Oracle Tuxedo Application Runtime for IMS

Reference Guide

11*g* Release 1 (11.1.1.3.0)

December 2011

ORACLE

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

Oracle Tuxedo Application Runtime for IMS Reference Guide

Oracle Tuxedo Application Runtime for IMS Reference Guide

The Oracle Tuxedo Application Runtime for IMS Reference Guide describes system processes and commands delivered with the Oracle Tuxedo Application Runtime for IMS software.

This chapter contains the following topics:

ARTIMS Utilities

Table 1 ARTIMS Utilities

Name	Description
MFSGEN	Binary control blocks generator for server ARTICTL.
chgcobol.sh	Shell script used to switch between MicroFocus and COBOL-IT for ARTIMS.
DFSRRC00	Utility used to activate the ARTIBMP server.

MFSGEN

Name

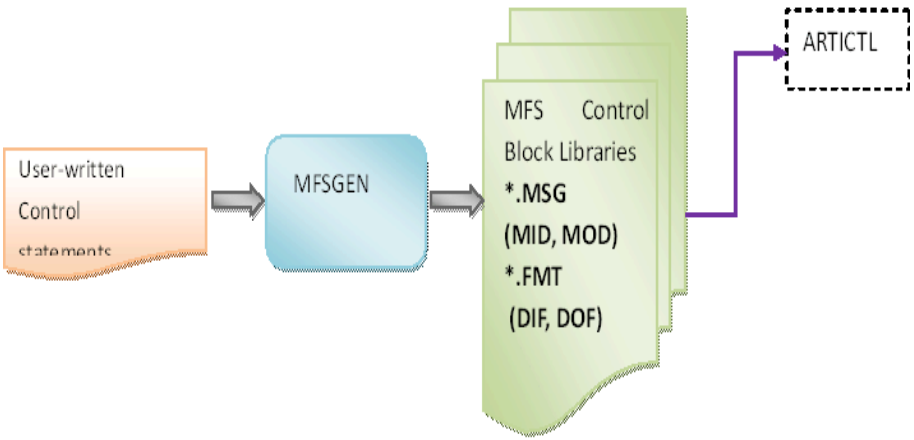
MFSGEN - Binary control blocks generator for server ARTICTL.

Description

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

Figure 1shows the MFSGEN workflow.

Figure 1 Mfsgen Workflow



Synopsis

```
mfsgen [-options…] files
```

Options

The command options are:

`-l`

Generate a listing (`.lst`) file for each input file.

`-d dir`

Specify an existing directory as the target directory to store all output files, including binary control blocks files (`*.MSG` and `*.FMT`)and listing files.

It is `<current directory>/format` by default.

Files

The MFSGEN utility creates the following files:

`FILE.MSG`

One category of binary files for MID and MOD control blocks. The name “FILE” should be obtained from input MSG definition statements.

`FILE.FMT`

One category of binary files for DIF and DOF control blocks. The name “FILE” should be obtained from input FMT definition statements.

`FILE.lst`

Source listing file. Here, “FILE” is the same as the base name of input file.

Return code

`0`

Success. All input file(s) is/are successfully parsed and converted to control blocks without any errors and warnings.

`1`

Success with warnings. All input file(s) is/are successfully parsed and converted to control blocks, but warnings exist.

`2`

Fail. Some/All of input file(s) are failed being parsed or converted to control blocks.

Example

To convert the source file file.mfs, use the following command:

```
$mfsgen file1.mfs  file.mfs   file3.mfs
```

Note: the input file `.mfs` suffix is not mandatory.

chgcobol.sh

Name

`chgcobol.sh` - shell script used to switch between MicroFocus and COBOL-IT for ARTIMS.

Synopsis

`chgcobol.sh` must be run under the root of IMS_RT
`chgcobol.sh [mf|cit]`

Description

You can have both MicroFocus and COBOL-IT installed at the same host, and may have requirement for switching from one to the other or back. `chgcobol.sh` is used to switch between MicroFocus and COBOL-IT. To switch COBOL runtime, ARTIMS application must be firstly shutdown.

`chgcobol.sh` takes the following options:

Without any option
: Show the current COBOL environment

cit
: Change COBOL Runtime to COBOL-IT

mf
: Change COBOL Runtime to MicroFocus

Example(s)

`./chgcobol.sh`
Output: Current COBOL Runtime is COBOL-IT

`./chgcobol.sh mf`
`Output: COBOL runtime has been changed to MicroFocus`

DFSRRC00

Name

`DFSRRC00` - Utility used to activate the ARTIBMP server

Description

`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. Currently, only the first 3 sub-parameters contained in the string is supported, i.e. "`BMP`, `$(PROG)`, `$(PSB)`,…", in which "`BMP`" is a hard-coded string, `$(PROG)` represents the batch program to be called, `$(PSB)` represents the PSB name to be used for the program, the remaining sub-parameters in the string are simply ignored.

Synopsis:

```
DFSRRC00 "BMP, $(PROG), $(PSB),……"
```

ARTIMS DL/I Support

In ARTIMS, DL/I is implemented in a group of dynamically loaded libraries. The supported DL/I functionalities are as follows:

- Message Processing
- Database Operation
- Plug-in Definition for Different Implementation of IMS/DB
- Transaction Management

Message Processing

DL/I is responsible for processing incoming messages and building outgoing messages against PCBs in IMS/TM. In ARTIMS, Tuxedo infrastructure is responsible for the message queue and message delivery, so the processing of incoming messages only involves the current request message. DLI library can retrieve the first and subsequent segments (FML fields) based on the request from any COBOL application. For building outgoing message, each PCB has an associated message buffer (FML) as the intermediate storage area, which holds the message data before the message is sent out. Detailed APIs for message processing are listed below.

Table 1 ARTIMS DL/I Processes and Commands

Name	Description
CBLTDLI	The 0 entry for DL/I calls in IMS/TM on OS/39
GU	Used to retrieve the first segment from the message queue in IMS/TM environment.
GN	Used to retrieve the subsequent segment from message queue in IMS/TM environment.
ISRT	Used to add a segment into the message associated with the specified PCB in IMS/TM.
PURG	Used to tell IMS/TM that the message is complete for non-express PCB
CHNG	Used to change the destination in PCB in IMS/TM

CBLTDLI

Name

CBLTDLI - The 0 entry for DL/I calls in IMS/TM on OS/39.

Description

In ARTIMS, CBLTDLI is a function acting as the entry of DLI library. CBLTDLI calls appropriate function based on the function code passed to it.

Parameter(s)

Function Code, e.g. “GU ”; I/O PCB or alternate PCB, I/O area, MOD

GU

Name

GU - Used to retrieve the first segment from the message queue in IMS/TM environment.

Description

GU is used to retrieve the first segment in a message. For conversational transaction, the first segment of a message is always SPA.

In ARTIMS, the simulated GU call is used to get the first field in the FML buffer of the message being processed. For conversational transaction, GU call always retrieve the field for SPA, otherwise retrieves the first field for user data.

I/O PCB
A pointer to the PCB that represents the source of the request

I/O Area
A pointer to a buffer to be filled in with the first segment

Result (status code)

'bb': successful (two blanks)

'AB': segment I/O area not specified

'AD': functional parameter invalid: function call not provided to CBLTDLI or invalid function call name provided to CBLTDLI

'QC': no input message

'QF': segment less than 5 characters

GN

Name

GN - Used to retrieve the subsequent segment from message queue in IMS/TM environment.

Description

After the last segment has been retrieved, a GN call results in a "QD" status code returned in PCB. In ARTIMS, the simulated GN call is used to get next field in the FML buffer of the message being processed.

Parameter(s)

I/O PCB
is a pointer to the PCB that represents the source of the request

I/O Area

s a pointer to a buffer to be filled in with the first segment

Result (Status Code):

'bb': successful (two blanks)

'AB': segment I/O area not specified

'AD': functional parameter invalid: function call not provided to CBLTDLI, invalid function call name provided to CBLTDLI

'QD': no more segments

ISRT

Name

ISRT - Used to add a segment into the message associated with the specified PCB in IMS/TM.

Description

In ARTIMS, the simulated ISRT call is used to add a field of CARRAY type into the FML buffer associated with the specified PCB. For conversational transaction, the first segment is always SPA.

Synopsis

```
I/O PCB or alternate PCB, I/O Area, MOD
```

Parameter

I/O PCB

is a pointer to the PCB that represents the destination of the outgoing message

I/O Area

is a pointer to a buffer that holds a segment to be sent. For conversational transaction code, the first segment must be SPA.

MOD

is the MOD to be used for the output message, the 8-byte MOD name must be left-justified and padded with blanks as necessary. It can be only accompanied with the first segment into a message.

Result (Status Code):

'bb': successful (two blanks)

'AB': segment I/O area not specified

'AD': functional parameter invalid: function call not provided to CBLTDLI, invalid function call name provided to CBLTDLI

'QF': segment less than 5 characters

'QH': No destination name in PCB

'XA': trying to forward the request to another transaction after responding the request

'XB': trying to respond the request after forwarding it to another transaction

'XC': Z1 bit is not 0, it is reserved and always kept as 0

PURG

Name

PURG - Used to tell IMS/TM that the message is complete for non-express PCB.

Description

PURG call is normally, but not send; or send out the message immediately for an express PCB.

If an I/O area is provided to PURG call, PURG call also acts as an ISRT call. That is, PURG marks the (current) message associated with the PCB as complete and "ISRT" the data in the I/O area as the first segment of the next message. The final result is same as a PURG call without I/O area followed by an ISRT call.

In ARTIMS, the simulated PURG call is used to mark the associated message as complete for a non-express PCB, or send out the associated message for an express PCB. If an I/O buffer is provided, however, it is ignored since multiple pending messages for a single PCB are not supported, therefore the MOD is ignored too. No special status code is added for this case, however, since the status code is checked by customer program.

Synopsis

```
I/O PCB or alternate PCB, I/O Area (optional), MOD (optional)
```

Parameter

I/O PCB

is a pointer to the PCB that represents the destination of the outgoing message

I/O Area

if provided, is a pointer to a buffer that is to be inserted as the first segment of next message

MOD

is the MOD to be used for the output message, the 8-byte MOD name must be left-justified and padded with blanks as necessary.

Result (Status Code):

'bb': successful (two blanks)

'AD': functional parameter invalid: function call not provided to CBLTDLI, invalid function call name provided to CBLTDLI

'A3': a modifiable TP PCB with no destination set but PURG called to it

CHNG

Name

CHNG -Used to change the destination in PCB in IMS/TM.

Description

In ARTIMS, the simulated CHNG is to specify another service name (only) in an alternate PCB. The destination transaction name is not greater than 8 bytes and is truncated to 8 if the limit is exceeded. The tailing blanks are removed too. The transaction name is evaluated as valid if it exists in the imstrans.desc file and has a legal configuration.

If one transaction code in one Tuxedo domain is designed to switch to another service in a different domain, the transaction code to be switched must be defined in the [imstrans.desc] configuration file as a valid transaction but should NOT be advertised with the control of its class.

For program switch, the new target is another transaction code. For conversational program switch, ARTIMS does not have limitation on the spa sizes of the originating code and the target code.

Synopsis

`Alternate PCB`, destination transaction code

Parameter

I/O PCB

is a pointer to the PCB that represents the destination of the outgoing message destination name is a string that represents the name of another transaction(service)

Result (Status Code):

‘bb’: successful (two blanks)

‘AD’: functional parameter invalid: destination not provided, functional call invalid

‘A2’: PCB is not modifiable or ISRT operation already done

‘QH’: the transaction to be specified in an alternate PCB is blank or invalid

Database Operation

DLI library performs the database operations issued from MPP or BMP programs. It can retrieves and hold one specific segment according the specified segment search criteria, update a specific segment, insert a segment at a specific position, delete a specific segment, and so on. Detailed APIs for database operation are listed below.

Table 2 Database Operation Processes and Commands

Name	Description
GU/GHU	Retrieve (and Hold) the first segment that satisfies the criteria (if any) from the current position (if any) or the beginning of the database
GN/GHN	Retrieve (and Hold) the next segment that satisfies the criteria (if any) from current position
GNP/GHNP	Retrieve (and Hold) the next segment that satisfies the criteria from the dependent segments of the established parent
ISRT	Used to insert a new occurrence of an existing segment type into a hierarchy database.
REPL	Used to update an existing segment
DLET	Used to remove a segment and its dependents.
FLD	Used to access a field within a segment.

GU/GHU

Name

GU/GHU - Retrieve (and Hold) the first segment that satisfies the criteria (if any) from the current position (if any) or the beginning of the database.

Description

GU is used to retrieve the first segment that satisfies the specified SSA and establishes a starting point for sequential processing. The search start point of GU is the beginning of the database, i.e. the root level. After locating the first segment that satisfies the call, current position is the starting position for sequential processing.

GHU locks the segment for sequential write operation, e.g. replace, delete, etc, in addition to GU.

Synopsis

```
DB PCB, I/O Area, and SSA list (optional)
```

Parameters

DB PCB
contains all the DB related information, especially the DB name

I/O Area
is used to received the returned segment

SSA:
the number of SSA is less than or equal to min (number of hierarchy levels, 15)

GN/GHN

Name

GN/GHN - Retrieve (and Hold) the next segment that satisfies the criteria (if any) from current position.

Description

GN is used to retrieve the next segment that satisfies the specified SSA, searching from current position. After locating the segment, the current position is updated for sequential processing. If no current position established in the DB, GN acts like GU, i.e. searching from the beginning. Sequential retrieval in hierarchy DB is always from top to bottom and from left to right, i.e. pre-order retrieval in a tree.

GHN locks the returned segment for sequential write operation on it, in addition to GN.

Parameters: DB PCB, I/O Area, and SSA list (optional)

The usage and restriction on parameters of GN are similar to that of GU.

GNP/GHNP

Name

GNP/GHNP - Retrieve (and Hold) the next segment that satisfies the criteria from the dependent segments of the established parent.

Description

GNP is used to retrieve the next qualified segment in the dependent segments of the established parent. The established parent in a hierarchy DB is the lowest-level segment returned in previous successful GU/GN call, and is canceled by an unsuccessful GU/GN call.

GHNP locks the returned segment for sequential write operation in addition to GNP.

Parameters

`DB PCB, I/O Area, SSA list (optional)`

ISRT

Name

ISRT - Used to insert a new occurrence of an existing segment type into a hierarchy database.

Description

ISRT is used to insert a new occurrence of an existing segment type into a hierarchy database. The insert location is determined by a series of qualified SSA excluding the level of the segment being inserted, or by current position if no qualified SSA.

Parameters

`DB PCB`, `I/O Area`, and `SSA list`

I/O

area contains the segment to be added

SSA

list contains a series of qualified/unqualified SSA to establish the position of the segment being inserted, the lowest-level SSA (i.e. the SSA at the level of the segment being inserted) must be unqualified. An unqualified SSA is satisfied with the first occurrence of the segment type.

REPL

Name

REPL - Used to update an existing segment.

Description

REPL call is used to update an existing segment. Firstly use a Get Hold call to retrieve the segment, then modify the segment and update the segment. The field length of the segment in the I/O area can't be changed.

Parameters

```
DB PCB, I/O Area, and SSA list (optional)
```

DLET

Name

DLET - Used to remove a segment and its dependents.

Description

DLET call is used to remove a segment and its dependents. It must follow a Get Hold call. Qualified SSA must NOT be specified for DLET call.

Parameters

```
DB PCB, I/O Area, and SSA list (optional)
```

FLD

Name

FLD - Used to access a field within a segment.

Description

FLD call is used to access a field within a segment. More info here?

Parameters

`DB PCB, I/O Area, and SSA list (optional)`

I/O

Area contains the Field Search Argument to locate a specific field.

Plug-in Definition for Different Implementation of IMS/DB

To support different kinds of implementation of IMS/DB, the support for IMS/DB is designed as a dynamically linked library (DLL) that can be plugged into ARTIMS and can be loaded by ARTIMS servers after being plugged. To enable this plug-and-play mechanism, there should be an agreement of what APIs exported by the DLL.

ARTIMS servers (ARTIMPP and ARTIBMP) are used as a container to run an online or batch COBOL program. To enable database access operations issued by the programs, normally the servers need to do some initialization or configuration for database implementation, need to do something prior to invoking a COBOL program, need to do something after completing the program, and need to do some cleanup before the servers go down. Besides, each database operation through "CBLTDLI" need to be mapped to a specific API.

Data structure Definition for IMS/DB Plug-in

A pointer to DB PCB structure is passed from ARTIMS servers to COBOL programs, and finally to `db_entry()` of plug-in for IMS/DB. To enable the plug-in work properly, the DB PCB structure should be filled in correctly before calling COBOL program each. This section defines the detailed requirement regarding how to fill in DB PCB.

The DB PCB structure example is shown in Listing 1.

Listing 1 DB PCB Structure

```
struct {
    char dbname[8];
    char seglevel[2];
    char stat_code[2];
    char opt[4];
    char res[4];
```

```
    char segname[8];
    char keylen[4];
    char segnum[4];
    char keyfa[IMS_FEEDAREA_LEN];
};
```

- **dbname**: If the PCB statement in PSB contains a `PROCSEQ=<name>`, populate dbname with `<name>`. Otherwise, populate `dbname` with the name in `NAME=<name>` from PSB.
- **seglevel**: Populate with NULL
- **stat_code**: Populate with spaces
- **opt**: Populate with value set to `PROCOPT` option from the PCB statement in PSB
- **res**: Do not populate with anything
- **segname**: Populate with NULL
- **keylen**: Do not populate with anything
- **segnum**:Do not populate with anything
- **keyfa**: Populate with NULL

API Definition for IMS/DB Plug-in

You must do the following steps to define an API for IMS/DB plug-in:

1. Initialization

   ```
   extern "C" int db_init(int argc, char * argv[])
   ```

   Functionality: Initialization for configuration or others required by an implementation

   Arguments: parameter list passed from CLOPT of the servers

   Return Value: 0 - success, -1 -- failure

   Where to use: This API is called while an ARTIMS server starts. If a specific implementation of IMS/DB doesn't need any initialization work, just provide an empty function and make it return 0.

2. Cleanup

   ```
   extern "C" int db_destroy()
   ```

Functionality: Cleanup for configuration or others required by an implementation

Arguments: None, because the servers can't provide input for Plug-in

Return Value: 0 - success, -1 -- failure

Where to use: This API is called while an ARTIMS server shuts down. If a specific implementation of IMS/DB doesn't need any initialization work, just provide an empty function and make it return 0.

3. Action Prior to Invoking a COBOL program

   ```
   extern "C" int db_pre()
   ```

   Functionality: Pre-Action required by an implementation before invoking a COBOL program which may access IMS/DB implementation

   Arguments: None, because the servers can't provide input for Plug-in

   Return Value: 0 - success, -1 -- failure

   Where to use: This API is called before ARTIMS servers invokes a COBOL program that may access IMS/DB implementation. If a specific implementation of IMS/DB doesn't need any initialization work, just provide an empty function and make it return 0.

4. Action After Invoking a COBOL program

   ```
   extern "C" int db_post()
   ```

   Functionality: Post-Action required by an implementation after invoking a COBOL program which may access IMS/DB implementation

   Arguments: None, because the servers can't provide input for Plug-in

   Return Value: 0 - success, -1 -- failure

   Where to use: This API is called after ARTIMS servers invokes a COBOL program that may access IMS/DB implementation. If a specific implementation of IMS/DB doesn't need any initialization work, just provide an empty function and make it return 0.

5. Database Access Entry

   ```
   extern "C"
   ```

   ```
   int dbentry(const char * op, __DB_PCB * pcb, void * io, char *
   ssa_list[], int ssa_cnt);
   ```

   Functionality: The entry point of accessing the IMS/DB implementation. Each DL/I call for database access issued from a COBOL program is finally handled by it.

   Argument:

op: function call name, e.g. "GU "

pcb: pointer to a __DB PCB, which is a superclass of user provided DB PCB.

io: pointer to a buffer, DB_IO_AREA is defined by the external implementer

ssa_list: an array of strings, each containing a SSA, the format is up to the external implementer

ssa_cnt: number of elements in ssa_list, the value range is [0..15]

Return Value: 0 - success; -1 - failure

Where to use: DL/I DB call issued by COBOL program

Default Implementation for IMS/DB

Within ARTIMS, a default DLL is provided as the placeholder of database plug-in. The default DLL contains only the implementation of GSAM database.

Transaction Management

DLI library performs the transaction management work, i.e .committing the changes that have been made and sending out the messages already built or rolling back all the changes and drop out all the messages, according to the direction passed from COBOL application. If the COBOL application does not issue a clear direction to commit the transaction, ARTIMPP commits the transaction.

Table 3 Transaction Management Processes and Commands

Name	Description
CHKP	Used to set an explicit commit point.
ROLB	Used to cancel the database updates

CHKP

Name

CHKP - Used to set an explicit commit point.

Description

CHKP call is used to set an explicit commit point. At a commit point, IMS/TM commits the changes made by application programs, normally database updates, sends out all the message marked as complete (by PURG for non-express PCB), and retrieves the next input message into the IOAREA provided.

In ARTIMS, the simulated CHKP is used to commit the changes already made by using `tpcommit()`. The messages that have been marked by complete are sent out. The messages that have not been marked by explicit PURG call are sent out too. Since there is only one message being processed, so no next message is retrieved.

Parameter

`I/O PCB, I/O Area`

I/O PCB
is a pointer to the PCB that represents a destination

I/O Area
is a pointer to a buffer to receive the next input message, it is ignored in DLI library

Result (Status Code)

‘ bb’: successful (two blanks)

‘AD’: functional parameter invalid: function call not provided to CBLTDLI, invalid function call name provided to CBLTDLI

ROLB

Name

ROLB - Used to cancel the database updates.

Description

ROLB call is used to cancel the database updates, cancels all the messages that were inserted but not available for transmission. For express PCB, the message is made available for transmission when IMS knows that the message is complete, i.e. when a PURG call is called. For non-express PCB, the message is not made available for transmission until the program reaches a commit point.

In ARTIMS, the simulated ROLB call is used to roll back all the changes made by the application program by using `tpabort()`, and empty the message buffers that have not been sent out.

Parameter

`I/O PCB, I/O Area`

I/O PCB
is a pointer to the PCB that represents a destination

I/O Area
is a pointer to a buffer to receive the next input message, it is ignored in DLI library

Result (Status Code):

‘ bb’: successful (two blanks)

‘AD’: functional parameter invalid: function call not provided to CBLTDLI, invalid function call name provided to CBLTDLI, or PCB not provided

Server Configurations

Table 4 Server Configuration Processes and Commands

Name	Description
ARTICTL	Used to join 3270 terminal to ART IMS Runtime
ARTIMPP	Service handler and container for TP type COBOL programs
ARTIMPP_ORA	Same as ARTIMPP. However, it requires the Oracle database as RM
ARTIBMP	Program container for BATCH type COBOL programs.
ARTIBMP_ORA	Same as ARTIBMP. However, it requires the Oracle database as RM
ARTIADM	A Tuxedo server responsible for the administration of ART IMS Runtime.
ARTITERM	Acts as messenger between ARTICTL and ARTIMPP located in different domains.

ARTICTL

Name

ARTICL -Used to join 3270 terminal to ART IMS Runtime.

Description

You must specify the MAXWSCLIENTS parameter in the MACHINES section of the UBBCONFIG file. MAXWSCLIENTS is the only parameter that has special significance for ARTICTL. MAXWSCLIENTS tells the Oracle ART at boot time how many accesser slots to reserve exclusively for 3270 terminals.

For MAXWSCLIENTS, specify the maximum number of 3270 terminal that may connect to a node. The default is 0. If not specified, terminal may not connect to the machine being described.

The syntax is `MAXWSCLIENTS=number`.

Synopsis:

```
ARTICTL SRVGRP="identifier"
                SRVID="number"
CLOPT="[servopts options]  -- -n netaddr -L pnetaddr [-m minh] [-M maxh] [-x
session-per-handler] [-D] [+H trace-level]"
```

Parameters:

-n netaddr

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

//computer:4000 designates port 4000 on machine computer. Character, 1-256, A-Za-z0-9[/:-]. Mandatory option.

-L pnetaddr

This address is used by the ARTICTL subsystem internally between TCPL and CTLH. The address is a string in standard internet URL format. For example: //computer1:4001 designates port 4000 on machine computer. Character, 1-256, A-Za-z0-9[/:-]. Mandatory option.

[-m minh]

The minimum number of handler processes that will be started by ARTICTL, minh is a number from 1 to 255, its default value is 1. The actual number of handler processes will always be between `minh` and **`maxh`** based on system load.

Note: Although `minh` is a number from 1 to 255, but it still should be equal to or smaller than (`FD_SETSIZE - 24`) according to the system resources limits. `FD_SETSIZE` means the maximum number of files that a process can have open at any time. The value can be acquired via system command `ulimit -n`.

[-M maxh]

The maximum number of handler processes started by `ARTICTL`, `maxh` is a number from 1 to the 1000; the default value is 1000. The actual number of handler processes is always between `minh` and `maxh` based on system load.

Note: Although `maxh` is a number from 1 to 1000, it should be equal to or smaller than (`FD_SETSIZE - 24`) according to the system resources limits. `FD_SETSIZE` means the maximum number of files that a process can have open at any time. The value can be acquired via system command ulimit `-n`.

[-x session-per-handler]

The number of sessions a CTLH can maintain concurrently in ARTICTL subsystem.

Numeric, 1-255. Default value is 32.

[-D]

Enable Debug, including the trace of TCPL and CTLH in ARTICTL subsytem.

[+H trace-level]

Specify the trace level:

-1: trace off. 0: trace for all CTLH. n (n>0): trace the first n CTLH.

When “-D” was specified without “+H trace-level”, the default CTLH trace level is: -1: trace off

All trace files will be placed in /tmp directory.

Examples:

```
*MACHINES
DEFAULT:
MAXWSCLINETS = 20
...
*SERVERS
ARTICTL SRVGRP="MFSGRP"
SRVID=1000
RESTART=Y GRACE=0
CLOPT="-- -n //hostname:4000 -L //hostname:4002 -m 1 -M 10"
```

ARTIMPP

Name

ARTIMPP -Service handler and container for TP type COBOL programs.

Description

ARTIMPP is a Tuxedo server to act as both service handler and container for COBOL programs of TP type, it invokes corresponding COBOL program according to the service request received.

Synopsis:

```
ARTIMPPSRVGRP=”identifier”
                SRVID=”number”
CLOPT="[servopts options] -- –l class_list -D -x parameter list for DB
plugin"
```

Parameters:

[-l class_list]

Specifies a list of transaction class, e.g. “1,3,5”; or a class range, e.g.“1-3”; or all classes, i.e *. The services whose class is specified in the `class_list` are advertised by ARTIMPP.

[-D]

Enables debug mode, trace is located at `$APPDIR/log`

[-x]

Indicates the server where the Database plug-in is to be used, the remaining parameter list following "-x" is passed to `db_init()`.

ARTIMPP_ORA

Description

`ARTIMPP_ORA` has all the functionalities of ARTIMPP. It can also support an Oracle database used as an external resource manager (RM). It uses on some libraries provided by Oracle database. In order to use this environment variable it with an Oracle database, the RM section must be configured correctly in the UBBCONFIG file.

ARTIBMP

Name

ARTIBMP - Program container for BATCH type COBOL programs.

Description

ARTIBMP is a Tuxedo server to act as the program container for COBOL programs of BATCH type, it invokes corresponding COBOL program according to the program name received.

Synopsis:

```
ARTIBMPSRVGRP=”identifier”
                SRVID=”number”
CLOPT="[servopts options] -- -D-x parameter list for DB plugin"
```

Parameters:

[-D]

enables debug mode, trace is located at `$APPDIR/log`.

[-x]

Indicates the server where the Database plug-in is to be used, the remaining parameter list following "-x" is passed to `db_init()`.

ARTIBMP_ORA

Description

`ARTIBMP_ORA` has all the functionalities of ARTIBMP. It can also support an Oracle database used as an external resource manager (RM). It uses on some libraries provided by Oracle database. In order to use this environment variable it with an Oracle database, the RM section must be configured correctly in the UBBCONFIG file.

ARTIADM

Name

ARTIADM - A tuxedo server responsible for the administration of ART IMS Runtime.

Description

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

When starting up, the administration server running on the master node reads in all the configuration files located in directory `${ART_IMS_CONFIG}`. When each administration server running on a slave node starts up, it communicates with the administration server on the master node and fetches the contents of the configuration files.

The administration server on the slave node then writes to the corresponding configuration files in directory `${ART_IMS_CONFIG}` on the slave node. New configuration files are created if none exist.

Synopsis

```
ARTIADMSRVGRP=”identifier”
                SRVID=”number”
CLOPT="[servopts options]"
```

ARTITERM

Name

ARTITERM - Acts as messenger between ARTICTL and ARTIMPP located in different domains.

Description

In cross-domain environment, ARTITERM server is used to act as messenger between ARTICTL and ARTIMPP located in different domains. ARTITERM provides a special service for ARTIMPP so that ARTIMPP can pass data to ARTITERM, which in turn passes the data to ARTICTL.

Synopsis:

```
ARTITERM SRVGRP=”identifier”
SRVID=”number”
CLOPT=””
```

ARTIMS MFS Support

IMS MFS Control Block Support

The definition of message formats and device formats is accomplished with separate hierarchic sets of definition statements. Table 5 shows all the definition statements and their descriptions.

Table 5 Definition Statements and Descriptions

Definition Statement Sets Name	Statement Name	Description
Message Definition Statement Set ---- Used to define message formats.	MSG	Initiates and names a message input or output definition.
	LPAGE	The optional LPAGE statement defines a group of segments comprising a logical page.
	PASSWORD	Identifies a field or fields to be used as an IMS password.
	SEG	Identifies a message segment.
	DO	Requests iterative processing of the subsequent MFLD statements.
	MFLD	The MFLD statement defines a message field as it will be presented to an application program as part of a message output segment. At least one MFLD statement must be specified for each MSG definition.
	ENDDO	Terminates iterative processing of the preceding MFLD statements.
	MSGEND	Identifies the end of a message definition.

Table 5 Definition Statements and Descriptions

Definition Statement Sets Name	Statement Name	Description
Format Definition Statement Set --- Used to define device formats.	FMT	Identifies the beginning of a format definition.
	DEV	Identifies the device type and operational options.
	DIV	Identifies the format as input, output, or both.
	DPAGE	Identifies a group of device fields corresponding to an LPAGE group of message fields.
	PPAGE	Identifies a group of logically related records that can be sent to a remote application program at one time.
	DO	Requests iterative processing of the subsequent DFLD statements.
	DFLD	Defines a device field. Iterative processing of DFLD statements can be invoked by specifying DO and ENDDO statements. To accomplish iterative processing, the DO statement is placed before the DFLD statements and the ENDDO after the DFLD statements.
	ENDDO	Terminates iterative processing of the previous DFLD statements.
	FMTEND	Identifies the end of a format definition.
END		Defines the end of the input file.

The “END” statement in above table defines the end of input file. All contents after it will be ignored. If the input file doesn’t have an “END”, MFSGEN will give an warning message and add one for the input file.

For each statement name, there are several fields. The detailed field name and value requirements are listed in Table 2 and Table 3. All other statements are assumed unsupported fields currently (listed in table 4). Sample 1 in Appendix A shows an example control statements file.

Table 6 Fields of each Message Definition Statement Set

Statement Name	Field	Possible Value	Note
MSG	TYPE	INPUT	Support
		OUTPUT	
	SOR=	(formatname, IGNORE)	"IGNORE" is a mandatory
	OPT=	1 or 2 or 3	Warning
	NXT=	msgcontrolblockname	Support
	PAGE=	No or YES	Warning
	FILL=	C' '	Warning if fill is not space.
		C'c'	
		NULL	
		PT	
LPAGE	SOR=	dpagename	Error
	COND=	(mfldname | mfldname(pp) | Segoffset, > | < | | | = | != , 'value')	
	NXT=	msgcontrolblockname	
	PROMPT=	(dfldname,'literal')	
PASSWORD	PASSWORD	blanks	Error
		comments	

Table 6 Fields of each Message Definition Statement Set

Statement Name	Field	Possible Value	Note
SEG	EXIT=	(exitnum,exitvect)	Warning
	GRAPHIC=	YES or NO	Warning
DO	count		Support
	SUF=	number	Support
MFLD	dfldname		Support
	'literal'		
	(dfldname, 'literal')		
	(dfldname, system-literal)		
	(,SCA)		TBD
	LTH=	1	Support
		nn	Support
		(pp, nn)	Warning
	JUST=	L or R	Support
	ATTR=	YES or NO, nn	Support
	FILL	X'40'	Warning
		X'hh'	Warning
		C'c'	Support only when c=SPACE
		NULL	Warning
	EXIT=	(exitnum, exitvect)	Warning

Table 6 Fields of each Message Definition Statement Set

Statement Name	Field	Possible Value	Note
ENDDO			Support
MSGEND			Support

Note: system-literals include: TIME, DATE1, DATE2, DATE3, DATE4, DATE1Y4, DATE2Y4, DATE3Y4, DATE4Y4, YYDDD, MMDDYY, DDMMYY, YYMMDD, YYYYDDD, MMDDYYYY, DDMMYYYY, YYYYMMDD, DATEJUL, DATEUSA, DATEEUR, DATEISO, LTSEQ, LTNAME.

For LTMSG and LPAGENO, user will get a warning message and they will not take any effect. For other strings not in above list, user will get a syntax error.

Table 7 Fields of each Format Definition Statement Set

Statement Name	Field	Possible Value	Support or Other
FMT			Support

Table 7 Fields of each Format Definition Statement Set

Statement Name	Field	Possible Value	Support or Other
DEV	TYPE=	3270	Support
		3270-A2 | | (3270,2)	Support
		(3270,1) | Other values	Error
	FEAT=	IGNORE	Support
		(CARD | NOCD | PFK | NOPFK | DEKYBD | PEN | NOPEN)	Warning
		1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10	Warning
	PEN=	dfldname	Warning
	CARD=	dfldname	Warning
	SYSMSG=	dfldlabel	Support
	DSCA=	X'value'	Support
		number	Support
	PFK=	(dfldname, 'literal'…)	Support
		(dfldname, integer='literal'…)	Support
		(dfldname, NEXTPP |NEXTMSG | NEXTMSGP | NEXTLP | ENDMPPI ….)	Warning
		(dfldname, integer= NEXTPP |NEXTMSG | NEXTMSGP | NEXTLP | ENDMPPI ….)	Warning
	SUB=	X'hh'	Warning
		C'c'	Warning
	PDB=	pdbname	Warning
DIV	TYPE=	INOUT	Support
		OUTPUT	

Table 7 Fields of each Format Definition Statement Set

Statement Name	Field	Possible Value	Support or Other
DPAGE	CURSOR=	(lll, ccc) (111,ccc,dfld)	Support; cursor > 1: Warning
	MULT=	YES	Warning
	PD=	pdname	Warning
	FILL=	PT or X'hh'	Warning
		C'c'	Support only when c=SPACE
		NULL	Warning
		NONE	Warning
	ACTVPID=	(for the 3290 in partition formatted mode)	Warning
PPAGE	comments		Error
DO	Count		Support
	1, MAX		Support
	line-inc, column-inc		Support
	Position-inc		Warning
	SUF=	number	Support
	BOUND=	LINE | FIELD	Support

Table 7 Fields of each Format Definition Statement Set

Statement Name	Field	Possible Value	Support or Other
DFLD	'literal'		Support
	PASSWORD		Error
	POS=	(lll,ccc)	Support
		(lll,ccc,pp)	Warning
	LTH=	nnn	Support
	PEN=	‘literal	Warning
		NEXTPP	Warning
		NEXTMSG	Warning
		NEXTMSGP	Warning
		NEXTLP	Warning
		ENDMPPI	Warning
	ATTR=	ALPHA|NUM	Support
		NOPROT|PROT	Support
		NODET|DET|IDET	Warning
		NORM|NODISP|HI	Support
		NOMOD|MOD	Support
		STRIP|NOSTRIP	Warning
	OPCTL=	tablename	Warning
	EATTR=	HD | HBLINK | HREV | HUL	Support

Table 7 Fields of each Format Definition Statement Set

Statement Name	Field	Possible Value	Support or Other
		CD | BLUE | RED | PINK | GREEN | TURQ | YELLOW | NEUTRAL	Support
		PX'00' | PX'hh' | PC'c' | EGCS | EGCS'hh'	Warning
		VDFLD | VMFILL, VMFLD | VMFILL | VMFLD	Warning
		OUTL | OUTL'hh' | BOX | RIGHT | LEFT | UNDER | OVER	Support
		MIX | MIXD	Warning
ENDDO			Support
FMTEND			Support

Table 8 Other definition statements and compilation statements which we don't support

Statement Name	Fields	Support or other
END		Support

Table 8 Other definition statements and compilation statements which we don't support

Statement Name	Fields	Support or other
PDB	LUSIZE	WARNINGS
	SYSMSG	
	PAGINGOP	
	LUDEFN	
PD	PID	
	VIEWPORT	
	VIEWLOC	
	PRESPACE	
	WINDOWOF	
	CELLSIZE	
	SCROLLI	
PDBEND	comments	
TABLE	comments	
IF	DATA	
	LENGTH	
	'literal'	
	NOFUNC	
	NEXTP	
	NEXTMSG	
	NEXTMSGP	
	NEXTLP	
	PAGEREQ	
	ENDMPPI	

Table 8 Other definition statements and compilation statements which we don't support

Statement Name	Fields	Support or other
TABLEEND	comments	
ALPHA	'EBCDIC literal character string'	
COPY	member-name	
EQU	number	Error
	alphanumeric identifier	
	literal	
	symbol	
RESCAN	OFF | ON	WARNING
	number	
STACK	ON | OFF	
	id	
UNSTACK	DELETE |KEEP	
	id	
TITLE	literal	
PRINT	ON | OFF	
	GEN | NOGEN	
SPACE	number	
EJECT	comments	

For the last column in above tables, "Support" means the field is supported; "Warning" means the field is not supported, and the tool ignores this field just like it is not specified, meanwhile a warning is generated; "Error" means the field is not supported, the tool reports an error and fails parsing current definition statement set.

Table 9 Message Dynamic Attribute Modification Support

Byte	Bit	Detail	Support
0	0-1	If both bits are on, requests that the cursor be placed on the first position of this field on the device. The first cursor-positioning request encountered in an LPAGE series (first MFLD with cursor attribute or cursor line/column value) that applies to a physical page is honored; these bits must be 00 or 11.	Yes
0	2-7	Must be off	Yes
1	0	Must be on	Yes
1	1	If on, replace If off, addition	Yes
1	2	Protected	Yes
1	3	Numeric	Yes
1	4	High-intensity	Yes
1	5	Nondisplayable	Yes
1	6	Detectable	Yes
1	7	Premodified	Yes

Table 10 Message Dynamic Modification of Extended Field Attributes Support

Type	Value	Detail	Support
01	0 – 4 bit Reserved 5 bit Mandatory fill 6 bit Mandatory field 7 bit Reserved	Validation replacement.	No
02	As above	Validation addition	No

Table 10 Message Dynamic Modification of Extended Field Attributes Support

Type	Value	Detail	Support
03	0 – 3 bit Reserved 4 bit Left line 5 bit Over line 6 bit Right line 7 bit Under line X'00' Default (no outline)	Field outlining replacement	Yes
04	As above	Field outlining addition	Yes
05	0 – 6 bit Reserved 7 bit SO/SI creation X'00' Default (no SO/SI creation)	Input control replacement	No
06	As above	Input control addition	No
C1	X'00' Device default X'F1' Blink X'F2' Reverse video X'F4' Underline	Highlighting	Yes
C2	X'00' Device default X'F1' Blue X'F2' Red X'F3' Pink X'F4' Green X'F5' Turquoise X'F6' Yellow X'F7' Neutral	Color	Yes
C3	Valid local ID values are in the range X'40'--X'FE', or X'00' for the device default.	Programmed Symbols	Yes

Table 11 Bit Settings for DSCA Field Support

Byte	Bit	Detail	Support
0	0-7	Should be 0	Yes
1	0	Should be 1	Yes
1	1	Force format write (erase device buffer and write all required data).	Yes
1	2	Erase unprotected fields before write.	Yes
1	3	Sound device alarm.	Yes
1	4	Copy output to candidate pointer.	No
1	5	Bit 1 - protect the screen when output is sent. Bit 1 - do not protect the screen when output is sent.	No
1	6-7	Should be 0	No

Security Configuration

Authentication configuration

In Tuxedo, each type of security mechanism requires that every user provide an application password as part of the process of joining the 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 on how to configure Tuxedo application password, please refer to Tuxedo documentation. 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 Tuxedo authentication service AUTHSVC, which is provided by the Tuxedo authentication server AUTHSVR. For more information on how to configure Tuxedo USER_AUTH and ACL/Mandatory ACL authentication, please refer to Tuxedo documentation.

Environment Variables

- **IMSDIR**
  An environment variable containing the root path (absolute path) of the installed ARTIMS subsystem.
- **ART_IMS_DB**
  is the container path where GSAM files are located.
- **ART_IMS_CONFIG**
  An environment variable required by ART IMS to specify the absolute path where the configuration files are located, such as *.desc and *.psb.
- **ART_IMS_FMT**
  An environment variable required by ARTICTL to specify the absolute path where the control block files which generated via MFSGEN are located. It is a series of paths similar to PATH environment variable, the separator is : .If this variable is not specified, the PATH APPDIR will be used.
- **COBPATH**
  An environment variable required by Microfocus COBOL environment. It defines one or more directories to search COBOL programs to be loaded dynamically. Its usage is similar to Unix PATH.
- **COB_LIBRARY_PATH**
  If you are using COBOL-IT, `COB_LIBRARY_PATH` is required by COBOL-IT to define the search order for COBOL programs. It defines one or more directories to search COBOL programs to be loaded dynamically. Its usage is similar to Unix PATH.

Commands and Parameters

Following table lists the commands and associated parameters that can be input on 3270 terminal and be processed by ARTIMS.

Table 12 3270 Terminal Commands and Parameters

Table 13 Command	Table 14 Shortening	Table 15 Parameter
/EXIT	/EXI	None

Table 12 3270 Terminal Commands and Parameters

Table 13 Command	Table 14 Shortening	Table 15 Parameter
/Format	/Forma, /Form, /For	modname
/Sign	/Sig	None Userid Passwd On On Userid Passwd Off

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:

- Transaction Definition - imstrans.desc
- Application Definition - imsapps.desc
- Database Definition - imsdbs.desc
- PSB Definition - $appname.psb

The general format for configuration files is as follows.

Listing 2 General Configuration File Format

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

```
[section name]
…
```

Transaction Definition - imstrans.desc

The fields in this configuration file are mapped from TRANSACT MACRO of IMS on z/OS

Table 16 Section Name: [imstran]

Field	Type	Value	Description	Filed in TRANSACT
NAME	X(1..8)	Mandatory	Transaction Code	CODE
SPA_SIZE	Integer	[16..32767]	SPA size	SPA
RESPONSE	Bool	Yes|No	Whether response is required for the transaction code specified in "NAME" field, default is No	MSGTYPE
EDIT	enum	UC/ULC	Whether the messages are converted to upper case automatically, default is UC	EDIT
APPNAME	X(1..8)	Mandatory	The name of the COBOL application program that processes the transaction code specified in "NAME" field	N/A
CLASS	Integer	[1..999]	The class of the transaction code, default is 1	MSGTYPE

Application Definition - imsapps.desc

The fields in this configuration file are mapped from APPLCTN MACRO of IMS on z/OS.

Table 17 Section Name: [imsapp]

Field	Type	Value	Description	Source in APPLCTN
NAME	X(1..8)	Mandatory	Application Name	N/A
PGMTYPE	Enum	TP|BATCH	TP for MPP, BATCH for BMP, default is TP	PGMTYPE

Database Definition - imsdbs.desc

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.

Table 18 Section Name: [imsdb]

Field	Type	Value	Description	Source in DBD
NAME	X(1..8)	database name	database Name	NAME
ACCESS	Enum	GSAM|MSDB|DEDB|HDAM|HIDAM|HISAM|HSAM|PHDAM|PHIDAM|PSINDEX|SHSAM|SHISAM	GSAM means GSAM DB	ACCESS
Following fields are only applicable to ACCESS=GSAM, will be ignored if ACCESS != GSAM				
DD1	X(1..8)	Literal String	Input File Name, the containing directory is defined by environment variable ART_IMS_DB	N/A
DD2	X(1..8)	Literal String	Output File Name, the containing directory is defined by environment variable ART_IMS_DB	N/A

Table 18 Section Name: [imsdb]

Field	Type	Value	Description	Source in DBD
RECFM	Enum	F|V	F: Fixed Record Length V: Variable Record Length	N/A
RECORD	Integer	positive integer	Record Length if RECFM=F, maximum record length if RECFM=V, its value range is 2~32579	N/A

PSB Definition - $appname.psb

$appname is the name of a COBOL application program with type of TP defined in imsapps.desc, $appname.psb is the PSB definition file corresponding to it. For application program with type of BATCH, $appname.psb is not used and the PSB must be provided by the script that calls DFSRRC00.

The fields in this configuration file are mapped from PCB statement for IMS on z/OS.

Table 19 Section Name: [imspcb]

Field	Type	Value	Description	Source in PCB
NAME	X(1..8)	blank or transaction code name	If the type is TP, this field represents a Transaction Code, only transaction code is supported for alternate PCB It is mandatory if modify = no, and optional if modify = yes. If the type is DB, this field represents the DB name. This field must be configured for a GSAM PCB, but optional for a DB PCB, please refer to PROCSEQ field for details.	NAME
TYPE	enum	TP|GSAM|DB	TP means Alternate PCB, GSAM means PCB for GSAM DB DB means PCB for DEDB	TYPE
Following fields are only applicable to TYPE=TP, i.e. alternate PCB, will be ignored if configured but TYPE!=TP				

Table 19 Section Name: [imspcb]

Field	Type	Value	Description	Source in PCB
MODIFY	Bool	Yes|No	Whether this PCB is modifiable, default is no	MODIFY
EXPRESS	Bool	Yes|No	Whether it is an express PCB, default is no	EXPRESS
Following fields are only applicable to TYPE=GSAM|DB, i.e. GSAM PCB, will be ignored if configured but TP!=GSAM|DB				
PROCSEQ	X(8)	Indexing Database name	If this field is configured, it should be filled in the first 8 bytes of DB PCB; otherwise, the value specified by NAME= should be filled in the first 8 bytes of DB PCB. At least one of NAM= and PROCSEQ= must be configured for a DB PCB.	PROCSEQ
PROCOPT	X(4)	GSAM: one of G|L|GS|LS DEDB: One or Combination of A|G|I|R|D|P|O|N|T|E|L|GS|LS|H	This field defines the access permission for the associated database. This field is only valid when TYPE=GSAM|DB, and is ignored if configured but TYPE=TP. GSAM: G|GS - Get Only; L|LS - Get and Append DEDB: TBD DEDB: ARTIMS servers don't check the validity of procopt, which is migrated from IMS on z/OS and only used by DB plug-in	PROCOPT

See Also

- Oracle ARTIMS Users Guide