp.so` should be under the `LD_LIBRARY_PATH` (Linux/Solaris) or `LIBPATH(AIX)`.

Running an MPP C Program

To run the `MPP` program `testmpp`, open a 3270 terminal and connect to `ARTICTL` server with the hostname and port defined in `UBBCONFIG` file, then format the screen and input the transaction code `TRAN3`, `testmpp` is invoked by `ARTIMPP` server.

Running a BMP C Program

To run the BMP program `testbmp`, the user need to convert the corresponding JCL on z/OS to shell script by ART Workbench so that it can be run by ART Batch runtime as a JOB. The JOB will invokes a utility `DFSRRC00`, which starts a specific advertised by `ARTIBMP` server, which in turn invokes the requested C application. For information, see ART Workbench documentation.

Using imsgenconf to Generate Tuxedo ART for IMS Configuration

You can use the `imsgenconf` tool to generate Tuxedo ART for IMS configurations from z/OS definitions. Before using `imsgenconf` you must create a directory (definition directory) containing the following files:

- One IMS online transaction/batch application definition dataset:

  One dataset that contains `APPLCTN` and `TRANSACT` macro. Only could have one dataset.

- PSB files:

  Datasets that contain `PSBGEN` macro. Could have multiple datasets.

- DBD files

  Datasets that contain DBD macro. Could have multiple datasets.

If you only wants to generate Tuxedo ART for IMS configurations for particular Transaction/Batch applications, you can create a white list (`IMS.WHITE`) or black list (`IMS.BLACK`), under the top definition directory.

- If `IMS.WHITE` exists under top definition directory, `imsgenconf` only generates configurations for transactions/batch applications which are defined in the white list.
- If `IMS.BLACK` exists under top definition directory, `imsgenconf` only generates configurations for transactions/batch applications which are not defined in the black list.
- If both the `IMS.WHITE` and `IMS.BLACK` exist under the top definition directory, `IMS.WHITE` is the default.
- If both `IMS.WHITE` and `IMS.BLACK` does not exist under the top definition directory, imsgenconf will generate configurations for all transaction/Batch applications.

  For more information, see *Configuration Files* in the Tuxedo ART for IMS Reference Guide.

For DB related configuration in `imsdbs.desc`, `imsgenconf` only generates configurations for the DBs that are actually used by the Transaction/Batch applications.

Notes: For Batch or transaction oriented batch programs, if the program name is not equal to `PSBNAME`, you must manually change "`NAME=program`" in `imsapps.desc`.

The following must also be configured manually:

`imsresources.desc`: The persistent transaction definition configuration.

`$ART_IMS_CONFIG/$dbname/segments.desc` : Segment description file of one IMS/DB for ODBA solution.

`$ART_IMS_CONFIG/$dbname/$segments.desc` : Description file of one segment for one IMS/DB for ODBA solution.

Oracle Solution for IMS/DB

Tuxedo ART for IMS can simulate most of the functionalities of IMS/TM, but do not have the implementation of IMS/DB since IMS/DB is a specially designed hierarchy database. To enable the COBOL/C programs running under the control of Tuxedo ART for IMS access IMS/DB, Oracle provide the support for Open Database Access (ODBA) in Tuxedo ART for IMS.

With ODBA, the programs running in Tuxedo ART for IMS can connect to IMS/DB residing on z/OS transparently. Whenever a COBOL/C program issues a DL/I call to access IMS/DB, the DL/I call is translated by underlying libraries to a request message, which in turn is sent to a proxy residing on the same z/OS image as IMS/DB. The proxy performs database operations according to the received request and sends the result back to the program running in Tuxedo ART for IMS.

Components

The principle of incorporating database solutions is to design and develop it as a plug-in that can be plugged into Tuxedo ART for IMS and can play in effective immediately. Basically the plug-in is designed as a shared library and exports a required set of APIs which are called by Tuxedo ART for IMS servers during runtime. Any implementation of the plug-in should follow the definition. For more information, see the *Oracle Tuxedo Application Runtime for IMS Reference Guide*.

Plug-In Library

As the plug-in definition in reference guide, the support of IMS/DB through ODBA is also implemented as a shared library based on the definition and can be simply used to replace the default version bundled in Tuxedo ART for IMS installation.

The following is the support list of Oracle plug-in for IMS/DB:

Oracle plug-in for IMS/DB supports GSAM database, Data Entry database, and other Full Function Databases, that is all the kinds of IMS/DB databases except for MSDB (Main Storage DB).

Inside this implementation, GSAM DB is implementation on UNIX file system, others are supported through ODBA protocol provided by IMS system.

Using ODBA Proxy

For Oracle IMS/DB connection to IMS/DB on z/OS, there must be communication between the programs running under Tuxedo ART for IMS control on open systems and on z/OS. As required by the ODBA protocol, a separate address space running on the same z/OS image as IMS/DB is developed to communicate with the programs in Tuxedo ART for IMS and perform database operations on behalf of these programs.

This section contains the following topics:

Functionality

Oracle IMS/DB supports following DL/I calls (issued in COBOL/C program):

- `GU/GHU`
- `GN/GHN`
- `GNP/GHNP`
- `ISRT`
- `REPL`
- `DLET`
- `INQY`
- `FLD`
- `POS`
- `CHKP`
- `SYNC`
- `ROLB`

IMS/DB Configuration

ARTIMPP/ARTIBMP Configuration

You need configure parameter required by Oracle plug-in for IMS/DB for `ARTIMPP/ARTIMBP`. For example:

```
ARTIMPP  SRVGRP=GROUP1
         SRVID=5
         CLOPT="-A -- -l 1 -x -o zoshost:port:BEA1"
```

For more information, see the *Oracle Tuxedo Application Runtime for IMS Reference Guide*.

IMS/DB Configuration

Each database consists of various segments, while each segment consists of various fields. So the configuration for a database consists of two kinds of files, the segment definition and the field definition.

`imsdbs.desc` is located under `$ART_IMS_CONFIG`. Some fields of `imsdbs.desc` configurations are mapped from some DBD statement of IMS on z/OS.

segments.desc defines the segments within a database, while `$segname.desc` defines the fields within a segment, they are located under `$ART_IMS_CONFIG/db/$dbname`.

`$segname.desc` only exists for databases with access type of neither GSAM nor MSDB defined in `imsdbs.desc`.

You also need to configure related field in `$appname.psb`. For more information, see Tuxedo ART for IMS Reference Guide.

ODBA Proxy Installation

IMS/DB support is provided through ODBA. This support consists of two parts:

Note: ODBA Proxy directories are generated under the `$IMSDIR` directory after Tuxedo ART for IMS installation is completed.

- z/OS (assuming that Tuxedo ART for IMS is installed in the `$IMSDIR` directory).

  The components for z/OS are under `$IMSDIR/odbaproxy/mvs` directory. These include:

  - ORACLE.ODBA.BACK.XMI - an XMI file consisting of the executable files to be run on z/OS
  - A group of JCL jobs under ./jcl which can be used to install the package and run the ODBA Proxy on z/OS

- open systems

  Linux 64 bit (OEL/RHEL), AIX 64-bit, and Solaris (Sparc) 64-bit are supported. Under $IMSDIR/bin, odbastop is a utility used to stop ODBA proxy running on an MVS Under $IMSDIR/lib, libdlidb.so is the Oracle plug-in for Tuxedo ART for IMS runtime connection to IMS/DB.

Installing Oracle Tuxedo Application Runtime for IMS Users ODBA Proxy on z/OS

To install Tuxedo ART for IMS ODBA Proxy on z/OS, you must do the following five steps:

1. Upload JCLs
   a. Create a PDS (LRECL=80, RECFM=FB) on z/OS to upload JCL files, for example USER.ODBA.JCL.
   b. Upload all the JCL under mvs/jcl into this PDS as separate members (ftp text mode).
2. Create XMI Dataset
   a. Modify USER.ODBA.JCL(CREDS) to make it suitable for user's environment, especially VOL=SER parameter, possibly user may also want to modify the dataset name to be created, for example USER.ODBA.XMI.
   b. Submit CREDS job and make sure it complete successfully, the result is USER.ODBA.XMI is created.
3. Upload XMI

   Upload local file ORACLE.ODBA.BACK.XMI to dataset USER.ODBA.XMI on z/OS (ftp binary mode).
4. Receive XMI
   a. Modify USER.ODBA.JCL(RECEIVE) to make it suitable for user environment, INDSNAME specifies the input dataset of this JCL, it should be identical to the XMI dataset, i.e. USER.ODBA.XMI in step 2; DSNAME is the output file of this JCL, assuming it's changed to USER.ODBA.BAK.

b. Submit RECEIVE job and make sure it complete successfully, the result is `USER.ODBA.BAK` is generated.

5. Restore the PDS Containing the Executables

   a. Modify `USER.ODBA.JCL(RESTORE)` to make it suitable for user's environment, `VOL=SER` should be changed to the volume on which the target PDS consisting of the executables will be created, `RENAMEU` specifies the source name and the target name, the source name must be kept, user can modify the target name, for example `USER.ODBASERV.LOAD`. Set `DSN` to the `DSNAME` value which is set in step 4 (assume it is set to `USER.ODBA.BAK`).

   b. Submit **`RESTORE`** job and make sure it complete successfully, now the executables have been extracted into a PDS named `USER.ODBASERV.LOAD`.

DRA Configuration Best Practice for ODBA on z/OS

According to z/OS ODBA best practise documentation, frequently creating and tearing down threads can cause timing-related errors while your application program is running. It is recommended that you set `MINTHREADS` equal to `MAXTHREADS` in your DRA startup parameters.

Oracle Tuxedo Oracle Tuxedo Application Runtime for IMS Users Servers and ODBA Proxy

To use ODBA Proxy together with Tuxedo ART for IMS Servers, you must do the following four steps:

1. Start ODBA Proxy on z/OS

   a. Modify `USER.ODBA.JCL (RUNPROXY)` to make it suitable for user's environment, especially the DD in STEPLIB to correctly specify `USER.ODBASERV.LOAD` and the dependent DD

   b. Submit `RUNPROXY` job to start the proxy. Ensure that IMS ODBA environment has been set up before starting the proxy.

2. Start Tuxedo ART for IMS Runtime on Oracle Tuxedo Server

   For more information, see *Oracle Tuxedo Application Runtime for IMS Reference Guide*.

   After `UBBCONFIG` and Tuxedo ART for IMS resources have been defined, compile `UBBCONFIG` and run Tuxedo ART for IMS by starting its Oracle Tuxedo Domain using the `tmboot` command or controls provided in Oracle Enterprise Manager TSAMPlus plug-in.

3. To Stop Tuxedo ART for IMS Runtime

Tuxedo ART for IMS must be shutdown before shutting down ODBA proxy on z/OS. Use `tmshutdown` command or controls provided in Oracle Enterprise Manager TSAMPlus plug-in.

4. To Stop ODBA Proxy

   a. From the mainframe: modify `USER.ODBA.JCL(STOPROXY) JCL` and submit it to stop ODBA proxy

   b. From the open systems: run `odbastop` command, which will send a message to ODBA Proxy triggering the shut down.

Using Oracle Solution for IMS/DB

To use IMS/DB, you must do the following four steps:

1. Starting ODBA Proxy

   Before applying the Oracle plug-in, ODBA proxy should be firstly started and listening for connection request from Tuxedo ART for IMS servers. ODBA proxy should be started in batch mode by JCL For how to start ODBA proxy, please refer to ODBA Proxy Installation Guide.

2. Change configurations for IMS/DB

   Change related configuration files `$appname.psb`, `imsdbs.desc` (`segments.desc/$segname.desc` if necessary), etc.

3. Change configurations and start `ARTIMPP/ARTIBMP`

   Put ODBA parameter for the `ARTIMPP/ARTIBMP` configuration. Reload `UBBCONFIG` and start `ARTIMPP/ARTIBMP`.

4. Invoke Programs

   After Tuxedo ART for IMS servers are started properly, user can execute a transaction processing program or a batch program.

Using Dynamic BMP

Using Dynamic BMP causes the BMP server to exit after executing a BMP program, and then is restarted by the Oracle Tuxedo framework. To use dynamic BMP, you must do the following steps:

1. Set environment variable `ARTIMS_DYNAMIC_BMP` to "`Y`".

2. In UBBCONFIG file, configure "RESTART=Y" and "GRACE=0" for each BMP server, so that the Oracle Tuxedo framework can restart killed servers unlimited times.
3. In UBBCONFIG file, configure SCANUNIT*SANITYSCAN to a smaller value (for example 2 seconds), so that Tuxedo framework can restart the killed servers in shorter intervals.

Tip: For Linux, reducing the max open files number (ulimit -n) can speed up BMP server rebooting . This is useful for heavy loads.

Performance Analysis

To initiate ARTIMPP/ARTIBMP/ARTIGW transaction/application performance data and analysis, do the following:

- Enable performance tracing

  Add the UBBCONFIG file -V option for ARTIMPP/ARTIBMP/ARTIGW to enable performance tracing for the server, or use the IMS_PERF_ENABLE environment variable as a global switch for performance tracing.

  If IMS_PERF_ENABLE is set, it takes precedence over UBBCONFIG file setting.

  Note: If performance and debug tracing are set at the same time, debug tracing is automatically disabled.

- Performance trace data location

  If IMS_TRACE_PATH is defined, the performance trace is located at $IMS_TRACE_PATH. If IMS_TRACE_PATH is *not* defined, the performance trace is located at $APPDIR/log.

  The performance trace names are defined as follows:

  - perf.mpp.trace.$pid: for MPP
  - perf.bmp.trace.$pid: for BMP
  - perf.gw.trace.$pid: for GW

Tip: Clear all previously generated performance trace files before initiating performance tracing.

The File system will cache data for a while before writing to the trace file. For large amounts of data, if you want to get the complete trace, you must wait and then shutdown

the Oracle Tuxedo domain (or just the IMS server), so the file system can flush data to the trace file.

- Using analysis tool

  Use the `imsperf` tool (located at `IMS_RT/bin` to get transaction/application performance reports as follows:

  ```
  imsperf [-d path]
  ```

  `path` is where the performance trace is located.

  Its output is the performance analysis results (`tranname.csv, appname.csv`), which can be opened directly in `CSV` format.

See Also

- Oracle Tuxedo Application Runtime for IMS Reference Guide
- Oracle Tuxedo Application Runtime for Batch Reference Guide