

# Developer's Guide

OpenFrame OSI 7.2

**TMAXSOFT**

# Copyright

Copyright 2025. TmaxSoft Co., Ltd. All Rights Reserved.

## Company Information

TmaxSoft Co., Ltd.

TmaxSoft Tower, Jeongjail-ro 45, Bundang-gu, Seongnam-si, Gyeonggi-do 13613, South Korea

Website: <https://www.tmaxsoft.com/en/>

## Restricted Rights Legend

All TmaxSoft Software(Tmax OpenFrame®) and documents are protected by copyright laws and international convention. TmaxSoft software and documents are made available under the terms of the TmaxSoft License Agreement and this document may only be distributed or copied in accordance with the terms of this agreement. No part of this document may be transmitted, copied, deployed, or reproduced in any form or by any means, electronic, mechanical, or optical, without the prior written consent of TmaxSoft Co., Ltd. Nothing in this software document and agreement constitutes a transfer of intellectual property rights regardless of whether or not such rights are registered) or any rights to TmaxSoft trademarks, logos, or any other brand features.

This document is for information purposes only. The company assumes no direct or indirect responsibilities for the contents of this document, and does not guarantee that the information contained in this document satisfies certain legal or commercial conditions. The information contained in this document is subject to change without prior notice due to product upgrades or updates. The company assumes no liability for any errors in this document.

## Trademarks

Tmax® and Tmax OpenFrame® are registered trademarks of TmaxSoft Co., Ltd. Other products, titles or services may be registered trademarks of their respective companies.

## Open Source Software Notice

This product includes open source software developed and/or licensed by "OpenSSL", "RSA Data Security, Inc.", "Apache Foundation", and "Jean-loup Gailly and Mark Adler". Information about the aforementioned and the related open source software can be found in the following directory:  
\${INSTALL\_PATH}/license/oss\_licenses

## Document History

Product Version	Guide Version	Date	Remarks
OpenFrame OSI 7.2	3.2.1	2025-08-14	
OpenFrame OSI 7.2	3.1.2	2023-12-29	-
OpenFrame OSI 7.2	3.1.1	2023-10-12	-

# Contents

1. Introduction	1
1.1. Overview	1
1.2. Database	1
1.3. Data Communications	1
1.4. Applications	1
2. Application Interfaces	4
2.1. Overview	4
2.2. DL/I Interface	4
2.3. AIBTDLI Interface	4
2.4. Database Call Functions	5
2.5. Data Communication Call Functions	7
3. Application Creation	9
3.1. Process of Creating Applications	9
3.1.1. Application Design	9
3.1.2. Programming	10
3.1.3. Batch Applications	10
3.2. MPP Applications	10
3.2.1. MPP User Server Preparation	11
3.2.2. MPP Programming	12
3.2.3. Message Segment Format	13
3.3. BMP Applications	15
Appendix A: DL/I Status code	17
Appendix B: IO-PCB Mask	21
Appendix C: AIB Mask	22
Appendix D: DL/I Call	23
D.1. System Service DL/I Calls	23
D.1.1. CHKP (basic)	23
D.1.2. CHKP (symbolic)	23
D.1.3. INIT	24
D.1.4. INQY	24
D.1.5. LOG	26
D.1.6. ROLB	26
D.1.7. SYNC	26
D.1.8. XRST	27
D.2. Transaction Management DL/I Calls	27
D.2.1. CHNG	27
D.2.2. CMD	28
D.2.3. GCMD	28

D.2.4. GN.....	29
D.2.5. GU.....	29
D.2.6. ISRT.....	29
D.2.7. PURG .....	30

# 1. Introduction

This chapter describes the concepts and structure of OpenFrame OSI application programming.

## 1.1. Overview

Developers can configure transaction services by developing new applications that the OSI system runs or by migrating applications that the existing IMS/DC runs. Many resources that are provided by the OSI system provide the same services as those of IMS/DC.

Based on **DL/I** function, OSI programming constitutes the work program by combining the features of each function. Refer to [Application Interfaces](#) for detailed information about types and features of DL/I function.

## 1.2. Database

The type of database that the IMS/DC system uses is a hierarchical database management system. It saves and manages clients' work data structurally. This database is called IMS/DB. Call the standard DL/I function when applications that are run in the IMS/DC or pure batch applications access this database.

By using **DL/I**, a standard application interface that each program language provides, applications can use the standard DL/I functions.

OSI rebuilds the database structure on UNIX and integrates with OpenFrame HiDB to provide the same features as IMS/DC.

## 1.3. Data Communications

IMS/DC provides a system server (Control Region) and a user server (Dependent Region) to view the results of the data that applications processed..

OSI provides features of IMS/DC by integrating with Tmax and efficiently handles transactions. To make clients' programs that are run in the IMS/DC equally run in the OSI, OSI provides the same interface as the DL/I interface that IMS/DC provides.

By using **DL/I**, user programs, that MPP and BMP in a user server (Dependent Region) run, can be run in the OSI following a simple migration.

## 1.4. Applications

OSI provides the same features and infrastructure that IMS/DC provides so, applications can run in the same way as they do in IMS/DC.

Unlike Non IMS/DC applications, IMS/DC applications must have a PSB (Program Specification Block). PSB functions as an interface to run the services that IMS provides for applications. Those services include how to send a message to a terminal, how to access a database, IMS commands, and IMS service Calls. OSI applications can determine how to run IMS services through PSB.

PSB and applications are concurrently loaded by the OSI system. The execution module of the OSI implements what applications request by using the Call function that is run within the applications. Each MPP application implements OSI features (DL/I Call function, commands, etc.) through a MPP-typed user server and for this, required system definitions must be registered in advance. The system can be defined by using the system macro that IMS/DC used.

Batch applications are run in a different way to the above case. To run batch applications in the OSI system, a BMP (Batch Message Processing Region) server is needed. This server can handle the functions of applications that are requested using the batch JOB (JCL start).

When an application is created, define PCB in the ENTRY statement. Through this defined PCB, features of the OSI system can be used. PCB is a control block that the system provides to communicate with VIEW about OpenFrame HiDB, message sources, or message destinations. Up to 400 PCBs can be used in OSI.

The following describes PCB for applications.

- Database Program Control Block (DB-PCB)

Describes how applications read information that is stored in the database. Even the same database can be read differently depending on how it is described in the DB-PCB.

- IO Program Control Block (IO-PCB)

OSI typically operates using the terminal and applications that are run online require IO-PCB.

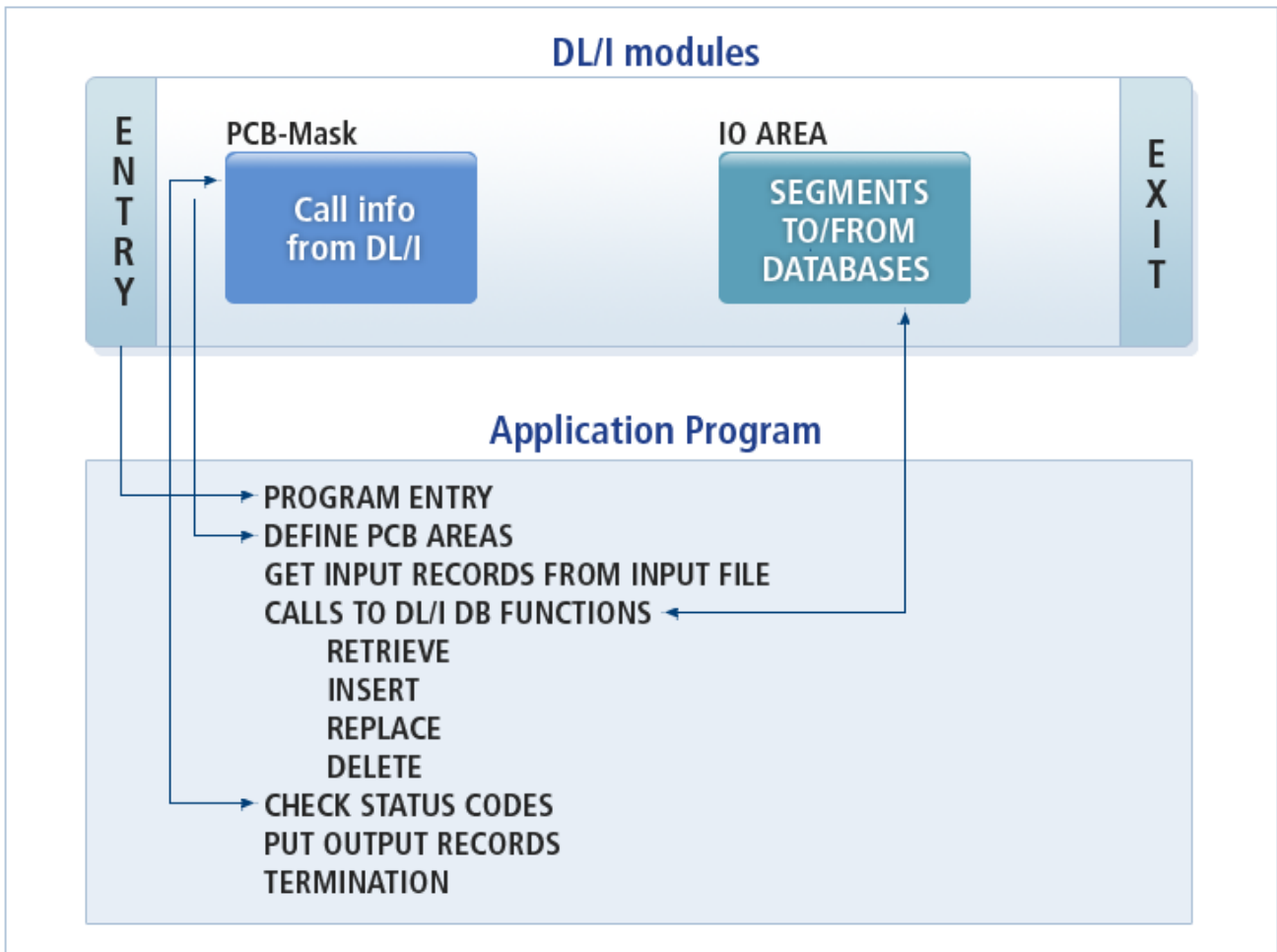
- Alternative Program Control Block (ALT-PCB)

Use ALT-PCB when a message is forwarded to other programs or terminals within applications. ALT-PCB can be used when changing the destination.



For more information about PSB and DBD, refer to *OpenFrame HiDB Guide*.

The following shows the general workflow of applications described earlier.



Application Workflow

## 2. Application Interfaces

This chapter describes types and usage of the interface that COBOL written in the OSI environment uses.

### 2.1. Overview

In the OSI environment, applications must follow the certain interface (**DL/I call function**) to access a database or MQ. Each function is used in the OSI area (online program) as well as the database. Their method of use is the same. Specific options such as segment search argument (SSA) are not used online. A database can be thought of as a Message Queue (MQ) online and assigned functions.

If a user do not follow the interface, return the status code which a user can recognize as a return value. Refer to [DL/I Status code](#) for detailed information about the status code.



For more information about DL/I call functions, refer to *OpenFrame HiDB Guide*.

### 2.2. DL/I Interface

To use DL/I Call functions, an application must use the DL/I interface. Depending on the application language, various DL/I interfaces can be used, such as CBLTDLI, PLITDLI, and ASMTDLI.

The following shows how to use the DL/I interface. Specify the DL/I Call function in argument 1 and the PCB in argument 2.

```
CALL 'CBLTDLI' USING argument 1, argument 2, ..., argument n
```

### 2.3. AIBTDLI Interface

The AIB (Application Interface Block) can be used when the PCB address is unknown or when the DL/I Call function does not require a PCB. If a DL/I Call function requires a PCB, the resource name in the AIB must match the PCB name registered during psbgen execution.

The following shows how an application uses the DL/I Call function via the AIBTDLI interface. Specify the DL/I Call function in argument 1 and the AIB in argument 2.

```
CALL 'AIBTDLI' argument 1, argument 2, ..., argument n
```

## 2.4. Database Call Functions

The following shows how to use functions that applications use to access a database.

- Usage

```
Function Code(argument 1) PCB(argument 2) Data Area(argument 3) SSA(argument 4)
```

- Parameters

- argument 1

Refers to the DL/I function of CBLTDLI. A function code determines the type of processing a database segment. This field's length is 4 bytes and it is declared within the application.

These are functions that can be specified in argument 1.

Function	Description
GU (GET UNIQUE) / GHU (GET HOLD UNIQUE)	A function code that searches a specific segment of a database. Segments can be gained by specifying SSA(Specify from the 4th to n-th of argument.).  In online programs, it requires messages that are loaded in the MQ to be read.
GN (GET NEXT) / GHN (GET HOLD NEXT)	A function code that searches the segment following the currently positioned segment (a retrieved segment right before Call issue). It is used when a database segment is searched in hierarchical order.  In online programs, MQ functions as a database. Features of a function do not target the database but the MQ.
GNP (GET NEXT WITHIN PARENT) / GHNP (GET HOLD NEXT WITHIN PARENT)	Certain segment types of the data structure are parents and can contain a segment in the lower level of the segment. It is called a family segment and GNP and GHNP function codes are searched in hierarchical order of this family segment.  It is not used when an online program (IO-PCB) is used.
ISRT (INSERT)	Insert data that is typed in the argument 3 into a specified database or MQ.
DLET (DELETE)	A function code to delete a segment. When deleting a segment, use DLET Call after positioning a segment using a hold-based search Call. The segment is removed from the database according to the delete rule.

Function	Description
REPL (REPLACE)	A function code to implement substitution of a segment. Like DLET Call, use REPL Call after positioning a segment using a hold-based search Call. Do not change a segment key using the REPL Call.  It is not used when an online program (IO-PCB) is used.

- argument 2

To make applications use data access and interface, define PCB Mask as either DB-PCB, IO-PCB, or ALT-PCB. When applications were written in COBOL, define PCB in linkage. When PCB that is defined in applications functions as Call, it is linked with the control section of OSI.

PCB Mask	Description
DB-PCB	Regarding database Call features, the defined PCB is <b>DB-PCB</b> .
IO-PCB	It is used when online features are used.
ALT-PCB	When data needs to be sent to another transaction or terminal, it is used.

- argument 3

Specifies the data area within the application. Data processing of applications and data access is used by specifying it in the data area. A user must declare a segment length and a necessary length by Command Code of SSA in the XX area within the application.

- argument 4

Defines the conditions for segments that are the subject of the process of SSA and sensitive segments. Specify SSA's leading name in accordance with the hierarchy level that is defined as the data structure. The maximum of n is 18. When OP-PCB Mask is used, the screen on the terminal can be changed by configuring the MOD name.

SSA is a specific argument of a database Call feature and divided into unqualified SSA and qualified SSA depending on the presence of the condition that is forwarded to a segment.

SSA	Description
unqualified SSA	Processes SSA using the segment name and does not attach conditions to the field information in the segment.
qualified SSA	Attaches conditions to the field information in the segment.



For more information about database call functions, refer to IBM's *IMS Application Programming: Database Manager*.

## 2.5. Data Communication Call Functions

The following describes features of the data communications Call function.

- Sending and receiving message segments
- Changes to the destination of transmission messages
- Transmission of the Scratch Pad Area (SPA)

Features of the data communications Call function is the same as those of the database Call function. But, messages that are loaded in MQ and messages in the SPA area do not have a hierarchical structure (a characteristic structure of OpenFrame HiDB). Therefore, the SSA of the database Call, which is a hierarchical search condition, is not needed.



For more information about Scratch Pad Area (SPA), refer to [IBM IMS Conversational Transactions](#).

- Usage

```
Function Code(argument 1) PCB(argument 2) Data Area(argument 3) MOD Name (argument 4)
```

- Parameters

- argument 1

Specifies function codes which define message segments and SPA type processing. As a 4 byte area, like the database Call feature, it is declared in the application.

The following shows functions that can be specified in argument 1.

Function	Description
GU (GET UNIQUE)	Receives the first segment (header segment) of a message. Applications must call GU first when receiving a message.  SPA is received during conversation mode.
GN (GET NEXT)	Receives message segments following header segment.
ISRT (INSERT)	Sends message segments or SPA.
PURG (PURGE)	Outputs message segments that are sent by ISRT, as a message to the terminal.
CHNG (CHANGE)	Uses ALT-PCB and sends messages to another terminal or transaction.

- argument 2

PCB for a data communications Call function is defined in the same way as the database PCB

and called a **logic terminal PCB**.

The following shows the type of logic PCB terminals which can be specified in argument 2.

PCB	Description
IO-PCB	To input and output messages from the logic terminal that requested the transaction.
Alternative PCB	To send messages using another terminal rather than the default terminal.
Alternative Program PCB	To send messages to another program.



1. Alternative Program PCB is called ALT-PCB. This logic terminal PCB within the application must be defined IO-PCB and ALT-PCB in order.
2. IO-PCB is automatically managed in the OSI system and ALT-PCB is defined as program definition utility.

◦ argument 3

A data area that is used for message segments, commands, and the SPA's sending/receiving.

◦ argument 4

For the terminal that performs MFS, define the address of the area where the MOD name (8Byte), which is defined by the message format definition utility, is configured. The MOD name is specified only if it is inserted.



For more information about data communication call functions, refer to IBM's *IMS Application Programming: Transaction Manager*.

# 3. Application Creation

This chapter describes how to create applications.

## 3.1. Process of Creating Applications

The process of creating applications can be separated into application design, programming, debugging, and execution.

Each step includes resources that must be defined in the OSI and programming that users randomly create.

### 3.1.1. Application Design

The application design process proceeds as follows:

1. A developer must decide what type of user server applications will run in.

There are two types of user server that process applications; MPP user servers and BMP user servers. It must be determined which server type will be implemented.

2. Once the type of the server is determined, the relationship between the application and the user server must be defined.

The relationship can be defined in the system resource definition.

The following is an example of system resource definition.

```
APPLCTN PSB=OIVPI001,PGMTYPE=(, ),SCHDTYP=PARALLEL
TRANSACTION CODE=OIVPMPP1,MSGTYPE=(SINGLSEG,RESPONSE,1),PRTY=(1,5),      X
        MODE=SNGL
APPLCTN PSB=OIVPI002,PGMTYPE=(TP, ,1),SCHDTYP=PARALLEL
TRANSACTION CODE=OIVPMPP2,MSGTYPE=(SINGLSEG,RESPONSE,1),PRTY=(1,5),      X
        MODE=SNGL,MAXRGN=1
APPLCTN PSB=OIVPIL05,PGMTYPE=BATCH,SCHDTYP=PARALLEL
TRANSACTION CODE=OIVPBMP5,MSGTYPE=(SINGLSEG,RESPONSE,1),PRTY=(1,5),      X
        MODE=SNGL
```

3. Applications need to be designed with consideration to the program structure, taking maintenance and scalability into account, as well as module design. When designing applications, note the following.
  - Effectively use the CBLTDLI function call.
  - Minimize the number of times data is being input or output.
  - Consider the data structure of the database and the processing options (search, add, and delete).

- Consider the configuration of the processing mode (interactive mode processing and general transaction mode).
- Consider the message path.

### 3.1.2. Programming

Write OSI applications in COBOL and note the following.

- Configuring the start conditions of the program.
- Configuring DB-PCB, IO-PCB, and ALT-PCB.
- Configuring the function call interface.
- Configuring the checking status codes and handling errors.
- Integrity during data processing. (e.g. Data length)
- Configuring the termination conditions of the program. (e.g. Return code)
- Regardless of input and output data size or SSA size, the data transfer area when programming must be large enough.

### 3.1.3. Batch Applications

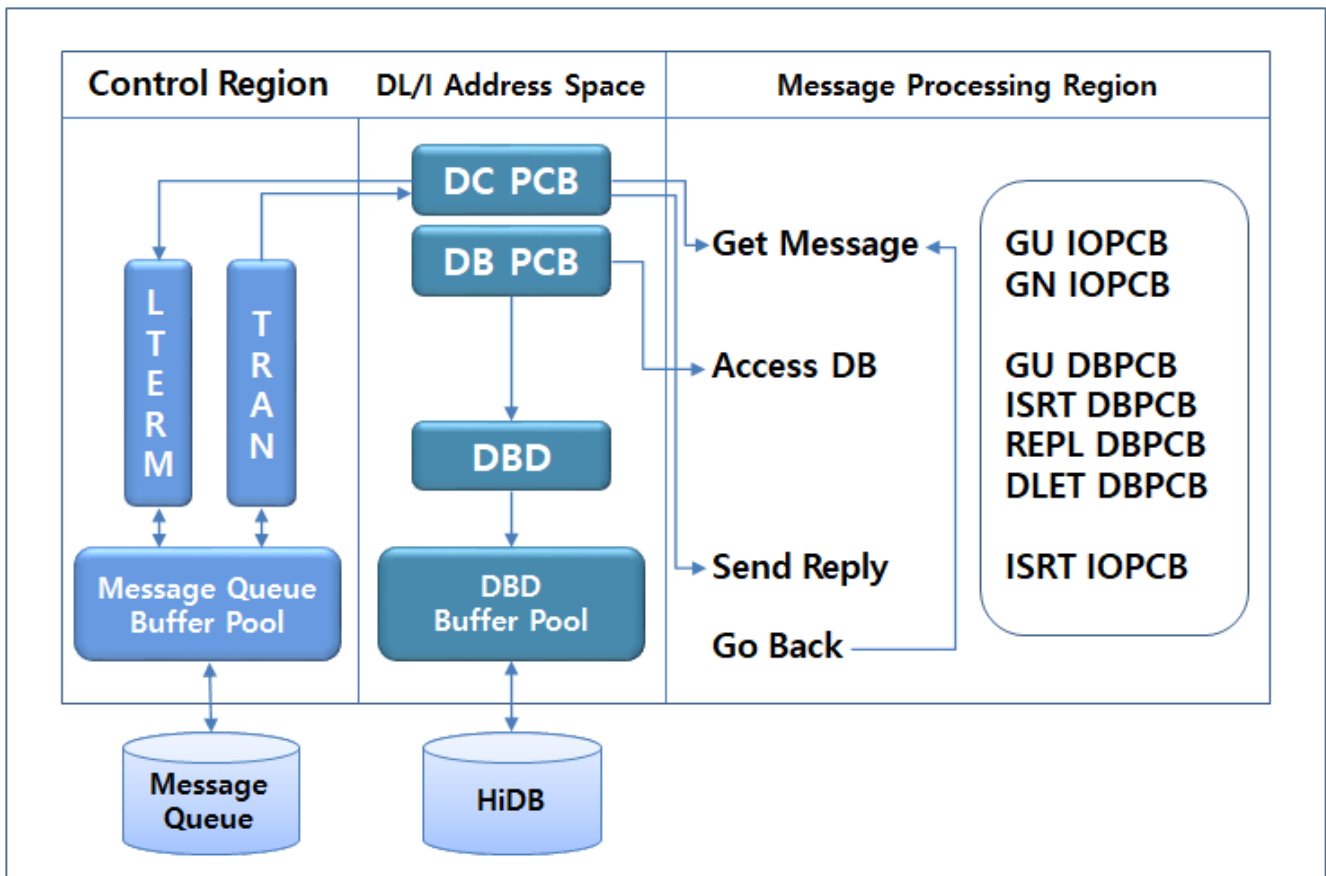
Batch applications are implemented by **tjesmgr**, which is OpenFrame's utility. The way of creating batch applications is the same as that of MPP applications. A difference is that it runs through JCL. **tjesmgr** is an utility to run JCL effectively.

Batch applications are not for online tasks. Therefore, there is no need to define the data area to communicate with terminals. Logic that can connect programs to databases, logic that can load messages on Message Queue (MQ), and logic that can stack data on MQ are only needed.

## 3.2. MPP Applications

Message Processing Program (MPP) and Message Processing Region (MPR) process messages in the IMS/DC Online environment. OSI separates them into the system server and the user server.

The following is a picture of the OSI system and application configuration of online environments.



OSI System and Application Configuration

In general, the transaction input from the terminal (LTERM) is analyzed by the system server (Control Region), and the MPP application that processes this transaction is loaded and executed in the MPP user server, which is one of the user servers of OSI. The data area that this transaction delivers to the OSI system consists of a transaction code (up to 8 characters) and a data area. The system server acquires class information from the RunTime System Definition (RTSD) table using the received transaction code, and schedules it with the MPP user server that processes the class.

The MPP application can get the message input from the terminal with Get Unique (GU) Call. It enables to process multiple MPP applications by starting multiple user servers and to access the database at the same time. In addition to LTERM, MPP applications can be started between transactions using ALT-PCB.

### 3.2.1. MPP User Server Preparation

To run MPP user servers, Tmax environment file settings and an MPP start JCL file are required. OSI can start MPP servers that process different class information, by using Tmax's target option with a OSIMPPSVR binary file.

- **Tmax Environment File Settings**

Register user servers on the Tmax server to normally run OSI as it runs in the Tmax environment. The method of registering a server is the same as that of registering a server in Tmax. When registering Tmax servers, related services must be registered as well.

```
#####
#   OpenFrame OSI Dependent Region Servers                               #
#####
OSIMPPSVR      SVGNAME = svg_node1, MIN = 1, MAX = 10, RESTART = NO
#####
#   OSI, example in which IMSID is IMSA                                 #
#####
IMSASCHD       SVGNAME = svg_node1, MAX = 1, SVRTYPE = UCS, RESTART = NO,
                TARGET = osisschd
IMSACMMD       SVGNAME = svg_node1, MAX = 1, SVRTYPE = UCS, RESTART = NO,
                TARGET = osicmdsv
IMSAMPP_TCL1   SVGNAME = svg_node1, MIN = 1, MAX = 10, RESTART = NO,
                TARGET = OSIMPPSVR
IMSAMPP_TCL2   SVGNAME = svg_node1, MIN = 1, MAX = 10, RESTART = NO,
                TARGET = OSIMPPSVR
IMSAMPP_TCL3   SVGNAME = svg_node1, MIN = 1, MAX = 10, RESTART = NO,
                TARGET = OSIMPPSVR
IMSAMPP_TCL4   SVGNAME = svg_node1, MIN = 1, MAX = 10, RESTART = NO,
                TARGET = OSIMPPSVR

*SERVICE
#####
#   OSI USER APPLICATION SERVER DEFAULT                               #
#####
OSIMPPSVRSVC   SVRNAME = OSIMPPSVR, SVCTIME=60
```

### • MPP Server Start JCL

The following is a JCL file that starts the MPP user server corresponding to a user server (Dependent Region) created by the administrator when a system server's (Control Region) name is IMSA. If no class is specified, it can be set to '000', and the first class must be specified. (In the following example, classes to be processed are 1, 2, 3, and 4.)

```
//IMSAMSG JOB
//STEP1 EXEC PGM=DFSRRC00,
//          PARM='MSG,001002003004,W00099000,,,R,,,,IMSA,,,,D2PA'
//STEPLIB DD DISP=SHR,DSN=OSI.IMSA.STEPLIB
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSDBOU DD SYSOUT=*
//
```

## 3.2.2. MPP Programming

The basic MPP programming consists of GU and ISRT calls. When the program issues a GU call to process a message, the DL/I function is called through IO-PCB. When a message consists of more than one segment, read the message using GN Call from the second segment. (In the case of a transaction using the SPA, read the SPA using GU and then process using GN).

The results of the GU and GN Call are configured in the status code field of IO-PCB. The MPP program must check the status code after processing the call. When message segments are sent to the terminal or other transactions, configure the message segments in the data area in a specified

format and call the DL/I function using ISRT Call.

When a message is sent to another MPP program or terminal, specify the destination using ALT-PCB.

## Note

Note the following during MPP programming.

- MPP is dynamically loaded onto the empty user server whenever a transaction is input. After processing MPP is complete, the user server is allocated to another MPP program or changes to IDLE state. Therefore, information cannot be saved on the MPP user server. Use the SPA to save the data.
- To run MPP, OSI has to do a lot of pre-processing. Once MPP is loaded, create a program that can process multiple messages. However, if too much time is spent processing messages, equalized transactions cannot be scheduled in accordance with the priority that is specified when the system is defined. In this case, use the PROCLIM specification in the TRANSACT macro.
- If the database is updated using MPP, care will be needed because processing of the database is implemented. If applications are too big, the database will be exclusively accessed. This situation should be avoided.
- To run equalized scheduling services, create MPP using multiple transactions rather than one transaction.
- During the MPP process, check the status code. If there are any errors, further processing will be impossible. If it is necessary to exit a database while processing is being carried out, terminate it abnormally (ABEND).
- If **USE\_DBD\_DBMS\_LOCK** option in **HIDB\_DEFAULT** section and **hidb** subject of OpenFrame configuration is set to YES, lock is created with the DBD name used by the transaction when the transaction starts, and lock is released when the transaction ends. For more information, refer to *OpenFrame Configuration Guide*.

### 3.2.3. Message Segment Format

The following describes receiving message segments, sending message segments, and message segments transferred between programs.

#### 3.2.3.1. Received Message Segments

User programs (applications) receive messages from terminals or from other transactions. Using the GU and GN Call of the DL/I function, data is received in the data area that COBOL defines.

The message segment format that applications receive is as follows:

```
LL ZZ DATA
```

Field	Description
LL	<p>The length of a message segment that includes the length of the LL and ZZ fields. It is a binary number and is 2 Bytes.</p> <p>(LL = DATA length+4)</p> <p>For applications written by PL/I, it is 4 Bytes and its length is the actual length minus 2.</p>
ZZ	<p>It is an area for data access and is 2 Bytes. A developer does not need to configure it. If it is randomly modified, errors will occur.</p>
DATA	<p>A message segment that is input from the terminal. A message segment consists of a transaction code and a data area. Front 8Byte of the data that is sent to the terminal becomes a transaction name and the rest becomes the actual data.</p> <p>For messages that are input as multiple message segments, a transaction code must be included within the first 8 Bytes of the first message segment.</p>

### 3.2.3.2. Sending Message Segment

When messages are sent to the terminal, other transactions, or other terminals in applications, send messages to the Destination that is configured in ALT-PCB or IO-PCB using the ISRT Call of the DL/I function.

The format for sending message segments is as follows:

```
LL ZZ DATA
```

Field	Description
LL	<p>The length of a message segment that includes the length of the LL and ZZ fields. It is a binary number and is 2 Bytes.</p> <p>(LL = DATA length + 4)</p> <p>For applications written in PL/I, it is 4 Bytes long. This field must be configured in applications.</p>
Z1	<p>It is an area for data access and is 1 Byte. In applications, 0 must be input as a binary number.</p>
Z2	<p>Its length is 1 Byte. For paging logic, input X'40' if the start of a new page needs to be notified to MFS. Otherwise, input 0 as a binary number. Refer to <i>OpenFrame OSI MFS Reference Guide</i> for detailed information.</p>
DATA	<p>A data area that is sent to the specific Destination(the current terminal, other transactions, and other terminals). The length of a message segment depends on the Destination. Multiple message segments are fine as well.</p>

### 3.2.3.3. Message Segments between Programs

Messages can be sent to another program from a specific program. Messages are transmitted between programs by using the previously mentioned method of sending message segments.

The format of delivering message segments between programs is as follows:

```
LL ZZ DATA
```

Field	Description
LL, ZZ	The format is the same as that of sending message segments.
DATA	Sends messages to the target program.

#### Note

Note the following when using message segments that are delivered between programs. The ALT-PCB that applications use must be defined at PSB in advance and PCBs (LINKAGE SECTION and ENTRY syntax) that applications use must be defined in the same order of PCBs which are defined in PSB.

- When application AP1 sends a message to application AP2, use ALT-PCB. After configuring the Destination of ALT-PCB as the name of AP2, use the DL/I function in the order of CHNG, ISRT, and PURG.
- When receiving a message from AP2, specify IO-PCB to get the message. ALT-PCB of AP1 must be configured at IO-PCB so that data can be normally received (data structure integrity).
- If a message needs to be sent to AP1 after processing a received message in AP2, specify the ALT-PCB of AP1. In this case, AP1 receives a message from AP2 and processes it. When a message is output to the terminal, send it by specifying IO-PCB. AP2 can receive a message in the case of MPP or BMP.

## 3.3. BMP Applications

Unlike MPP, a user uses data management in addition to the ability to access a database and access user data set in OSI. How to process a database or a message is the same as MPP. A difference from MPP programming is that data which is loaded onto MQ is not processed in real time and is processed by running BMP applications through JCL when required by a user.

For example, when the output processed through the MPP application is sent to the BMP application using ALT-PCB (Alternative Program PCB), OSI loads the result into an MQ table, waits, and then when the user executes the corresponding BMP application with JCL, the MQ table sends all messages loaded in the BMP application to the destination.

The following is an example of JCL that starts the BMP program.

<OSIBMP.jcl>

```
//OSIBMPT JOB CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//TSTEP1 EXEC PGM=DFSRRRC00,
// PARM=(BMP,OIVPIL04,,OIVPBMP4,,,,,,,,,IMSA,)
//SYSOUT DD SYSOUT=*
```

The following is an example of implementing a tjesmgr utility that executes above JCL.

```
$ tjesmgr run OSIBMP.jcl
> Command : [run OSIBMP.jcl]
Node name : NODE1
/home/oframe/OpenFrame/volume_default/SYS1.JCLLIB/OSIBMP.jcl is submitted as OSIBMPT(JOB00001).
$ tjesmgr ps
> Command : [ps]
JOBNAME JOBID CLASS STATUS RC NODE JCL
-----
OSIBMPT JOB00001 A Done R00000 NODE1 OSIBMP.jcl
```

# Appendix A: DL/I Status code

This appendix describes DL/I status codes and how to handle them.

The result of DL/I Call processing can be identified using the code value of the PCB status code field. Application programmers can check causes for the results of processing and how to respond to them, by checking the status code field.

- AB

<b>Cause</b>	In function calls that require an I/O Area and an I/O Area has not been specified as a parameter.
<b>Action</b>	Specify the correct data area that corresponds to the function calls.

- AD

<b>Cause</b>	Specified PCB that does not correspond to the function call as a parameter.  (1) IO-PCB was not used in the system service call.  (2) ALT-PCB was used rather than IO-PCB in the GU call or GN call.  (3) The message GU call was used for the IO-PCB without defining IN=trancode in BMP JCL.
<b>Action</b>	Determine if the PCB is appropriate to the function call.

- AZ

<b>Cause</b>	(1) Used the PURG call to send the SPA.  (2) The first message segment is not SPA when a conversational transaction issues an ISRT ALTPCB call.
<b>Action</b>	Correctly use the PURG and ISRT call.

- A2

<b>Cause</b>	Used the CHNG call incorrectly.  (1) Used the CHNG call if not ALT-PCB.  (2) Used the CHNG call for the PCB not for ALT-PCB.  (3) Used the CHNG call for the PCB whose message segment has not been sent.
--------------	---

<b>Action</b>	(1) Use a modifiable ALT-PCB as the CHNG call.  (2) If the destination needs to be changed after sending a segment, change the destination using the CHNG call after a message purge.
---------------	---

- A3

<b>Cause</b>	For the modifiable ALT-PCB, ISRT or the PURG call was used without specifying the destination. The PURG call is processed as a parameter for a single I/O Area.
<b>Action</b>	Use ISRT or the PURG call for the PCB which defined the destination using the CHNG call.

- A4

<b>Cause</b>	No access to the destination designated by CHNG Call.
<b>Action</b>	Check if the sender designated by CHNG Call is correct and check whether the sender is authorized.

- A5

<b>Cause</b>	Used the wrong parameter list when using a message ISRT call. It is not the first segment of the output message but the 4th parameter (MOD name) was specified.
<b>Action</b>	If it is not the first segment of the message, do not specify the 4th parameter.

- A6

<b>Cause</b>	The size of a message segment that is sent using the message ISRT call or PURG call is bigger than the SEGSIZE of the TRANSACT macro.
<b>Action</b>	Make the size of the message segments to be sent, smaller than the SEGSIZE of the TRANSACT macro.

- QC

<b>Cause</b>	As a message for the applicable program does not exist in the input queue, the GU call was used.
<b>Action</b>	The applicable code is output to notify status information, as application programs stop.

- QD

<b>Cause</b>	The GN call was used but there is no segment in the applicable message.
<b>Action</b>	Process the message properly in the application program as there is no segment.

- QE

<b>Cause</b>	The message GU call was not used but the message GN call.
<b>Action</b>	To use the message GN call, first use the message GU call.

- QH

<b>Cause</b>	The name of the output logical terminal or transaction code was not registered in IMS. The length of the message segment that was sent using the message ISRT call or PURG call is smaller than 5 bytes.
<b>Action</b>	Check the name of the logical terminal and transaction code.

- XA

<b>Cause</b>	After responding to the originating terminal, SPA was sent to another program.
<b>Action</b>	Modify application programs.

- XB

<b>Cause</b>	After sending the SPA to another program, the output message was sent to the originating terminal.
<b>Action</b>	Modify application programs.

- X2

<b>Cause</b>	The destination of ALT-PCB is interactive transaction. The first ISRT call was not sent to the SPA.
<b>Action</b>	After the SPA is inserted, messages are inserted.

- X3

<b>Cause</b>	As the SPA's first 6-byte field has changed, it cannot be defined as the SPA message.
<b>Action</b>	Do not change SPA's first 6 bytes.

- X4

<b>Cause</b>	The SPA was sent using a transaction in which interactive processing is not defined.
<b>Action</b>	Request data messages, except for the SPA, to be inserted.

- X5

<b>Cause</b>	Tried to send the SPA using the ISRT call, more than twice.
<b>Action</b>	Send a SPA to a message.

- X6

<b>Cause</b>	Invalid transaction code for transmitting SPA through an ISRT call.
<b>Action</b>	Specify the destination using interactive transaction code that is registered in the MS system.

# Appendix B: IO-PCB Mask

The following shows the IO-PCB Mask fields provided by IBM mainframe's IMS/DC. It includes the field lengths and support status in the OSI product.

<b>Descriptor</b>	<b>Length (Bytes)</b>	<b>Support</b>
Logical terminal name	8	Supported
Reserved for IMS	2	Supported
Status code	2	Supported
Local date and time	4 (Date) + 4 (Time)	Supported
Input message sequence number	4	Not Supported
Message output descriptor name	8	Supported
Userid	8	Supported
Group name	8	Not Supported
Time Stamp	4 (Date) + 6 (Time6) + 2 (UTC Offset)	Not Supported
Userid Indicator	1	Not Supported
Reserved for IMS	3	Supported

# Appendix C: AIB Mask

The following shows the field length of the AIB Mask provided by IBM mainframe's IMS/DC. It includes the field lengths and support status in the OSI product.

Descriptor	Length (Byte)	Support
AIB identifier	8	Supported
DFSAIB allocated length	2	Not Supported
Subfunction code	8	Supported
Resource name 1	8	Supported
Resource name 2	8	Not Supported
Reserved	8	Supported
Maximum output area length	4	Supported
Output area length used	4	Supported
Resource field	4	Not Supported
Optional area length	4	Not Supported
Reserved	4	Not Supported
Return code	4	Supported
Reason code	4	Supported
Error code extension	4	Not Supported
Resource address 1	4	Support
Resource address 2	4	Not Supported
Resource address 3	4	Not Supported
User defined token	16	Not Supported
Return token	8	Not Supported
Reserved	16	Not Supported

# Appendix D: DL/I Call

To use the system service function in application programs, use DL/I calls. This appendix describes the system service and transaction management calls that OSI supports.



For detailed information about DL/I calls, refer to *IBM IMS Version 7 Application Programming: Transaction Manager*.

## D.1. System Service DL/I Calls

### D.1.1. CHKP (basic)

The basic CHKP call is used to specify a recovery point when an application program applies changes or terminates abnormally.

- Format

```
>>-CHKP---+i/o pcb---+---i/o area-----><
```

- Parameter

Parameter	Description
i/o pcb	Specifies the IO-PCB that will be sent as the first PCB of the applicable program. This is an input and output parameter.
i/o area	Specifies an 8-byte checkpoint ID as an input parameter.

### D.1.2. CHKP (symbolic)

The symbolic CHKP call is used to specify a recovery point when an application program attempts to apply changes or terminates abnormally. The Extended Restart (XRST) call is used to restart the program that has terminated abnormally. Up to 7 data entries can be saved and restored when a program restarts.

- Format

```
>>-CHKP---+i/o pcb---+---i/o area length--i/o area--+-----+--><  
|-----|  
| V |  
|-----|  
'---area length--area+--'
```

- Parameter

Parameter	Description
i/o pcb	Specifies the IO-PCB that will be sent as the first PCB of the applicable program. This is an input and output parameter.
i/o area	Specifies an 8-byte checkpoint ID as an input parameter.
area length	Specifies the length of the area as 4 bytes. This is an input parameter.
area	Stores data of the specified area length. This is an input parameter.

### D.1.3. INIT

In IBM mainframe IMS/DC, the INIT call allows an application program to verify the data availability status code by checking the data availability of each DB PCB.

In OSI, an INIT call is used only to set a flag for checking whether the application program has invoked the INIT call.

- Format

```
>>-INIT---i/o pcb---i/o area-----<<
      '---aib-----'
```

- Parameter

Parameter	Description
i/o pcb	Specifies the IO-PCB that will be sent as the first PCB of the applicable program. This is an input and output parameter.
aib	Specifies the aib as an input and output parameter.
i/o area	Specifies the I/O area that is sufficient for sending segments. This is an input and output parameter.

### D.1.4. INQY

The INQY call is used to request information containing the execution environment, destination type and status, and session status. In OSI, AIBSFUNC can be set to ENVIRON or FIND.

- Format

```
>>-INQY---aib---i/o area-----<<
```

- Parameter

Parameter	Description
aib	Specifies the aib as an input and output parameter.
i/o area	Specifies the I/O area that is sufficient for sending segments. This is an input and output parameter.

#### D.1.4.1. ENVIRON

The ENVIRON subfunction obtains the application's execution environment information.

- Output

Type	Length	Value	Description
IMS ID	8		IMS ID.
Release level	4		Currently not supported.
CTL Reg Type	8	DB/DC	Type of the control region.
App Reg Type	8	MPP, BMP	Type of the running application.
Reg ID	4		Currently not supported.
App Pgm Name	8		Name of the running application.
PSB Name	8		Name of the running PSB.
Tran Name	8		Name of the running transaction.
User ID	8		Currently not supported.
Group Name	8		Currently not supported.
Stat Grp Indicator	4		Currently not supported.
Addr of Recovery Token	4		Currently not supported.
Addr of the App Param Str	4		Currently not supported.
Shared Queues Indicator	4		Currently not supported.
User ID of Address Space	8		Currently not supported.
User ID Indicator	1		Currently not supported.
Res. Recov. Serv. Ind.	3		Currently not supported.
catalog enablement indi.	8		Currently not supported.

#### D.1.4.2. FIND

In IBM mainframe IMS/DC, the FIND subfunction returns the PCB address for the requested PCB name.

In OSI, the FIND subfunction retrieves information about the application's execution environment.

## D.1.5. LOG

In IBM mainframe IMS/DC, the LOG call is used to store information to the IMS system log.

In OSI, the LOG call stores information in the OFM\_OSI\_LOG table. Currently, OSI only supports AIB.

- Format

```
>>-LOG---+aib---+-----<<
```

- Parameter

Parameter	Description
aib	Specifies the aib as an input and output parameter.

## D.1.6. ROLB

The ROLB call is used by application programs to cancel output messages and database changes.

- Format

```
>>-ROLB---+i/o pcb---+-----<<
```

- Parameter

Parameter	Description
i/o pcb	Specifies the IO-PCB that will be sent as the first PCB of the applicable program. This is an input and output parameter.

- Limitations

- OSC messages do not support ROLB calls.

## D.1.7. SYNC

The SYNC call is used for the application program to commit. It only applies in the BMP program.

- Format

```
>>-SYNC---+i/o pcb---+-----<<  
      '-aib-----'
```

- Parameter

Parameter	Description
i/o pcb	Specifies the IO-PCB that will be sent as the first PCB of the applicable program. This is an input and output parameter.
aib	Specifies the aib as an input and output parameter.

## D.1.8. XRST

The XRST call is used to restart application program from a symbolic checkpoint of a previous execution of the program.

- Format

```
>>-XRST---+i/o pcb---+---i/o area length--i/o area---+-----+<
                                     |-----|
                                     |  V  |
                                     |-----|
                                     ---area length--area+--<
```

- Parameter

Parameter	Description
i/o pcb	Specifies the IO-PCB that will be sent as the first PCB of the applicable program. This is an input and output parameter.
i/o area length	This is an input and output parameter and currently not used
i/o area	Specifies a checkpoint ID as an input parameter.
area length	Specifies the length of the area as 4 bytes. This is an input parameter.
area	Restores data of the specified area length. This is an input and output parameter.

## D.2. Transaction Management DL/I Calls

### D.2.1. CHNG

The CHNG (Change) call is used to change where the message segment is sent. The destination of the modifiable ALT-PCB can be specified with logical terminal, LU 6.2 descriptor, or transaction codes using CHNG call.

- Format

```
>>-CHNG---+alternate pcb+--destination name-----<
```

- Parameter

Parameter	Description
alternate pcb	To use CHNG call, specify the modifiable ALT-PCB as an input and output parameter.
destination name	Specifies the destination of the message segment that will be changed.

## D.2.2. CMD

The CMD call allows application programs to input IMS commands. It can be used with the GCMD call to send commands to the OSI command server and receive responses. It retrieves the first segment of a command response message.

- Format

```
>>-CMD---i/o pcb---i/o area-----<
```

- Parameter

Parameter	Description
i/o pcb	Specifies the IO-PCB that will be sent as the first PCB of the applicable program. This is an input and output parameter.
i/o area	Specifies the I/O area that is sufficient for sending segments. This is an input and output parameter.

## D.2.3. GCMD

The GCMD call retrieves the subsequent segment of a command response message processed by a CMD call.

- Format

```
>>-GCMD--i/o pcb---i/o area-----<
```

- Parameter

Parameter	Description
i/o pcb	Specifies the IO-PCB that will be sent as the first PCB of the applicable program. This is an input and output parameter.
i/o area	Specifies the I/O area that is sufficient for sending segments. This is an output parameter.

## D.2.4. GN

When an input message contains more than one segment, a GU call retrieves the first segment of the message and a GN call retrieves the subsequent segments.

- Format

```
>>-GN---i/o pcb---i/o area-----<<
```

- Parameter

Parameter	Description
i/o pcb	Specifies the IO-PCB that will be sent as the first PCB of the applicable program. This is an input and output parameter.
i/o area	Specifies the I/O area that is sufficient for sending segments. This is an output parameter.

## D.2.5. GU

The GU call retrieves the first segment of a message.

- Format

```
>>-GU---i/o pcb---i/o area-----<<
```

- Parameter

Parameter	Description
i/o pcb	Specifies the IO-PCB that will be sent as the first PCB of the applicable program. This is an input and output parameter.
i/o area	Specifies the I/O area that is sufficient for sending segments. This is an output parameter.

## D.2.6. ISRT

The ISRT call is used to send a message segment to a specific destination. The destination is specified by IO-PCB and alternate PCB.

- Format

```
>>-ISRT---i/o pcb-----i/o area---+-----+-----<<  
      '-alternate pcb-'          '-mod name-'
```

- Parameter

Parameter	Description
i/o pcb	Specifies the IO-PCB that will be sent as the first PCB of the applicable program. This is an input and output parameter.
alternate pcb	Specifies the PCB as an input and output parameter.
i/o area	Specifies the I/O area that is sufficient for sending segments. This is an output parameter.
mod name	Specifies the MOD name as an 8-byte input parameter for sending output messages. The output message is formatted according to the specified MOD.

## D.2.7. PURG

The PURG call is used to output message segments sent by the ISRT call. When more than one message segments are sent in application programs, it is used to purge the previous messages or complete sending a message.

- Format

```
>>-PURG--+i/o pcb-----+--+-----+-----><
      '-alternate pcb-' '-i/o area--+-----+-'
                               '-mod name-'
```

- Parameter

Parameter	Description
i/o pcb	Specifies the IO-PCB that will be sent as the first PCB of the applicable program. This is an input and output parameter.
alternate pcb	Specifies the PCB as an input and output parameter.
i/o area	Specifies the I/O area that is sufficient for sending segments. This is an output parameter.
mod name	Specifies the MOD name as an 8-byte input parameter for sending output messages. The output message is formatted according to the specified MOD.  The PURG call specifies the MOD name for the first segment of the output message.