

Oracle Tuxedo Application Runtime for IMS

Reference Guide

12c Release 1 (12.1.1)

September 2013

ORACLE®

Copyright © 2012, 2013 Oracle and/or its affiliates. All rights reserved.

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

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

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

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

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

Oracle Tuxedo Application Runtime for IMS Reference Guide

ARTIMS Utilities	1
MMSGEN	2
chgcobol.sh	4
DFSRRC00	5
prepro-ims.pl	6
odbastop	10
STOPPROXY(z/OS)	11
ARTIMS DL/I Support	11
Supported DL/I Interfaces	12
CBLTDLI	12
AIBTDLI	12
CTDLI	15
Message Processing	16
GU	17
GN	18
ISRT	19
PURG	21
CHNG	22
CMD	23
GCMD	25
Database Operation	26
GU/GHU	27
GN/GHN	29
GNP/GHNP	29
ISRT	30
REPL	31

DLET	31
FLD	32
POS	33
OPEN	34
CLSE	35
Plug-in Definition for Different Implementation of IMS/DB	36
Data structure Definition for IMS/DB Plug-in	36
API Definition for IMS/DB Plug-in	39
Default Implementation for IMS/DB	41
Transaction Management	41
CHKP (Basic)	42
CHKP (Symbolic)	44
ROLB	46
ROLL	47
SYNC	48
INQY	49
XRST	50
ARTIMS MFS Support	52
IMS MFS Control Block Support	52
ARTIMS Non-Terminal Access Support	67
Programming Interface	68
Programming Interface for Non-terminal Oracle Tuxedo Clients	68
Supported MQ Messages for MQ Applications	69
Configuration	70
Oracle Tuxedo MQ Adapter Configuration	70
ARTIGW Configuration	71
Cross Domain Configuration	71
Limitations	72

Non-Terminal Oracle Tuxedo Client Limitations	72
MQ Application Limitations	72
ARTIMS Persistent Message Support	73
Server Configurations	73
ARTICTL	74
ARTIMPP	76
ARTIMPP_ORA	77
ARTIBMP	78
ARTIBMPT	78
ARTIBMP_ORA	80
ARTIADM	80
ARTITERM	81
ARTIGW	81
ODBAPROX	83
Security Configuration	83
Authentication configuration	83
Environment Variables	84
Commands and Parameters	85
Configuration Files	86
Transaction Definition - imstrans.desc	86
Application Definition - imsapps.desc	87
Persistent Transaction Definition - imsresource.desc	88
Database Definition - imsdbs.desc	89
PSB Definition - \$appname.psb	91
Segments Definition - segments.desc	94
Segment Definition - \$segname.desc	96
See Also	101

Oracle Tuxedo Application Runtime for IMS Reference Guide

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

This chapter contains the following topics:

- [ARTIMS Utilities](#)
- [ARTIMS DL/I Support](#)
- [ARTIMS MFS Support](#)
- [ARTIMS Non-Terminal Access Support](#)
- [Server Configurations](#)
- [Security Configuration](#)
- [Environment Variables](#)
- [Commands and Parameters](#)
- [Configuration Files](#)

ARTIMS Utilities

[Table 1](#) lists the ARTIMS utilities.

Table 1 ARTIMS Utilities

Name	Description
<code>MFSGEN</code>	Binary control blocks generator for ARTICTL server
<code>chgcobol.sh</code>	Shell script used to switch between MicroFocus and COBOL-IT for ARTIMS.
<code>DFSRRRC00</code>	Utility used to activate the ARTIBMP server.
<code>prepro-ims.pl</code>	Utility used to transfer C program on z/OS to the format that could be run in ARTIMS.
<code>odbastop</code>	A tool on open system and it is used to stop ODBA proxy on z/OS
<code>STOPPROXY(z/OS)</code>	A JCL used to stop ODBA proxy on z/OS.

MFSGEN

Name

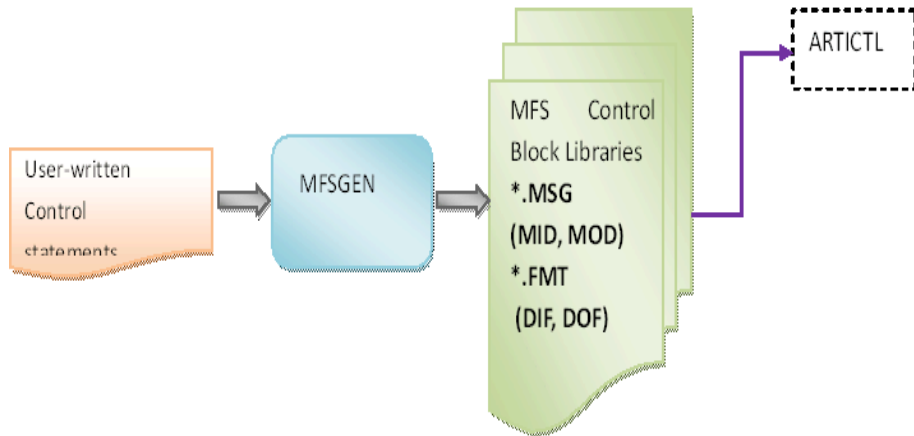
`MFSGEN` - Binary control blocks generator for ARTICTL. server

Description

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

[Figure 1](#) shows 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

Specifies 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 IMS_RT root

```
chgcobol.sh [mf|cit]
```

Description

You can have both MicroFocus and COBOL-IT installed at the same host, and can switch from one to the other or back. chgcobol.sh is used to switch between MicroFocus and COBOL-IT. To switch to COBOL runtime, you must first shutdown the ARTIMS application.

chgcobol.sh takes the following options:

Without any option

Shows 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

```

DFSRR00

Name

DFSRR00 - Utility used to activate the ARTIBMP or ARTIBMPT server.

Description

DFSRR00 is used to activate the ARTIBMP/ARTIBMPT server which waits for DFSRR00 input . The DFSRR00 parameter is a string passed from the script converted by workbench from JCL. Currently, only 5 sub-parameters contained in the string are supported : "BMP , \${MBR} , \${PSB} , \${IN} , \${CKPTID}" . The remaining sub-parameters in the string are ignored.

Note: 16 additional parameters are reserved for future use and, currently, are not supported.

Synopsis:

```
DFSRR00 "BMP,${MBR},${PSB},${IN},,,,,${CKPTID},,,,,,,,,,,,,,"
```

Parameter(s)

BMP
A hard-coded string.

\${MBR}
Represents the batch program to be called and it is required by both normal BMP and transaction oriented BMP.

\${PSB}
Represents the PSB name used for the program.

{IN}

Represents the transaction/queue name for a transaction oriented BMP program,

{CKPTID}

Represents the check point ID used in XRST.

Notes: When $\$ \{ \text{MBR} \}$ is empty, DFSRRCO0 exits with error directly.

When $\$ \{ \text{IN} \}$ is empty, the request is sent to ARTIBMP/ARTIBMP_ORA to call a BMP program.

When $\$ \{ \text{IN} \}$ is not empty, the request is sent to ARTIBMPT to call transaction oriented BMP program whose transaction code is $\$ \{ \text{IN} \}$.

prepro-ims.pl

Name

prepro-ims.pl - Utility used to transfer C program on z/OS to the format that could be run in ART IMS.

Description

prepro-ims.pl is used to transfer C programs on z/OS to a format that can be run in ART IMS. When file conversion fails, the failure information and lines of source file leading to failure are printed to the `stderr`. When complete, a summary is reported to `stdout`.

Processing rules are as follows:

- Delete

Comment the "#pragma runopts" line and other pragma directives.

The `env(IMS)` establishes the correct operating environment and `plist(IMS)` establishes the correct parameter list when invoked under IBM IMS. They are not necessary in ARTIMS Runtime and should be removed.

- Re-construct

Functions `ctdli()`/`aibtcli()` are reconstructed by adding additional trailing `NULL` to the argument list. This `NULL` is used to mark the end of the ARTIMS argument list. For `aibtcli()`, the `parmcoun` parameter is removed.

The function `main()` argument list is eliminated (including "argc", "argv" and "envp"). On mainframes, IMS uses a global list and a macro to define the list. On ARTIMS, the `__getcb(int)` function is implemented as `GET` method to obtain the PCB list.

The `exit()` function is renamed to `__art_ims_return()`. According to IBM IMS documentation, when there are no messages for the program to process, the program returns control to IMS by returning from `main` or by calling `exit()`. To avoid `exit()` unexpectedly exiting the container server, The `__art_ims_return()` function is used and helps to return control to container server.

Table 2 shows processing rules examples.

Table 2 Processing Rules Examples

Rules	Source	Destination
Delete	<code>#pragma runopts(env(IMS), plist(IMS))</code>	<code>/*#pragma runopts(env(IMS), plist(IMS))*/</code>
Re-construct	<code>ctdli(func_GU, io_pcb, msg_seg_io_area);</code>	<code>ctdli(func_GU, io_pcb, msg_seg_io_area, NULL);</code>
	<code>aibtdli(parmcount, func_GU, io_pcb, msg_seg_io_area);</code>	<code>aibtdli(func_GU, io_pcb, msg_seg_io_area, NULL);</code>
	<code>main(/* if any argument */)</code>	<code>main()</code>
	<code>exit(val)</code>	<code>__art_ims_return(val);</code>

Note: `prepro-ims.pl` cannot handle all generic C porting from mainframes to open systems.

Limitations

1. If one program requires multiple source files, the preprocessor performs simple processing on the source file containing the main entry. Users have to modify the `makefile` manually to add other depending C files.
2. Invoking `ctdli/aibtdli` as a function pointer is not supported.
3. The lines with valid a pragma symbol are commented out.
4. The directory of source file must be different from the destination source file.
5. If the input is a directory, its subdirectories will not be processed.

- Columns 73-80 containing sequence numbers are not supported. These columns should be replaced with blank spaces or left empty.

Dependencies

- `gmake` should be used for the generated makefile.
- Before conversion, you must ensure that the C source code is in UNIX format.
- Some header files provided by IMS, (e.g., `ims.h`), should be copied from mainframes to the open systems.

Synopsis:

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

Parameter(s)

`prepro-ims.pl` parameters, exit codes, and support modes are listed in [Table 3](#), [Table 4](#), and [Table 5r](#) respectively.

Table 3 prepro-ims.pl Parameters

Option name	Value range	Comment
<code>-i</code>	existing file(s) or directory	It's the original C program file[s] from IBM IMS application project. The suffix of file name must be <code>.c</code> or <code>.h</code> . As well, simple wildcards like <code>*</code> and <code>?</code> are supported.
<code>-o</code>	file or directory	It's the destination file or directory to be written. If the directory doesn't exist it will be created.
<code>-m</code>	file	It's the makefile generated for destination programs. It is placed in the destination directory. This file name should not be relative or absolute path.

Table 4 Exit Codes

Exit code	Meaning	Comment
0	Success	
1	Failure	general failure
2	Failure	invalid argument

Table 5 Support Modes

-i	-o	Support?
directory	directory	Y
dir1/file1	dir2/file2	Y
dir1/file1	dir2	Y
dir1/file1	dir3	Y
dir2/file2		Directories of source files must diff with dir3.
...		

The source program should look like that shown in [Listing 1](#).

Listing 1 Source Program

```

/* #pragma runopts(env(IMS), plist(IMS)) */
#include <ims.h>
#include <stdio.h>
#define n 20 /* I/O area size - Application dependent */
typedef struct {PCB_STRUCT(10)} PCB_10_TYPE;

int main()
{
static const char func_GU[4] = "GU ";
static const char func_ISRT[4] = "ISRT";

```

```
char ssa_name[] = "ORDER ORDER  (ORDERKEY  = 666666)";

int rc;
char msg_seg_io_area[n];
char db_seg_io_area[n];
char alt_msg_seg_out[n];

PCB_STRUCT_8_TYPE *alt_pcb;
PCB_10_TYPE *db_pcb;
IO_PCB_TYPE *io_pcb;

io_pcb = (IO_PCB_TYPE *)__pcblist[0];
alt_pcb = __pcblist[1];
db_pcb = (PCB_10_TYPE *)__pcblist[2];
..
/* get first message segment from message area */
rc = ctdli(func_GU, io_pcb, msg_seg_io_area, NULL);
..
/* get the data from the database having the specified key value */
rc = ctdli(func_GU, db_pcb, db_seg_io_area, ssa_name, NULL);
..
/* build output message in program's I/O area */
rc = ctdli(func_ISRT, alt_pcb, alt_msg_seg_out, NULL);
..
}
```

odbastop

Name

odbastop - Open system tool used to stop ODBA proxy on z/OS.

Description

Use odbastop on open systems to stop ODBA proxy running on z/OS.

Synopsis

```
odbastop -l host -p port -c cmd
```

Parameter(s)

host

The ODBA proxy hostname or ipv4 address.

port

The ODBA proxy port for receiving ODBA request.

cmd

Only supports SHUTDOWN

STOPROXY(z/OS)

Name

STOPROXY - Used to stop ODBA proxy on z/OS

Description

Use STOPROXY on z/OS to stop ODBA proxy running on z/OS.

Synopsis:

Modify `USER.ODBA.JCL(STOPROXY) JCL` and submit it to stop ODBA proxy. For more information, see Using ODBA PRoxy in the [Oracle Tuxedo Application Runtime for IMS Users Guide](#).

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:

- [Supported DL/I Interfaces](#)
- [Message Processing](#)
- [Database Operation](#)
- [Plug-in Definition for Different Implementation of IMS/DB](#)
- [Transaction Management](#)

Supported DL/I Interfaces

Table 6 lists supported DL/I interfaces.

Table 6 Supported DL/I Interfaces

Interfaces Name	Description
CBLTDLI	The 0 entry for DL/I calls in IMS/TM on OS/39
AIBTDLI	AIBTDLI is an entry function for the whole DL/I layer. The essential difference from CBLTDLI is that AIBTDLI uses AIB mask as the first parameter to pass control information in and out.
CTDLI	CTDLI is an entry function for the whole DLI library. Any request to DLI is passed by calling this function. CTDLI can only be called from C programs.

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

AIBTDLI

Name

AIBTDLI - an entry function for the whole DL/I layer.

Description

AIBTDLI is an entry for DL/I calls in IMS/TM on z/OS. In ARTIMS, AIBTDLI is a function acting as an entry of the DL/I library. AIBTDLI gets the PCB address according to the PCB name

specified in AIB mask, then calls appropriate function based on the function code passed to it with the PCB address found.

Parameter(s)

Function Call: e.g. 'GU '; AIB Mask, I/O Area: Input or Output Buffer. [Table 7](#) lists the AIB mask parameters.

Table 7 AIB Mask Parameters

Name	Value	Description	Support
AIBID	X(8)	AIB Identifier, it must be set to "DFSAIBbb". It's for input only	Yes
AIBLEN	X(4)	AIB Length, the length of the AIB mask. It must be set correctly in the application program and its minimum value is 128. It's for input only.	Yes
AIBSFUNC	X(8)	AIB subfunction, the subfunction required by some DL/I calls. It must be set correctly by application program if it's required.	Yes. In IMS, we only support INQY subfunction.
AIBRSNM1	X(8)	This field contains the PCB name. It must be specified by program correctly. For its setting, referring to AIBTDLI section.	Yes
RESERV1	X(16)	Reserved 16 bytes	No
AIBOALEN	X(4)	AIB Output Area Length, the length of the I/O area specified for AIBTDLI	Yes
AIBOAUSED	X(4)	AIB output area used length, the length of returned data in the I/O area. It's valid only in case the I/O area is used as output buffer.	Yes
RESERV2	X(12)	Reserved 12 bytes	No

Table 7 AIB Mask Parameters

Name	Value	Description	Support
AIBRETRN	X (4)	AIB Return Code, the return code of AIBTDLI call.	<p>Partial support.</p> <p>Currently, we only support setting this field in the following scenarios:</p> <ol style="list-style-type: none"> 1. INQY subfunction. 2. DLI call upon DBPCB(excludes GSAM database). <p>For other scenarios, if operation fails, this field is set to 0x0900, You must check the related PCB status.</p> <p>If operation is successful success, this field is set to 0x0000.</p>
AIBREASN	X (4)	AIB Reason Code, the reason code of AIBTDLI call.	<p>Partial support.</p> <p>Currently, we only support setting this field in the following scenarios:</p> <ol style="list-style-type: none"> 1. NQY subfunction. 2. DLI call upon DBPCB(excludes GSAM database). <p>If operation is successful success, this field is set to 0x0000.</p>
AIBERRXT	X (4)	AIB Error Code Extension, contains the additional error information	No

Table 7 AIB Mask Parameters

Name	Value	Description	Support
AIBRSA1	X(4)	AIB Resource Address, it's an output field to hold the PCB address corresponding to the PCB name specified in AIBRSNM1. In ART/IMS 64-bit, the PCB address is 8 bytes long, this field occupies actually 8 bytes, the other 4 coming from the first 4 bytes in the following reserved area.	Yes. But on 64bit platform, customer needs to use the AIBRSA1 plus the first 4 bytes in RESERV3 to get the right PCB's address.
RESERV3	X(48)	In ART/IMS 64-bit, its firstly 4 bytes are borrowed to store the returned PCB address in combination with AIBRSA1, so it can't be used for any other purpose.	No

The detailed rules for specifying PCB name in AIB mask field AIBRSNM1 is as follows:

I/O PCB

The name of I/O PCB must be specified as "IOPCBbbb",

Alternate PCB

The name of alternate PCB must be configured with "label=" in \$appname.psb configuration file, and must be specified correctly in AIB mask, the name (label) of each alternate PCB must be unique in a single PSB (i.e., in a single \$appname.psb file).

DB PCB

The name of DB PCB must be configured with "label=" in \$appname.psb configuration file, and must be specified correctly in AIB mask, the name (label) of each DB PCB must be unique in a single PSB (i.e., in a single \$appname.psb file).

CTDLI

Name

CTDLI - An entry function for the whole DL/I layer.

Description

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

Parameter(s)

Function Call: e.g. 'GU'; PCB: I/O PCB or Alternate PCB, I/O Area: Input or Output Buffer.

The `op` argument specifies the DL/I function to be performed. The `ctdli()` call format depends on the function selected. For more information, see [CBLTDLI](#).

Note: When using CTDLI, SSA parameter must be a pointer parameter.

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 in [Table 8](#).

Table 8 ARTIMS DL/I Processes and Commands

Name	Description
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
CMD	Sends/issues IMS commands, and retrieves the first segment of the response message.
GCMD	Retrieves the second and subsequent segments of the response message of a CMD command

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.

Synopsis

I/O PCB or AIB, I/O Area

Parameter(s)

I/O PCB

A pointer to the PCB that represents the source of the request

AIB

Specifies the application interface block (AIB) that is used for the call. This parameter is an input and output parameter. The following fields must be initialized in the AIB:

AIBID Eyecatcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the PCB name IOPCBbbb.

AIBOALEN I/O area length.

This field must contain the length of the I/O area that is specified in the call list.

I/O Area

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.

Synopsis

I/O PCB or AIB, I/O Area

Parameter(s)

SI/O PCB

A pointer to the PCB that represents the source of the request.

AIB

Specifies the application interface block (AIB) that is used for the call. This parameter is an input and output parameter. The following fields must be initialized in the AIB:

AIBID Eyecatcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the PCB name IOPCBbbb.

AIBOALEN I/O area length.

This field must contain the length of the I/O area that is specified in the call list.

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, 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 or AIB, I/O Area, MOD

Parameter(s)

I/O PCB or alternate PCB

Pointer to the PCB that represents the destination of the outgoing message

AIB

Specifies the application interface block (AIB) that is used for the call. This parameter is an input and output parameter. The following fields must be initialized in the AIB:

AIBID Eyecatcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the PCB name `IOPCBbbb` (if the TP PCB is used), or the name of an alternate PCB (if an alternate PCB is used)

AIBOALEN I/O area length.

This field must contain the length of the I/O area that is specified in the call list.

I/O Area

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

MOD

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.

If MOD equals DFS.EDT or DFS.EDTN, the message bypasses MFS formatting. When MFS is bypassed on output, the application program is responsible for constructing the entire 3270 data stream (beginning with the command code and ending with the last data byte).

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 or AIB, I/O Area (optional), MOD (optional)

Parameter(s)

I/O PCB or alternate PCB

Pointer to the PCB that represents the destination of the outgoing message.

AIB

Specifies the application interface block (AIB) that is used for the call. This parameter is an input and output parameter. The following fields must be initialized in the AIB:

AIBID Eyecatcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the PCB name IOPCBbbb (if the TP PCB is used), or the name of an alternate PCB (if an alternate PCB is used)

AIBOALEN I/O area length.

This field must contain the length of the I/O area that is specified in the call list.

I/O Area

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

MOD

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 trailing 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 or AIB, destination transaction code

Parameter(s)

I/O PCB

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).

AIB

Specifies the application interface block (AIB) that is used for the call. This parameter is an input and output parameter. The following fields must be initialized in the AIB:

AIBID Eyecatcher.

This 8-byte field must contain `DFSAIBbb`.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the name of a modifiable alternate PCB.

AIBOALEN I/O area length.

This field must contain the length of the I/O area that is specified in the call list

Result (Status Code):

‘bb’: successful (two blanks).

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

‘A1’: PCB is not valid.

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

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

CMD

Name

`CMD` - Used to enable a program to issue IMS commands.

Description

Sends/issues IMS commands, and retrieves the first segment of the response message.

CMD is used to issue IMS commands. It forwards the IMS command to an interface which can assumedly process all supported IMS commands. The CMD call waits for the interface to process IMS commands, and get the first field of the response message.

Restrictions

For commands, only "/DIS TRAN", "/DIS PGM" and "/DIS USER" are supported. Once these commands are issued by CMD API, related title segment will be returned through the I/O area, which describes the meaning of each field in subsequent segments.

- "/DISP TRAN", transaction related information that can be retrieved in Tuxedo is returned in I/O area. Returned segment is (excluding the llzz part): CUR_TRAN CUR_QCNT

- "/DISP user", user related information that can be retrieved in Tuxedo is returned. Returned segment is (excluding the llzz part): CUR_USER CUR_TRAN

- "/DISP PGM", program related information that can be retrieved in Tuxedo is returned. Returned segment is (excluding the llzz part): CUR_PGM CUR_TRAN

Note: Currently, we ignore all other parameters in above three commands. If other commands than above three supported ones are issued by "CMD" call, we return success status without any response segment.

Synopsis

I/O PCB or AIB, I/O Area

Parameter(s)

I/O PCB

Represents the source of the request

AIB

Specifies the application interface block (AIB) that is used for the call. This parameter is an input and output parameter. The following fields must be initialized in the AIB:

AIBID Eyecatcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the PCB name IOPCBbbb.

AIBOALEN I/O area length.

This field must contain the length of the I/O area that is specified in the call list

I/O Area

Pointer to a buffer, which contains the IMS command with its parameters, and to be filled in with the first segment. The general format of I/O area is:

LLZZ/verb KEYWORD1 P1 KEYWORD2 P2, P3. Comments

LL Two-byte field containing the length of the command text, including LLZZ

ZZ Two-byte field reserved for IMS

/ Indicates that an IMS command follows

verb The command you issued

KEYWORDx Keywords that apply to the command you issued

Px Parameters for the keywords you specified

. (period) End of the command

Result (status code):

'bb': successful (two blanks), but no response segments.

'CC': one or more response segments have been produced.

'AB': segment I/O area not specified

'CH': IMS ignored the CMD call just issued because the AOI command interface detected a system error and was unable to process the command. IMS processing continues.

GCMD

Name

GCMD - Retrieves the second and subsequent segments of the response message of a CMD command.

Description

GCMD retrieves the second and subsequent response segments from IMS TM when your application program processes IMS commands using the CMD call. Each returned segment contains the fields according to the title segments of above "CMD" Call. After the last segment has been retrieved, a GCMD call results in a "QD" status code returned in PCB.

Synopsis

I/O PCB or AIB, I/O Area

Parameter(s)

I/O PCB

Pointer to the PCB that represents the source of the request.

AIB

Specifies the application interface block (AIB) that is used for the call. This parameter is an input and output parameter. The following fields must be initialized in the AIB:

AIBID Eyecatcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the PCB name IOPCBbbb.

AIBOALEN I/O area length.

This field must contain the length of the I/O area that is specified in the call list

I/O Area

Pointer to a buffer to be filled in with the first segment.

Result (Status Code):

'bb': a segment was retrieved successfully (two blanks)

'AB': segment I/O area not specified

'QD': no more segments

'QE': GCMD request before CMD.

Database Operation

The DLI library performs the database operations issued from MPP or BMP programs. It can retrieve and hold one specific segment according the specified segment search criteria, updates a specific segment, inserts a segment at a specific position, deletes a specific segment, etc. Detailed APIs for database operation are listed in [Table 9](#).

Table 9 Database Operation Processes and Commands

Name	Description
GU/GHU	Retrieves (and Holds) the first segment that satisfies the criteria (if any) from the current position (if any) or the beginning of the database
GN/GHN	Retrieves (and Holds) the next segment that satisfies the criteria (if any) from current position
GNP/GHNP	Retrieves (and Holds) 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.
POS	A qualified Position (POS) call is used to retrieve the location of a specific sequential dependent segment. An unqualified POS points to the logical end of the sequential dependent segment (SDEP) data.
OPEN	Used to explicitly open a GSAM database
CLSE	Used to explicitly close a GSAM database

GU/GHU

Name

GU/GHU - Retrieves (and Holds) 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, the 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. GHU is not applicable for GSAM.

Synopsis

DB PCB, GSAM PCB or AIB, I/O Area, and SSA list (optional) or RSA (Mandatory for GSAM)

Parameter(s)

DB PCB

Contains all the DB related information, especially the DB name.

AIB

Specifies the AIB for the call. This parameter is an input and output parameter.

These fields must be initialized in the AIB:

AIBID Eye catcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the name of a DB PCB.

AIBOALEN I/O area length.

This field must contain the length of the I/O area specified in the call list

I/O Area

Used to receive the returned segment.

SSA:

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

RSA:

Specifies the area in your program that contains the record search argument. This required input parameter is only used for GSAM.

Note: The RSA "00010000" resets the position to the start of the GSAM.

GN/GHN

Name

GN/GHN - Retrieves (and Holds 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. GHN is not applicable for GSAM.

Synopsis

DB PCB,GSAM PCB or AIB, I/O Area, and SSA list (optional) or RSA(optional for GSAM)

Parameter(s)

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

RSA:

Specifies the area in your program where the RSA for the record should be returned. This output parameter is used for GSAM only and is optional.

GNP/GHNP

Name

GNP/GHNP - Retrieves (and Holds) 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.

Synopsis

DB PCB, GSAM PCB or AIB, I/O Area, and SSA list (optional)

Parameter(s)

The usage and restriction on parameters of GNP/GHNP are similar to [GU](#).

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.

Synopsis

DB PCB, GSAM PCB or AIB, I/O Area, and SSA list or RSA(optional for GSAM)

Parameter(s)

I/O

Area contains the segment to be added

AIB

Specifies the AIB for the call. This parameter is an input and output parameter. These fields must be initialized in the AIB:

AIBID Eye catcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the name of a DB PCB.

AIBOALEN I/O area length.

This field must contain the length of the I/O area specified in the call list

SSA

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.

RSA

Specifies the area in your program where the RSA should be returned by DL/I.

REPL**Name**

REPL - Used to update an existing segment.

Description

REPL is used to update an existing segment. you must first 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 cannot be changed.

Synopsis

DB PCB or AIB, I/O Area, and SSA list

Parameter(s)

The usage and restriction on parameters are similar to [GU](#).

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.

Synopsis

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

Parameter(s)

The usage and restriction on parameters are similar similar to [GU](#).

FLD

Name

FLD - Used to access and change a field within a segment.

Description

FLD call is used to access and change a field within a segment.

Synopsis

DB PCB or AIB, I/O Area, and SSA list

Parameter(s)

I/O

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

AIB

Specifies the AIB for the call. This parameter is an input and output parameter. These fields must be initialized in the AIB:

AIBID Eye catcher.

This 8-byte field must contain `DFSAIBbb`.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the name of a DB PCB.

AIBOALEN I/O area length.

This field must contain the length of the I/O area specified in the call list

SSA

Specifies the SSA used with this call. You can use up to 15 SSAs in this input parameter. The SSA that you supply will point to those data areas that you have defined for the call. This parameter is mandantory for the FLD call.

POS

POS - A qualified Position (POS) call is used to retrieve the location of a specific sequential dependent segment. An unqualified POS points to the logical end of the sequential dependent segment (SDEP) data.

Description

POS only supports DEDB. In ARTIMS, it has the following limitations:

1. keyword parameters are not supported.
2. The LL and corresponding numeric field (Field 4, Field 5) are stored in host byte endian

Synopsis

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

Parameter(s)

DB PCB

Contains all the DB related information, especially the DB name

AIB

Specifies the AIB for the call. This parameter is an input and output parameter. These fields must be initialized in the AIB:

AIBID Eye catcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the name of a DB PCB.

AIBOALEN I/O area length.

This field must contain the length of the I/O area specified in the call list I/O Area is used to received the returned output.

Because keywords are not supported, the output format of the I/O area is as following:

LL Field 1 Field 2 Field 4 Field 5

LL: This 2 bytes number, indicates the whole length.

Field 1? This 8-byte field contains the ddname from the AREA statement

Field 2: Sequential dependent next to allocate CI.

Field 4: This 4-byte field contains the number of unused control intervals in the sequential dependent part.

Field 5: This 4-byte field contains the number of unused control intervals in the independent overflow part

Note: The I/O data area will have 24 bytes of positioning information for every area in the DEDB.

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.

OPEN

Name

`OPEN` - Used to explicitly open a GSAM database.

Description

Explicitly opens a GSAM database. The following operation for the GSAM database will not open the GSAM database again. If you do not open GSAM database explicitly, other operations open the GSAM database implicitly.

Synopsis

`GSAM PCB or AIB, i/o area`

Parameters

GSAM PCB

The GSAM database corresponding PCB.

AIB

Specifies the AIB for the call. This parameter is an input and output parameter. These fields must be initialized in the AIB:

AIBID Eye catcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the PCB name of a GSAM PCB.

AIBOALENI/O area length.

This field must contain the length of the I/O area specified in the call list. I/O Area.

Specifies the kind of data set you are opening.

Note: This parameter is ignored. In Oracle Implementation for IMS/DB, we simulate GSAM database with local file system, so, I/O area is not used to specify the data set kind. We use the PROCOPT option in \$appname.psb to define whether the data set is to be read only or to be get and append.

CLSE

Name

CLSE - Used to explicitly close a GSAM database.

Description

Explicitly closes a GSAM database. If you do not close the GSAM database explicitly, IMS closes the GSAM database implicitly.

Synopsis

GSAM PCB or AIB

Parameters

GSAM PCB

The GSAM database's corresponding PCB.

AIB

Specifies the AIB for the call. This parameter is an input and output parameter. These fields must be initialized in the AIB:

AIBID Eye catcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the PCB name of a GSAM PCB.

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 2](#).

Listing 2 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

[Listing 3](#) shows the structure definition used for the `get_dbpcbattr` interface. The content is read from the PSB file and returned to application through `get_dbpcbattr` interface.

Listing 3 Structure Definition Used for the `get_dbpcbattr` Interface

```

enum PCBTYPE {IOPCB = 1, ALTPCB = 2, GSAMPCB = 3, DBPCB = 4};
enum SEQUENTIALBUFFERING {NO = 1, COND = 2};
enum PCBPOS {SINGLE = 1, MULTIPLE = 2};
enum SENSITIVITY {READ = 1, UPDATE = 2};
struct __SENFLD {
    char name[8];          /* mandatory, less than 8 filled with blank */
    unsigned short start; /* mandatory, range [1-32767] */
};

```

```

        int replace;          /* optional, default is 1 */
};

struct SSPTR {
    unsigned short pointer; /* range[1-8], default 0 */
    enum SENSITIVITY sens;
};

struct __SENSEG {
    char segname[8]; /* mandatory, less than 8 filled with blank */
    char parent[8]; /* mandatory, less than 8 filled with blank */
    char procopt[4]; /* optional, default filled with blank */
    SSPTR ssptr[8]; /* each slot contains a subset pointer number and
associated sensitivity, pointer of 0 indicates end, totally up to 8 can be
specified */
    char indices[8]; /* optional, default filled with blank */
    SENFLD * senfld; /* optional, default is NULL */
    unsigned short senfld_num; /* optional, default is 0, up to 255
SENFLD can be defined for each SENSEG */
};

struct __DB_PCB_ATTR { /* PCB Attributes */
    enum PCBTYPED type; /* mandatory */
    char dbname[8]; /* db name, default filled with blank */
    char pcbname[8]; /* pcb name, optional, default filled with blank */
    char procopt[4]; /* procopt , default filled with blank */
    enum SEQUENTIALBUFFERING sb; /* optional, default is NO */
    enum PCBPOS pos; /* optional, default is SINGLE */
};

```

```

int keylen; /* optional, default is invalid value 0 */
char procseq[8]; /* optional, default filled with blank */
int msdb_commit; /* optional, default is 0 */
int list; /* optional, default is 1 */

char *areas; /* area list set by SETR in DFSCTL, no change will be
applied on it in ART/IMS */

int senseg_num; /* optional, default is 0 */
struct SENSEG * senseg; /* optional, default is 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 db_entry(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

6. Get db pcb attributes

```
extern "C"
const __DB_PCB_ATTR * get_dbpcbattr (struct __DB_PCB * pcbm)
```

Functionality: Used to get additional attributes of db pcb. These attributes are configured in PSB file.

Argument:

pcbm: PCB Mask pointer

Return Value: PCB Attribute pointer, the user can't modify anything contained in the PCB attribute structure returned by this API; If the input is not DB PCB or GSAM PCB, null is returned; If any optional attribute is not configured, default value is returned; No need to free the pointer;

Where to use: 3rd party DB plugin when called with db pcb.

7. Fill db pcb segment name

```
extern "C"
int fill_dbpcb_segname (struct __DB_PCB * pcb)
```

Functionality: Used to get segment name of db pcb. This interface should be provided by DB plugin.

Argument::

pcb: PCB Mask pointer(DB PCB or GSAM PCB), segname in the pcb mask should be filled with correct value;

Return Value: 0 : success, other: failed.

Default Implementation for IMS/DB

Within ARTIMS, a default DLL is provided as an Oracle solution for IMS/DB.

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 10 lists the transaction management processes and commands.

Table 10 Transaction Management Processes and Commands

Name	Description
CHKP (Basic)	Used to set an explicit commit point.
CHKP (Symbolic)	Used to set an explicit commit point. Sets a check point from which the program can be started, saves as many as seven data areas in the program, and records current GSAM DB retrieval position.
ROLB	Used to cancel the database updates
SYNC	Commits the changes made by application programs.
INQY	Used to request information regarding execution environment, destination type and status, and session status.
XRST	Used to enable a program to start normally or to restart from a check point ID specified in a symbolic CHKP call.

CHKP (Basic)

Name

CHKP (Basic) - Used to set an explicit commit point.

Description

CHKP 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.

If the transaction is persistent transaction and handled by ARTIMPP in persistent mode or handled by ARTIBMPT, the next message will be retrieved from /Q of this transaction.

If the transaction is not persistent transaction, no next message is retrieved.

Parameter(s)

I/O PCB or AIB, I/O Area

I/O PCB

Pointer to the PCB that represents a destination.

AIB

Specifies the application interface block (AIB) that is used for the call. This parameter is an input and output parameter. The following fields must be initialized in the AIB:

AIBID Eye catcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the PCB name IOPCBbbb.

AIBOALEN I/O area length.

This field must contain the length of the I/O area that is specified in the call list.

I/O Area

Pointer to a buffer to receive the next input message. The area must be long enough to hold the longest message that can be returned.

Result (Status Code)

'bb': successful (two blanks).

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

'AB': no I/O area provided.

'QC': no input message.

'QF': segment less than 5 characters.

CHKP (Symbolic)

Name

CHKP (Symbolic) - Used to set an explicit commit point. Sets a check point from which the program can be started, saves as many as seven data areas in the program, and records current GSAM DB retrieval position.

Description

CHKP (Symbolic) can be used for recovery purposes. It commits all changes made by the program and, if your application program abends, establishes the point at which the program can be restarted. In addition, the symbolic CHKP call can:

- Work with the extended restart (XRST) call to restart your program if your program ends.
- Enables you to save as many as seven data areas in your program, which are restored when your program is restarted.

In ARTIMS, the simulated CHKP (Symbolic) is used to:

1. Commit the changes already made by using `tpcommit()`, which is the same as basic CHKP.
2. Retrieve the next message which is the same as basic CHKP.
3. Accept at most 7 data areas and save it with the check pint ID.

Note: If ART BMP server was restarted after user used symbolic CHKP to store data area, but before user uses XRST to restart program, the stored data area by symbolic CHKP will not be restored by XRST.

Once the environment variable "ART_IMS_CLEAN_DBSTATUS" is set to "Y" (or "y"), the status code of all related GSAM/DB PCB is set to blank after the CHKP (symbolic) call.

4. Record current retrieval position of all related GSAM DB.

Parameter(s)

I/O PCB or AIB, I/O Area, I/O Area Length, IO Area, area length, area, ...

I/O PCB

Pointer to the PCB that represents a destination.

AIB

Specifies the application interface block (AIB) that is used for the call. This parameter is an input and output parameter. The following fields must be initialized in the AIB:

AIBID Eye catcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the PCB name IOPCBbbb.

AIBOALEN I/O area length.

This field must contain the length of the I/O area that is specified in the call list.

I/O Area Length

No longer used. For compatibility reasons, this parameter must still be included in the call, and it must contain a valid address.

I/O Area

As an input parameter, used to specify the check point ID (8-bit). As an output parameter, pointer to a buffer to receive the next input message. The area must be long enough to hold the longest message that can be returned.

Area length

Specifies a 4-byte field in your program that contains the length in binary of the first area to checkpoint. This parameter is an input parameter. Up to seven area lengths can be specified. For each area length, you must also specify an area parameter.

Area

Specifies the area in your program that you want IMS to checkpoint. Always specify the area length parameter first, followed by the area parameter.

Result (Status Code)

'bb': successful (two blanks)

'AD': functional parameter invalid: function call not provided by ARTIMS

'AB': no I/O area provided

'QC': no input message

'QF': segment less than 5 characters

ROLB

Name

ROLB - Used to cancel the database updates.

Description

ROLB 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.

For persistent transaction (TP or transaction oriented BMP program), if it is handled by ARTIMPP in persistent mode or handled by ARTIBMPT, ROLB will return the first segment of the first message from last commit point to IOAREA, and delete first message from last commit point in /Q.

For non persistent transaction (TP or Transaction oriented batch), ART IMS will only return the first segment of the current handling message.

Synopsis

I/O PCB or AIB, I/O Area

Parameter(s)

I/O PCB

Pointer to the PCB that represents a destination.

AIB

Specifies the application interface block (AIB) that is used for the call. This parameter is an input and output parameter. The following fields must be initialized in the AIB:

AIBID Eye catcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the PCB name IOPCBbbb.

AIBOALEN I/O area length.

This field must contain the length of the I/O area that is specified in the call list.

I/O Area

Pointer to a buffer to receive the first segment of the returned message.

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.

'QE': GU is not called previously when IOAREA is not NULL.

'QC': no input message.

'QF': segment less than 5 characters.

ROLL

Name

ROLL - Used to cancel the database updates and return to ART IMS.

Description

ROLL 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 ROLL 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, then it return control to ART IMS but don't return control to the calling program.

For persistent transaction (TP or transaction oriented BMP program), if it is handled by ARTIMPP in persistent mode or handled by ARTIBMPT, ROLL deletes the current message from last commit point in /Q.

Synopsis

ROLL

Parameter(s)

The only parameter required for the ROLL call is the call function.

Result (Status Code):

No returned status codes.

SYNC

Name

SYNC - Used to commit the changes made by application programs (normally database updates).

Description

In ARTIMS, the simulated SYNC is used to commit the changes already made, not to establish places in your program where you can restart, if your program terminates abnormally.

Synopsis

I/O PCB or AIB

Parameter(s)

I/O PCB

Pointer to the PCB that represents a destination.

AIB

Specifies the application interface block (AIB) that is used for the call. This parameter is an input and output parameter. The following fields must be initialized in the AIB:

AIBID Eye catcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the PCB name IOPCBbbb

INQY

Name

INQY-Used to request information regarding execution environment, destination type and status, and session status. INQY is valid only when using the AIBTDLI interface.

Description

In IMS, only the following subfunctions are supported: NULL, "FINDbbbb", "PROGRAMb", "DBQUERYb".

For NULL subfunction, ART/IMS can only return the PCB related information that ART/IMS can support into the I/O area. "Terminal Location" and "Transaction Location" are only supported using "LOCAL".

For "FINDbbbb", the PCB address is returned in the AIBRSA1 field.

On 64-bit platform, the first 4 bytes of AIBRES3 is also used because the address length is 8 bytes.

Subfunction "PROGRAMb" returns the program name in the first 8 bytes of the I/O area.

For "DBQUERYb" subfunction, if there is no DBPCB in defined in PSB, "BJ" is returned in the IO PCB status. Otherwise, the IO PCB status is returned according to database availability.

Synopsis

AIB I/O Area

Parameters

AIB

Specifies the address of the application interface block (AIB) that is used for the call. This parameter is an input and output parameter. The following fields must be initialized in the AIB:

AIBID Eye catcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBSFUNC Subfunction code.

This field must contain one of the 8-byte subfunction codes as follows:

```
bbbbbbbbb (Null)  
DBQUERYb  
FINDbbbb  
PROGRAMb
```

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the PCB name of any PCB named in the PSB.

AIBOALEN I/O area length.

This field must contain the length of the I/O area that is specified in the call list. This field is not changed by IMS.

I/O Area

Used to received the returned output.

Result in AIB:

Return code and Reason code is aligned to IBM's return code and reason code description. For "DBQUERYb", The database availability status is in the IOPCB status.

'bb' (two blanks): The call is successful and all databases are available.

'BJ': No DB PCB exists in the PSB. Or None of the databases in the PSB are available, or no PCBs exist in the PSB. All database PCBs (excluding GSAM) contain an NA status code as the result of processing the INQY DBQUERY call.

'BK' At least one of the databases in the PSB is not available or availability is limited. At least one database PCB contains an NA or NU status code as the result of processing the INQY DBQUERY call.

In IMS, after the "DBQUERYb" call, status codes in each DB PCB should not be used to check DB status.

XRST

Name

XRST - Used to restart your program. If you use the symbolic Checkpoint call in your program, you must use the **XRST** call.

Description

XRST is used to restart your program. If you use the symbolic Checkpoint call in your program, you must use the **XRST** call.

In ARTIMS, the simulated XRST is used to recover the data that was saved in the related CHKP (symbolic) call. The GSAM is repositioned to the recorded position where the CHKP (symbolic) call occurs so that all subsequent “GN” calls continue with the recovered position.

Once the environment variable “ART_IMS_CLEAN_DBSTATUS” is set to “Y” (or “y”), the status code of all related GSAM/DB PCB is set to blank after the XRST call (with an existing CHKPID).

Parameter(s)

I/O PCB or AIB, I/O Area, I/O Area Length, IO Area, area length, area, ...

I/O PCB

Pointer to the PCB that represents a destination

AIB

Specifies the application interface block (AIB) that is used for the call. This parameter is an input and output parameter.

The following fields must be initialized in the AIB:

AIBID Eye catcher.

This 8-byte field must contain DFSAIBbb.

AIBLEN AIB lengths.

This field must contain the actual length of the AIB that the application program obtained.

AIBRSNM1 Resource name.

This 8-byte, left-justified field must contain the PCB name IOPCBbbb.

AIBOALEN I/O area length.

This field must contain the length of the I/O area that is specified in the call list. This parameter is not used during the XRST call. For compatibility reasons, this parameter must still be coded.

I/O Area Length

No longer used. For compatibility reasons, this parameter must still be included in the call, and it must contain a valid address.

I/O Area

Used to specify the check point ID from which the program should be restarted. If the program is to start normally, the first 5 characters of I/O area must be blanks.

Area length

Specifies a 4-byte field in your program that contains the length in binary of the first area to checkpoint. This parameter is an input parameter. For each area length, you must also specify an area parameter. The number of areas you specify on a XRST call must be less than or equal to the number of areas you specify on the CHKP calls the program issues.

Area

Specifies the area in your program that you want IMS TM to restore. Always specify the area length parameter first, followed by the area parameter.

Result (Status Code)

'bb': successful (two blanks)

'AD': functional parameter invalid: function call not provided by ARTIMS

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 11](#) shows all the definition statements and their descriptions.

Table 11 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 11 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 12](#) and [Table 13](#). All other statements are assumed unsupported fields currently (listed in [Table 14](#)).

Table 12 Message Definition Statement Set Fields

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=	msgcontrolbl ockname	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=	msgcontrolbl ockname	
	PROMPT=	(dfldname, 'l iteral')	
	PASSWORD	PASSWORD	
		blanks comments	Error

Table 12 Message Definition Statement Set Fields

Statement Name	Field	Possible Value	Note
SEG	EXIT=	(exitnum,exitvect)	Warning
	GRAPHIC=	YES or NO	Warning
DO	count		Support
	SUF=	number	Support

Table 12 Message Definition Statement Set Fields

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

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

For LTMSG and LPAGENO, a warning message is displayed and will not take any effect. For other strings not in [Table 12](#), a syntax error. is displayed

Table 13 Format Definition Statement Set Fields

Statement Name	Field	Possible Value	Support or Other
FMT			Support

Table 13 Format Definition Statement Set Fields

Statement Name	Field	Possible Value	Support or Other
DEV		3270	Support
	TYPE=	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=	dflename	Warning
	CARD=	dflename	Warning
	SYMSG=	dflename	Support
	DSCA=	X'value'	Support
		number	Support
		(dflename, 'literal'...)	Support
	PFK=	(dflename, integer='literal'...)	Support
		(dflename, NEXTTP NEXTMSG NEXTMSGP NEXTLP ENDMPPI ...)	Warning
		(dflename, integer= NEXTTP NEXTMSG NEXTMSGP NEXTLP ENDMPPI ...)	Warning
	SUB=	X'hh'	Warning
		C'c'	Warning
PDB=	pdbname	Warning	
DIV	TYPE=	INOUT	Support
		OUTPUT	

Table 13 Format Definition Statement Set Fields

Statement Name	Field	Possible Value	Support or Other	
DPAGE	CURSOR=	(111, 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 13 Format Definition Statement Set Fields

Statement Name	Field	Possible Value	Support or Other	
DFLD	'literal'		Support	
	G'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 13 Format Definition Statement Set Fields

Statement Name	Field	Possible Value	Support or Other
		CD BLUE RED PINK GREEN TURQ YELLOW NEUTRAL	Support
		PX'00' PX'hh' PC'c'	Warning
		EGCS EGCS'hh'	Support, For EGCS'hh', only 'hh' value of X'00' and X'F8' are supported
		VDFLD VMFILL, VMFLD VMFILL VMFLD	Warning
		OUTL OUTL'hh' BOX RIGHT LEFT UNDER OVER	Support
		MIX MIXD	Support
ENDDO			Support
FMTEND			Support

Table 14 Other definition statements and compilation statements which we don't support

Statement Name	Fields	Support or other
END		Support

Table 14 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 14 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 15 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	Non-displayable	Yes
1	6	Detectable	Yes
1	7	Premodified	Yes

Table 16 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 16 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 17 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

ARTIMS Non-Terminal Access Support

To support the Non terminal access to service exported by MPP, and to enable more client to utilize service exported by MPP, this feature enhances ARTIMS to support Non terminal access. User can utilize the ARTIMS MPP service via Non- terminal applications, such as native Oracle Tuxedo client, SALT client and JCA client.

This feature support both non-terminal tuxedo clients (including native Tuxedo client, SALT client, JCA client) and MQ application.

For non-terminal tuxedo clients, programing interface is provided. Client can access the MPP service by following the programing interface.

Different from non-terminal Oracle Tuxedo clients, a WebSphere MQ application is not an Oracle Tuxedo client. The MQ message format and message flow between MQ applications and traditional IMS applications is already defined in the WebSphere MQ Application Programming Guide. In an IMS environment, MQ application utilizes MQ-IMS Bridge to enable implicit MQI support so that tradition IMS applications can be accessed by WebSphere MQ messages, without having to rewrite, recompile, or re-link them. This feature leverages the Oracle Tuxedo MQ Adapter to convert the MQ application to an Oracle Tuxedo client.

Programming Interface

Programming Interface for Non-terminal Oracle Tuxedo Clients

ARTIMS provides a server, ARTIGW, working as a bridge between Non-terminal clients and the ARTIMS MPP server. A non-terminal client calls the ARTIGW service following the programming interface list below. ARTIGW forwards the service request to ARTIMPP.

The only interface between ARTIGW and Non terminal Oracle Tuxedo clients is an FML table.

To run an IMS transaction (for example, TRANS1) with application buffer, do the following steps:

1. The client user must prepare the send buffer containing these FML fields:

IMS_SVC_NAME

IMS transaction name (i.e., "TRANS1").

IMS_SVC_FLAG

Reserved for future use.

IMS_SEG_DATA

Application buffer data. LLZZ is not expected in the buffer. The maximum segment length is 32764 (which is the ARTIMPP limit).

2. Client issues a `tpcall()/tpacall()` with the buffer prepared in step 1.

```
ret = tpcall(< tuxclt_service_name>, ...)
```

Here `< tuxclt_service_name>` is the ARTIGW advertised service; the service name is configurable. For more information, see ARTIGW CONFIGURATION.

3. Client gets the reply. In the reply message, the following FML fields are present:

IMS_SVC_RESULT

0: ARTIMPP processes the request successfully with a response message.

1: ARTIMPP processes the request successfully without a response message.

-1: ARTIGW error

-2: ARTIMPP error

IMS_SEG_DATA

Buffer contains reply data. LLZZ is not included in the buffer.

IMS_SVC_SYSMSG

Verbose error message if IMS_SVC_RESULT is a negative integer.

The interface FML Fields table (ARTIGWFML) and header file (ARTIGWFML.h) can be found under \$IMSDIR/include. Listing 4 shows the ARTIGWFML contents.

Listing 4 ARTIGWFML Contents

```
*base 30000700

#name          rel-number   type      flags      comment
#-----      -
IMS_SVC_NAME   181          string
IMS_SVC_FLAG   182          long
IMS_SVC_RESULT 183          long
IMS_SEG_DATA   184          carray
IMS_SVC_SYMSG  185          carray
```

Supported MQ Messages for MQ Applications

The MQ-ARTIMS Bridge accepts the following message types:

- Messages containing IMS transaction data and an MQIIH structure:

```
MQIIH LLZZ<trancode><data>[LLZZ<data>][LLZZ<data>]
```

- Messages containing IMS transaction data but no MQIIH structure:

```
LLZZ<trancode><data>[LLZZ<data>][LLZZ<data>]
```

Notes:

1. The square brackets, [], represent optional multi-segments.
2. If message contains MQIIH structure, the MQMD structure Format field is set to MQFMT_IMS.
3. If message does not contain MQIIH structure, the MQMD structure Format field is set to MQFMT_IMS_VAR_STRING.

Configuration

Oracle Tuxedo MQ Adapter Configuration

To support MQ application, user must apply Tuxedo 12cR1 RP17 or later and follow the instruction list below:

1. In the Oracle Tuxedo MQ Adapter configuration file, the `TM_MQI *SERVICE` sections must be defined as follows:

For message with MQIHH structure:

```
*SERVICE
  NAME=< mq_service_name >
  FORMAT= MQIMS
  TRAN = N
```

For message without MQIHH structure:

```
*SERVICE
  NAME=< mq_service_name>
  FORMAT= MQIMSVS
  TRAN = N
```

Here `< mq_service_name>` is the ARTIGW advertised service; the service name is configurable. For more information, see [ARTIGW Configuration](#).

2. In the `TM_MQI` configuration file `*SERVER` section, the following parameters must be specified.

```
*SERVER
  TPESVCFAILDATA=Y
  REPLYONSVCCERR=Y
  MSGTYPEONTPFAIL=Y
  IMPORTMQMD=Y
```

3. In the `UBBCONFIG` file, `TM_MQI` server must be in a Group configured with TMS Server for the WebSphere MQ Resource Manager.

ARTIGW Configuration

ARTIGW is a Tuxedo server working as a bridge between non-terminal client and the ARTIMPP server. For more information, see [Server Configurations](#).

Cross Domain Configuration

If ARTIGW and ARTIMPP are deployed in different domains, ARTIGW exports services named as `<tuxclt_service_name>_REPLY_<grpId>_<srvid>` and `<mq_service_name>_REPLY_<grpId>_<srvid>`. The GRPID and SVRID are 5 characters (starting with 0).

The service names mentioned above should be configured in the `DM_REMOTE_SERVICES` section of the `DMCONFIG` file of every remote domain which ARTIMPP belongs to. In addition, make sure correct service names are exported by each domain where ARTIGW lives, and make sure there are no service conflicts.

For example, assume that ARTIGW is in domain GW and ARTIMPP is in domain MPP. ARTIGW is configured to use default service name with `SRVID=101`, `SRVGRP= GROUP1`. [Listing 5](#) shows an example of the `DMCONFIG` files for MPP and GW domains.

Listing 5 MPP and GW Domains DMCONFIG Files

- MPP domain:

```
*DM_REMOTE_SERVICES
IMSGW_SVC_REPLY_00001_00101
```

- GW domain:

```
*DM_LOCAL_SERVICES
IMSGW_SVC_REPLY_00001_00101
```

Notes:

1. No data conversion is done by the ARTIGW for non-terminal tuxedo clients. The data provided by the client application should be in the format expected by ARTMPP server and application programs.
2. ARTIGW is a single thread Tuxedo server. User can deploy multiple instance of ARTIGW for performance tuning.

3. The Oracle Tuxedo MQ Adapter is the MQI interface with the MQ application. Due to the different behavior of WebSphere MQ on mainframe and open system, the Tuxedo MQ Adapter behavior may look different either.

For example, if an MQ application puts a message with the `MQPMO_NONE` option on a Mainframe, the Oracle Tuxedo MQ Adapter will not trigger an IMS transaction until `MQCMIT` is called by the MQ application.

Note: On open systems, IMS transaction is triggered immediately.

4. For MQ Applications, if an unexpected event occurs with `ARTIGW` or `ARTIMPP` during message processing, a report message is generated and sent to the reply queue specified by original message. The report message contains no data from the original message, only a string containing the detailed error message

Limitations

Non-Terminal Oracle Tuxedo Client Limitations

1. Global transaction is not supported. If a `tpcall` to the `ARTIGW` is in the global transaction, the `TPNOTRAN` flag must be set.
2. If there is an access control violation when accessing an `ARTIMPP` service, a non-terminal client will not get an error message like "Access control violation" immediately, but waits until a time out occurs. You must check `ULOG` to get a detailed error report.

MQ Application Limitations

1. Only Commit mode 0 (`COMMIT_THEN_SEND`) is supported. `ARTIMS` will always handle the transaction as Commit mode 0 no matter what the Commit mode value is set in `MQI IH`.
2. Authentication with the RACF password or `passticket` is not supported.
3. MFS transaction is not supported.
4. Conversation is not supported.
5. Transaction code specified in the input message cannot be a command.
6. When Tuxedo MQ Adapter detects an access control violation, it will put original message into Dead Letter Queue. There will be no report message in the reply queue. You can also find the error report in the `ULOG` file.

ARTIMS Persistent Message Support

ART IMS supports persistent message only for programs using `ALT PCB`. For the program switch via `ALT PCB`, when the target transaction is a persistent transaction, the message will be put into the `/Q` of that transaction. For how to define a persistent transaction, please refer to the `imsresource.desc` section.

For persistent message support, we differentiate the `ARTIMPP` into two running mode. One is `ARTIMPP` in normal mode, one is `ARTIMPP` in persistent mode. For how to define the two `ARTIMPP` server mode, please refer to `ARTIMPP` configuration section.

`ARTIMPP` in normal mode will advertise transactions as services and handle the normal service request from front end, such as terminal request, request from `ARTIGW`. The service request is scheduled by Tuxedo framework and is passed via Tuxedo IPC queue.

`ARTIMPP` in persistent mode will not advertise any services. It will monitor every `/Q` for the persistent transaction whose `CLASS` belong meet the "`-l class_list`" parameter of the `ARTIMPP`. It gets the message from the `/Q` of the persistent transaction and execute corresponding program for the transaction.

`ARTIBMPT` is also a server that is related to persistent message support. The transaction oriented `BMP` program that `ARTIBMPT` serves must only be persistent transaction. For what is "transaction oriented `BMP`", please refer to `ARTIBMPT` configuration section. In transaction oriented `BMP` program, the `GU` operation will get message from the `/Q` of the transaction.

Server Configurations

[Table 18](#) Lists the server configuration processes and commands.

Table 18 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/C programs
ARTIMPP_ORA	Same as <code>ARTIMPP</code> . However, it requires the Oracle database as <code>RM</code>
ARTIBMP	Program container for BATCH type COBOL/C programs.
ARTIBMP_ORA	Same as <code>ARTIBMP</code> . However, it requires the Oracle database as <code>RM</code> .

Table 18 Server Configuration Processes and Commands (Continued)

Name	Description
ARTIBMPT	An Oracle Tuxedo server that handles transaction oriented batch programs.
ARTIADM	An Oracle Tuxedo server responsible for the administration of ART IMS Runtime.
ARTITERM	Acts as messenger between ARTICTL and ARTIMPP located in different domains.
ARTIGW	An Oracle Tuxedo server that acts as a bridge between non-terminal clients and the ARTIMPP server.
ODBAPROX	A socket server on z/OS and is to communicate with ARTIMS servers through TCP/IP.

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 access 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]"
```


Parameter(s)

-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.

Example(s)

```
*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/C programs.

Description

ARTIMPP is a Tuxedo server to act as both service handler and container for COBOL/C programs of TP type. There are two running modes for ARTIMPP, one is ARTIMPP in normal mode (without `-p` in CLOPT), another is ARTIMPP in persistent mode (with `-p` in CLOPT).

ARTIMPP in normal mode invokes corresponding COBOL/C program according to the service request received from front end (request from terminal, ARTIGW).

ARTIMPP in persistent mode will monitor /Q for persistent transaction whose class is defined in the `"-l class_list"` parameter in CLOPT for the ARTIMPP. It will get the message from the /Q and invoke corresponding COBOL/C program.

Synopsis:

```
ARTIMPP SRVGRP="identifier"
          SRVID="number"
CLOPT="[servopts options] -- -l class_list[-p][-x parameter list for DB
plugin]"
```

Parameter(s)

[-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 in normal mode.

-p

ARTIMPP is in persistent mode. ARTIMPP in persistent mode will not advertise any services/transactions.

[-x]

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

For the Oracle IMS/DB solution, the parameter list is as follows:

```
-o host:port:dra
```

This is the parameter required by Oracle plug-in for IMS/DB.

```
host      Hostname or ipv4 address of ODBA proxy to connect to.
port      Port of ODBA proxy for receiving ODBA request.
dra       Name of the DRA table in which the IMS/DB system to be accessed is defined.
          E.g., CLOPT="-A -- -x -o zosmachine:1234:BEA1"
```

Notes: When ARTIMPP server is configured in persistent mode in the `UBBCONFIG` file and persistent transaction is configured in `imsresource.desc`, `TMQQUEUE` server must also be configured (before ARTIMPP), in the `UBBCONFIG` file according to the `/Q` configuration in `imsresource.desc`.

Before starting ART IMS, you must also create `/Q` according to the information in `imsresource.desc`.

Limitation(s)

For a persistent transaction that may issue `ROLB` or `ROLL`, the transaction could only be handled by one ARTIMPP in persistent mode server. That is, the `CLASS` definition of the transaction (defined in `imstrans.desc`) could only match one ARTIMPP in persistent mode server `-1` parameter.

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/C programs.

Description

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

Synopsis:

```
ARTIBMP SRVGRP="identifier"  
                SRVID="number"  
CLOPT="[servopts options] -- [-x parameter list for DB plugin]"
```

Parameter(s)

[-x]

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

For the Oracle IMS/DB solution, the parameter list is as follows:

```
-o host:port:dra
```

This is the parameter required by Oracle plug-in for IMS/DB.

host

Hostname or ipv4 address of ODBA proxy to connect to.

port

Port of ODBA proxy for receiving ODBA request.

dra

Name of the DRA table in which the IMS/DB system to be accessed is defined.

E.g., `CLOPT="-A -- -x -o zosmachine:1234:BEA1"`

ARTIBMPT

Name

ARTIBMPT - An OracleTuxedo server that handles transaction oriented batch programs.

Description

Transaction oriented BMP program should be served by ARTIBMPT and triggered via DFSRRC00 with `${IN}` parameter.

A transaction oriented BMP program is a program that is defined in `imstrans.desc` and also defined in `imsapps.desc` with a `TYPE=BATCH`.

The transaction oriented BMP program must be persistent transaction, it must be defined in `imsresource.desc`.

ARTIBMPT server only handle the transaction oriented BMP programs.

Synopsis

```
ARTIMPP SRVGRP="identifier" SRVID="number"
CLOPT="[servopts options] -- -l class_list -x parameter list for DB plugin"
```

Parameter(s)

[-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 ARTIBMPT.

For the Oracle IMS/DB solution, the parameter list is as follows:

```
-o host:port:dra
```

This is the parameter required by Oracle plug-in for IMS/DB.

`host`

Hostname or ipv4 address of ODBA proxy to connect to.

`port`

Port of ODBA proxy for receiving ODBA request.

`dra`

Table name which specifies the IMS/DB to connect to; this parameter is optional; if it's configured, all the DB operations are performed through ODBA, otherwise all DB operations are through actual DB implementation on open systems. The DRA table name must be uppercase and 4 bytes long.

Note: When ARTIBMPT is configured in `UBBCONFIG` file and there is transaction oriented BMP transaction which is configured in `imsresource.desc`, `TMQUEUE` server must be also configured in `UBBCONFIG` file according to the /Q'configuration in `imsresource.desc`.

Limitation(s)

For a transaction oriented BMP program that may issue ROLB or ROLL, the program could only be handled by one ARTIBMPT server.

That is, the CLASS definition of the transaction (defined in imstrans.desc) could only match one ARTIBMPT server "-l class_list" parameter.

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 - An Oracle 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=" "
```

ARTIGW

Name

ARTIGW - An Oracle Tuxedo server working as a bridge between non-terminal client and the ARTIMPP server.

Description

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.

- Session Management.

A session management is used to associate the asynchronous reply from ARTIMPP with original request from clients of ARTIGW.

Synopsis:

```
ARTIGWSRVGRP="identifier"  
SRVID="number"  
CLOPT="[servopts options] [-m mq_service_name] [-s tuxclt_service_name] [-D  
trace_level]"
```

Parameter(s)

mq_service_name

Specifies the service name advertised by server, which is dedicated for message handling for MQ application. It is an optional parameter, the default service name <IMSGW_MQ_SVC> is used if this parameter is not present.

tuxclt_service_name

specifies the service name advertised by server, which is dedicated for message handling for non-terminal tuxedo client. It is an optional parameter, the default service name <IMSGW_SVC> is used if this parameter is not present.

[-D trace_level]

Enables debug mode, trace is located at \$APPDIR/log. Currently, the value of trace_level can be 0, 1 or 2. There are no significant differences with these values in this feature.

Note: Both the mq_service_name and tuxclt_service_name cannot begin with "<domainid>_". Otherwise ARTIGW cannot get the response message correctly. Here <domainid> is the ID of the domain which ARTIMPP belongs to

ODBAPROX

Name

ODBAPROX - is a socket server on z/OS and is to communicate with ARTIMS servers through TCP/IP.

Description

ODBA proxy is a socket server to communicate with ARTIMS servers through TCP/IP. ODBAPROX is developed to communicate with the programs in ARTIMS and perform database operations on behalf of these programs

Synopsis:

```
ODBAPROX -h host -l command_port -p odba_port -n max_handler_num [-D]
```

Parameters:

- [-h]** Specifies the host (hostname or ipv4 address) where the ODBA proxy is running.
- [-l]** Specifies the port for receiving command from external utility.
- [-p]** Specifies the port for receiving odba request.
- [-n]** Specifies the maximum number of handlers to be started.
- [-D]** Enables debugging info be printed in stdout

For more information, see Using ODBA Proxy in the [Oracle Tuxedo Application Runtime for IMS Users Guide](#).

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 tpsur. Client name is not used. The checking of per-user password against the password and user name in tpsur 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**

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 is 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.

- **INTERCODE**

Specifies which encoding type of inbound data will be used. The value could be any encoding type used in universal platform. If this variable is not specified, ASCII will be used. If this variable is specified, then `EXTERCODE` should be specified as well.

- `EXTERCODE`

Specified which encoding type of outbound data will be used. The value could be any EBCDIC encoding type used in z/OS platform. If this variable is not specified, IBM-37 will be used. If this variable is specified, then `INTERCODE` should be specified as well.

- `ART_IMS_CLEAN_DBSTATUS`

Used to enable the setting for `GSAM/DB PCB` status codes. If set to “Y” or “y”, the `DLI` calls for `CHKP` (symbolic) and `XRST` (with an existing `CHKPID`, not normally start) resets all `GSAM/DB PCB` status codes to blanks; otherwise, `ART IMS` will not reset these status codes.

Commands and Parameters

[Table 19](#) lists the commands and associated parameters that can be input on 3270 terminal and be processed by `ARTIMS`.

Table 19 3270 Terminal Commands and Parameters

Table 20 Command	Table 21 Shortening	Table 22 Parameter
<code>/EXIT</code>	<code>/EXI</code>	None
<code>/Format</code>	<code>/Forma ,</code> <code>/Form ,</code> <code>/For</code>	modname
<code>/Sign</code>	<code>/Sig</code>	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](#)
- [Persistent Transaction Definition - imsresource.desc](#)
- [Database Definition - imsdbs.desc](#)
- [PSB Definition - \\$appname.psb](#)
- [Segments Definition - segments.desc](#)
- [Segment Definition - \\$segname.desc](#)

The general format for configuration files is as follows.

Listing 6 General Configuration File Format

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

Transaction Definition - imstrans.desc

[Table 23](#) shows an example configuration file with field names mapped from TRANSACT MACRO of IMS on z/OS

Table 23 Section Name: [imstran]

Field	Type	Value	Description	Field in TRANSCAT
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

Table 24 shows an example configuration file with field names mapped from APPLCTN MACRO of IMS on z/OS.

Table 24 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 and BMPT, default is TP	PGMTYPE
LANG	Enum	COBOL C	COBOL for cobol program, C for c program, default is COBOL	LANG

Persistent Transaction Definition - imsresource.desc

Every transaction that is defined in this file is a persistent transaction. Messages for the persistent transaction via program switch is put into /Q. You must create the /Q for every persistent transaction in corresponding queue space configured in this file with the queue name equal to the transaction name before ART IMS is booted. [Table 25](#) list the `imsresource.desc` field names.

Table 25 Section Name: [imsresource]

Field	Type	Value	Description	Field in TRANSACT
QSPACE	X(1..16)	Mandatory	Qspace name	
TRANNAMES	X(1..1024)	Mandatory	The persistent transaction names. The transaction name could be separated by comma, e.g: TRAN1, TRAN2. Every transaction name could not exceeds 8 bytes.	N/A

[Listing 7](#) shows a script example to create queue space and queues for your reference; you can customize the script to adapt to the real requirement. For more information, see Oracle Tuxedo /Q guides.

Listing 7 Create Queue Space Example Script

```
qmadmin ${ARTIMS QSPACE_DEVICE} <<!end
crdl . . .
qspacecreate . . .
qcreate TRAN11 fifo none 2 30 80% 0 ""
qcreate TRAN12 fifo none 2 30 80% 0 ""
!end
--->
qmadmin ${ARTIMS QSPACE_DEVICE} <<!
echo
```

```

crdl ${ARTIMS QSPACE_DEVICE} 0 10000
qspacecreate
${QUEUE_SPACENAME}
22839
5000
50
1000
1000
10000
errque
Y
16
qopen ${QUEUE_SPACENAME}
qcreate TRAN11 fifo none 2 30 80% 0% ""
qcreate TRAN12 fifo none 2 30 80% 0% ""
qcreate errque fifo none 2 30 80% 0% ""
q
!
```

For persistent transactions that issue ROLB or ROLL, the retry count for creating the transaction queue should be set to a relatively large number (up to 2147483647).

Note: It is suggested that you set the retry count to a number that is more than twice the number that a particular message enters a persistent transaction.

Database Definition - imsdbs.desc

imsdbs.desc is located under \$ART_IMS_CONFIG.

Some imsdbs.desc field configurations are mapped from some DBD statement of IMS on z/OS. For persistent transactions that issue ROLB or ROLL, the retry count for creating the transaction queue should be set to a relatively large number (the largest number could up to 2147483647).

Note: It is suggested that the retry count is set to a number that is more than twice the number that a particular message enters a persistent transaction.

Table 26 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 PSINDE X 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. If this name is defined in JCL with the same DD name, the real input file is decided by the DSNNAME associated with the DD name in JCL. Otherwise, the real input file is \$ART_IMS_DB/Input file name.	N/A
DD2	X(1..8)	Literal String	Output File Name.If this name is defined in JCL with the same DD name, the real output file is decided by the DSNNAME associated with the DD name in JCL. Or else, the real output file is \$ART_IMS_DB/Output File Name.	N/A
MINRECORD	Integer	Positive Integer	Minimum record length if RECFM=V, its value range is 1~268435455.	N/A
MAXRECORD	Integer	Positive Integer	Maximum record length if RECFM=V, its value range is 1~268435455.	N/A

Table 26 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, 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 imsappls.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 DFSRRCO0.

Table 27 shows an example configuration file with field names mapped from PCB statement for IMS on z/OS.

Table 27 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

Table 27 Section Name: [imspcb]

Field	Type	Value	Description	Source in PCB
LIST	Bool	Yes No	Specifies whether the named PCB is included in the PCB list passed to the application program at entry, must together with the LABER= parameter	LIST
Following fields are only applicable to TYPE=TP, i.e. alternate PCB, will be ignored if configured but TYPE!=TP				
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
LABEL	X(8)	YES	This parameter is required for DB PCBs while using ODBA, and also required for alternate PCBs if AIBTDLI interface is used to call DL/I.	PCBNAME
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 L S DEDB: One or Combinat ion 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 do not check the validity of PROCOPT, which is migrated from IMS on z/OS and only used by DB plug-in	PROCOPT

Table 27 Section Name: [imspcb]

Field	Type	Value	Description	Source in PCB
Following fields are only applicable to TYPE=DB, i.e. GSAM PCB, will be ignored if configured but TP !=DB				
SB	Enum	COND NO	Specifies whether this PCB should be buffered.	SB
POS	Enum	SINGLE M ULTIPLE	Specifies single or multiple positioning for the logical data structure.	POS
KEYLEN	int	Positive integer	Specifies bytes of the longest concatenated key for a hierarchic path of sensitive segments.	KEYLEN
MSDB	bool	YES NO	Specifies the MSDB commit view.	VIEW
SENSESEG	X(8)	Literal String	Specifies the SENSESEG name belongs to this pcb, must be unique in current PSB file.	
Section Name: [\$SEGSEG]			The section name is defined in imspcb section through SENSESEG item.	
NAME	X(8)	Literal String	Specifies segment name.	SENSESEG - NAME
PARENT	X(8)	Literal String 0	The name of the segment parent.	SENSESEG - PARENT
PROCOPT	X(4)	One or Combination of A G I R D P O E L GS LS H	Indicates the processing options valid for use of this sensitive segment, the SENSESEG PROCOPT overrides the PCB PROCOPT	SENSESEG - PROCOPT

Table 27 Section Name: [imspcb]

Field	Type	Value	Description	Source in PCB
SSPTR	Struct array	One or Combination of interger ,R U linked with semicolon	Specifies the subset pointer number and the sensitivity for the pointer.	SENSEG - SSPTR
INDICES	X(8)	Literal String	Specifies which secondary indexes contain search fields that are used to qualify SSAs for an indexed segment type.	SENSEG - INDICES
SENFLD	X(8)	Literal String	Specifies the SENFLD name belongs to this senseg, must be unique in current PSB file.	
Section Name: [\$SENFLD]	X(8)		The section name is defined in senseg section through SENFLD item.	
NAME	X(8)	Literal String	Specifies field name.	SENFLD - NAME
START	Int	Positive integer [1-32767]	Specifies the starting position of this field relative to the beginning of the segment.	SENFLD - START
REPLACE	bool	YES NO	Specifies whether or not this field can be altered on a replace call.	SENFLD - REPLACE

Segments Definition - segments.desc

segments.desc defines the segments within a database. One database (except the GSAM database), has one segments.desc, which is under \$ART_IMS_CONFIG/db/\$dbname.

The fields in segments.desc are mapped from SEGM statement of DBD.

Table 28 segments.desc

Field	Type	Value	Description	Source in SEGM
NAME	X(1..8)	Mandatory	Segment Name	NAME
BYTES	Integer	Mandatory	Maximum Length of the segment, 4~32760	BYTES
SEGPGM_A2E	X(1..128))	Optional	COBOL Program name used to convert segment data from open system to Mainframe	N/A
SEGPGM_E2A	X(1..128))	Optional	COBOL Program name used to convert segment data from Mainframe to open system	N/A
SSAPGM_A2E	X(1..128))	Optional	COBOL Program name used to convert qualified SSA for the segment from open system to Mainframe	N/A
KFAPGM_E2A	X(1..128))	Optional	COBOL Program name used to convert Key Feedback area for the segment from Mainframe to open system	N/A

Notes: For variable length segment, the BYTES definition is mapped from SEGM statement (e.g., BYTES= (max bytes,min bytes)), max bytes must be greater or equal to min bytes. A variable length segment must starts with a 2-byte field, which defines the length of the segment including the 2-byte length field.

For SEGPGM_A2E, SEGPGM_E2A and SSAPGM_A2E, KFAPGM_E2A:

1. If any of the upper four parameters is not defined in segments.desc, \$segname.desc must exist and \$segname.desc will be used to do data converting as described later.
2. SEGPGM_A2E and SEGPGM_E2A.

SEGPGM_A2E and SEGPGM_E2A must be both defined or both not defined together. When SEGPGM_A2E and SEGPGM_E2A are defined, the ODBA Plugin uses

SEGPGM_A2E/ SEGPGM_E2A to do segment data converting, not use the \$segname.desc to perform segment data converting even if there is \$segname.desc.

When SEGPGM_A2E and SEGPGM_E2A are not defined, the ODBA plug-in uses the FIELDS definition in \$segname.desc to perform segment data converting.

3. SSAPGM_A2E

When SSAPGM_A2E is defined, the ODBA plug-in uses the defined COBOL program to do qualified SSA converting for this segment.

When SSAPGM_A2E is not defined, the ODBA Program uses the KEY Field type definition in \$segname.desc to convert the KEY value in the SSA for this segment.

4. KFAPGM_E2A.

When KFAPGM_E2A is defined, the ODBA plug-in uses the defined COBOL program to do data converting for Key Feedback area for this segment.

If KFAPGM_E2A is not defined, ODBA program uses the FBAFIELD definition in \$segname.desc to do data converting for Key feedback area for this segment.

For how to generate and compile the upper COBOL programs, please refer to buffer converting in [Oracle Tuxedo Application Rehosting Workbench User Guide](#) and [Oracle Tuxedo Application Rehosting Workbench Reference Guide](#).

Put the compiled converting program under COBPATH (for Microfocus) or COB_LIBRARY_PATH (for CIT) before booting ART IMS servers.

Segment Definition - \$segname.desc

\$segname.desc defines the fields within a segment. \$segname.desc only exist for databases with access type of neither GSAM nor MSDB defined in imsdbs.desc, \$segname.desc is located under \$ART_IMS_CONFIG/db/\$dbname.

Table 29 \$segname.desc

[imsdb]Field	Type	Value	Description	Source in SEGM
NAME	X(1..8)	Mandatory	Field Name	NAME
START	Integer	Mandatory	Offset of the field, inside the segment, 0~length of segment	START

- 4

Table 29 \$segname.desc

[imsdb]Field	Type	Value	Description	Source in SEGM
BYTES	Integer	Mandatory	Length of the field, 4~32760	BYTES

Table 29 \$segname.desc

[imsdb]Field	Type	Value	Description	Source in SEGM
TYPE	Enum	Mandatory	<p>C: Alpha-Numeric string, requires conversion.</p> <p>P: Packed Decimal, conversion not required.</p> <p>X: Hexadecimal, requires conversion.</p> <p>M: Mixed type, used when a FIELD contains different types of sub pieces. Requires conversion.</p>	TYPE

Table 29 \$segname.desc

[imsdb]Field	Type	Value	Description	Source in SEGM
FORMAT	X(1...512)	Optional	<p>Only needed (mandatory) for TYPE=M.</p> <p>The configuration rule for this format is as follows:</p> <pre>format=PIECE1-LEN,PIECE1-TYPE;PIECE2-LEN,PIECE2-TYPE;PIECE3-LEN,PIECE3-TYPE.</pre> <p>For example:</p> <pre>format=3,P;4,C;3,p.</pre> <p>Please note the following:</p> <ol style="list-style-type: none"> 1. The PIECE-TYPE must be C/P/X. 2. Tailing - is not allowed. 3. The format string must cover all bytes in the field. 	N/A

The field's type definition is not only from FIELD statement of DBD, user also needs to define the field's type according to its usage in COBOL program.

If you do not define the type for a field, the default field type is C

[Table 30](#) shows the Field Definition mapping table according to field usage in the COBOL program.

Table 30 Field Definition Mapping Table

COBOL Picture	Description	IMS Type
PIC X	Alph-numeric character display	C
PIC 9	Numeric display	C
PIC S9	Signed numeric display	C
PIC S9 COMP-3	Packed decimal	P
PIC S9 COMP; PIC S9 COMP-4	Binary	X

The conversation/translating rule is as follows:

- Type C, requires ASCII/EBCDIC translation.
- Type P, does not require translation. (Both ASCII/EBCDIC translation and endian translation are not done. This for COMP-3 data area.)
- Type H, requires endian translation.

At the completion of a retrieval or `ISRT` call, the Key Feedback Area is returned from ODBA Proxy. We reserve one special `FIELD` whose name is `FBAFIELD` to convert Key Feedback Area in DB PCB.

A segment concatenated key is made up of the keys of each of its parents and its own key. Key formats are positioned left to right, starting with the key format of the root segment and following the hierarchic path.

This special `FIELD(FBAFIELD)` defined in `$segment` must defined the format of the concatenated key of the segment. The Key FeedBack Area will be converted according to the definition of this field for the segname.

[Listing 8](#) shows an example definition in `$segname.desc`.

Listing 8 Example Definition

```
[field]
NAME=FBAFIELD
START=0
BYTES=11
TYPE=M
FORMAT=5,P;6,C
```

START should be 0.

BYTES defines the total length of the KEY FeedBack Area, that is the concatenated key total length.

TYPE defines the concatenated key's type. If the concatenated key has different type, the TYPE should be set to M.

FORMAT should define the concatenated key format if the TYPE=M.

If the special FIELD(FBAFIELD) is not defined in \$ segname.desc, the KEY FeedBack Area is converted as TYPE C by default for the segname.

Note: For search key field in SSA, its field name must be the same as the Search KEY FIELD name defined in DBD and the name used in the SSA.

See Also

- [Oracle ARTIMS Users Guide](#)

