Oracle Tuxedo Application Runtime for IMS

Users Guide

11*g* Release 1 (11.1.1.3.0)

December 2011

ORACLE

Oracle Tuxedo Application Runtime for IMS Users Guide, 11*g* Release 1 (11.1.1.3.0)

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 (ARTIMS), 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 Oracle Tuxedo ARTIMS 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 ARTIMS on UNIX

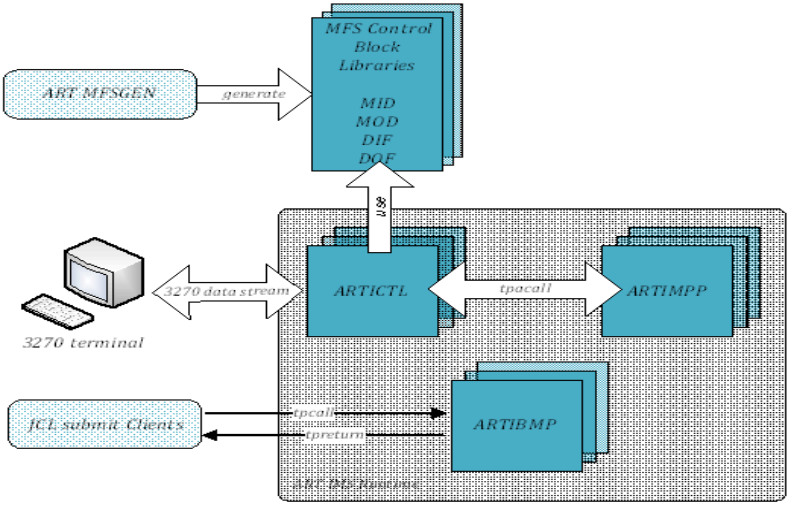
To emulate the functionality of IMS on z/OS, ARTIMS provides a group of servers running under Oracle Tuxedo control (including mandatory servers and optional servers). Mandatory servers include `ARTICTL`, `ARTIMPP` and `ARTIBMP`. Optional servers include `ARTIADM` and `ARTITERM`. In addition, ARTIMS provides a group of DLLs and utilities to assist the servers. The following is an IMS on z/OS to ARTIMS 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 ARTIMS architecture is shown in Figure 1.

Figure 1 ARTIMS Architecture



ARTIMS 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`

ARTIMS 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`).

ARTIMS Servers

The ARTIMS servers are:

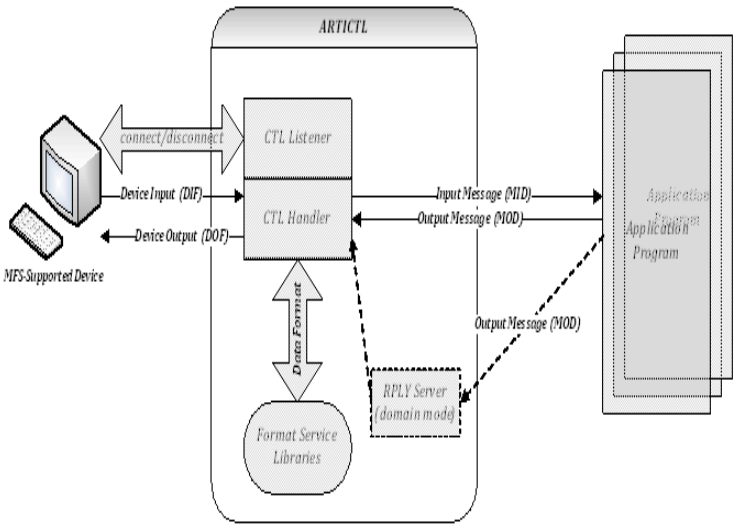
- ARTICTL
- ARTIMPP
- ARTIMPP_ORA
- ARTIBMP
- ARTIBMP_ORA
- ARTIADM
- ARTITERM

For more information, see 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 ARTIMS 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 a TUXEDO client on behalf of the terminal. When you enter a transaction ID from a terminal, the ARTCTLH converts the transaction ID into a 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

ART 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 ART 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 ART 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 ARTIMS 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 ART 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.

ARTIMPP

ARTIMPP 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 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.

Dynamic Service Advertising

ARTIMPP 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 application programs to be called by ARTIMPP. Each `$appname.psb` defines the alternate PCBs required by that application. For information, see ARTIMS 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 function with the support of MicroFocus libraries. Each COBOL application is compiled to a `.gnt` file and put in a directory in COBOL search order. For MicroFocus COBOL environment, the program search order is defined by environment variable `$COBPATH`.

Note: 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.

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 reqires some libraries provided by Oracle database. To use ARTIMPP_ORA with an Oracle database, the RM sectionmust 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.

ARTIBMP_ORA

ARTIBMP_ORA has all the functionalities of ARTIBMP. It can support an Oracle database used as an external resource manager (RM). It reqires some libraries provided by Oracle database. To use ARTIMPP_ORA with an Oracle database, the RM sectionmust 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.

ARTIMS Deployment

One general note should be paid much attention by the users of ARTIMS: ARTIMS can only be deployed across homogeneous machines since COBOL application programs are called by ARTIMPP and ARTIBMP servers and the message passed between ARTIMS servers are filled by the COBOL programs, but ARTIMS servers do not know about the copybook defined in the COBOL programs for messages.

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

In SHM mode, ARTICTL and ARTIMPP are required, ARTIBMP is required too if the users need to run BATCH program. In MP mode, besides the servers required in SHM mode, ARTIADM is required too. 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 for the domain and a hard-coded string “RPLYSVC”, i.e. ${DOMAINID}_RPLYSVC. The service name mentioned above should be configured in the `DM_REMOTE_SERVICES` section of DMCONFIG 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 3 domains to be deployed, the domain IDs configured in their UBBCONFIG are DOM1, DOM2 and DOM3 respectively. ARTITERM servers exist in DOM1 and DOM2. According to the above deployment rule, the DMCONFIG for DOM3 should declare DOM1_RPLYSVC and DOM2_RPLYSVC; the DMCONFIG for DOM2 should declare DOM1_RPLYSVC, the DMCONFIG for DOM1 should declare DOM2_RPLYSVC.

ARTIMS 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 ARTIMS, 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 ARTIMS,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 ARTIMS 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.

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
- 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

ART 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 ART 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 ART 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.

SIGN Command

ART IMS supports three types of SIGN Command:

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

ARTIMS Utilities

ARTIMS uses the following utilities:

MFSGEN

DFSRRC00

chgcobol.sh

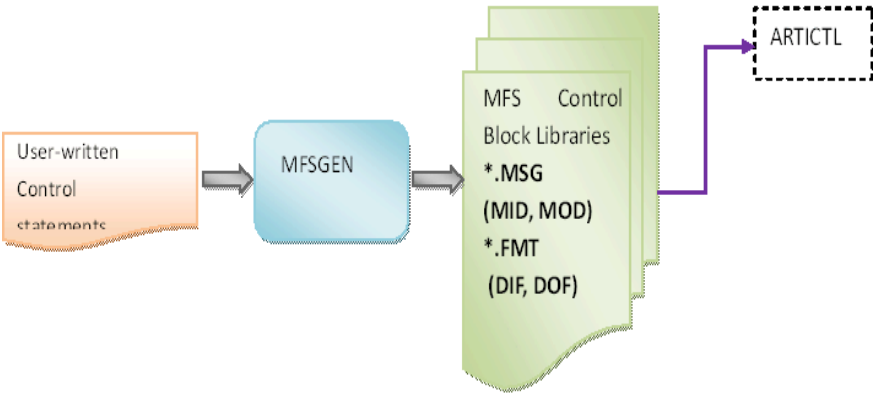
For more information, see Oracle Tuxedo Application Runtime for IMS Reference Guide.

MFSGEN

This utility is meant for the development of ART MFS. 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.

Figure 3 MFSGEN Utility



chgcobol.sh

`chgcobol.sh` is used to switch the COBOL runtime between Microfocus and COBOL-IT if they both exist. It can also show the current COBOL runtime being used without any arguments. Oracle Tuxedo Application Runtime for IMS Reference Guide.

DFSRRC00

`DFSRRC00` is 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.

Migrating COBOL Applications from IMS on z/OS to ARTIMS 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. customizing configurations files of ARTIMS according to IMS Macros on z/OS,
3. defining and starting an Oracle Tuxedo application consisting of ARTIMS 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 MicroFocus 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 MicroFocus 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 `$COBPATH` (MicroFocus) or `$COB_LIBRARY_PATH` (COBOL-IT).

Customizing Configuration Files

To run a COBOL application migrated from IMS on z/OS in ARTIMS, 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=2P
```

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

Each COBOL applicationmust have its specific definition identify it is a MPP or BMP 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 ARTIMS, 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 `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 ARTIMS, 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 ARTIMS server parameters in the Oracle Tuxedo Application Runtime for IMS Reference Guide.

Running a 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.

See Also

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