

Administrator'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. OSI System Overview	1
1.2. OSI System Components	2
1.3. OSI System Structure	2
1.3.1. OSI System Server (Control Region)	3
1.3.2. OSI User Server (Dependent Region)	4
1.3.3. OSIOMSVR	4
1.3.4. OSI OTMA Server	5
1.3.5. OSI Administrator Server	5
1.3.6. OpenFrame Gateway	5
1.3.7. OpenFrame OSC	6
1.3.8. System and User Server DB Sessions	6
2. System Environment Configuration	9
2.1. Overview	9
2.2. Configuring System Configuration Files	9
2.2.1. openframe_osi.conf	9
2.2.2. osi.ofsys.seq	10
2.2.3. Configuring TCache	10
2.3. Configuring Libraries	11
2.3.1. DBDLIB	12
2.3.2. DFSRESLB	13
2.3.3. FORMAT, FORMATA and FORMATB	15
2.3.4. IMSACB, IMSACBA and IMSACBB	18
2.3.5. MODBLKS, MODBLKSA and MODBLKSB	19
2.3.6. MODSTAT	22
2.3.7. PSBLIB	23
2.3.8. RESLIB	24
2.3.9. STEPLIB	24
2.4. Configuring Storage	25
2.4.1. MQ	25
2.4.2. REGION	26
2.4.3. RTSD	27
3. System Server Configuration	28
3.1. Overview	28
3.2. Start/Stop Server (osiomsvr)	28
3.2.1. Tmax Environment Configuration	28
3.3. Schedule Server (osisschd)	29
3.3.1. Tmax Environment Configuration	29

3.4. Command Server (osicmdsv)	29
3.4.1. Tmax Environment Configuration	30
3.5. OTMA Server (osiotmasvr)	30
3.5.1. Tmax Environment Configuration	30
3.6. Administrator Server (osiofmgr)	31
3.6.1. Tmax Environment Configuration	31
4. User Server Configuration	32
4.1. Overview	32
4.2. Server Group Configuration	32
4.2.1. General Configuration	32
4.2.2. XA Configuration	33
4.3. MPP User Server	34
4.4. BMP User Server	35
5. System Operation	36
5.1. Starting and Shutting Down OSI	36
5.1.1. Starting OSI	37
5.1.2. Shutting Down OSI	40
5.1.3. Restarting OSI	42
5.2. Log Management	42
5.2.1. System Server and MPP User Server	42
5.2.2. BMP User Server	44
Appendix A: IMSBATCH Procedure	45
Appendix B: Printer Facilities	46
B.1. Overview	46
B.2. Display HardCopy	46
B.3. SCS-DATA Printer	46
Appendix C: Resource Tables	48

1. Introduction

This chapter introduces the OpenFrame OSI (Online Server type I) system and its components and describes its entire structure.

1.1. OSI System Overview

OpenFrame OSI is one of many products that make up OpenFrame (a rehosting solution). It allows online applications running in a mainframe to run in an open system (Unix).

The following questions may arise when considering migration of applications running in an existing mainframe, to an open system.

- How can the open system provide the same performance and reliability as the existing Mainframe ?
- How easily can migration be performed ?

OSI addresses these two issues by using Tmax TP-Monitor and the migration tool.

To solve performance and reliability issues, OSI uses the Tmax engine (a TP-monitor) which has proven reliability and performance in open systems. Therefore, OSI enjoys the following benefits of Tmax.

- Convenient process management

In OSI, user-created processes are managed by Tmax from start-up to shut-down and use the various monitoring information that Tmax provides. This allows processes to be conveniently managed.

- Large-scale transaction

Tmax contains built-in scheduling and service queue management functions for large-scale transaction processing.

Tmax is suitable for mission-critical systems in an open environment. OSI, which is based on Tmax, also reliably supports large-scale transactions.

- Unrestricted integration in an open environment

Integration is not the most important issue arising immediately after rehosting from an IBM mainframe to an open system environment. However, there are many cases in which systems in operation have to be extended or integrated with other systems. In these situations, Tmax seamlessly integrates with commercial TP-monitors such as Tuxedo which conforms to a different X/Open DTP model, and therefore OSI has an excellent advantage over other rehosting solutions.

This feature of Tmax allows an easy extension of business applications written with OSI. Furthermore, it can also seamlessly integrate with the web environment by linking with JEUS (TmaxSoft's Web Application Server).

OSI provides various tools to manage applications in an open system in the same way as in a mainframe, by migrating user programs and their associated resources. An interface similar to DL/I is provided to allow user programs operating in Mainframe's IMS/DC, MPP (Message Processing Program) and BMP (Batch Message Processing) to run in the same way in OSI.

For more information about MPP and BMP in OSI, refer to [OSI User Server \(Dependent Region\)](#).

1.2. OSI System Components

OSI is composed of two modules: the OSI system server (system servers such as the schedule server and command server) which is responsible for the important functions of the OSI system and the OSI user server (user servers such as the MPP and BMP servers) which runs user applications.

- OSI System Server (Control Region)

A system module required for operating the OSI system.

The functions of the OSI system servers include: message scheduling, message queue management, message transformation via MFS, DB integration, and system and user command processing.

- OSI User Server (Dependent Region)

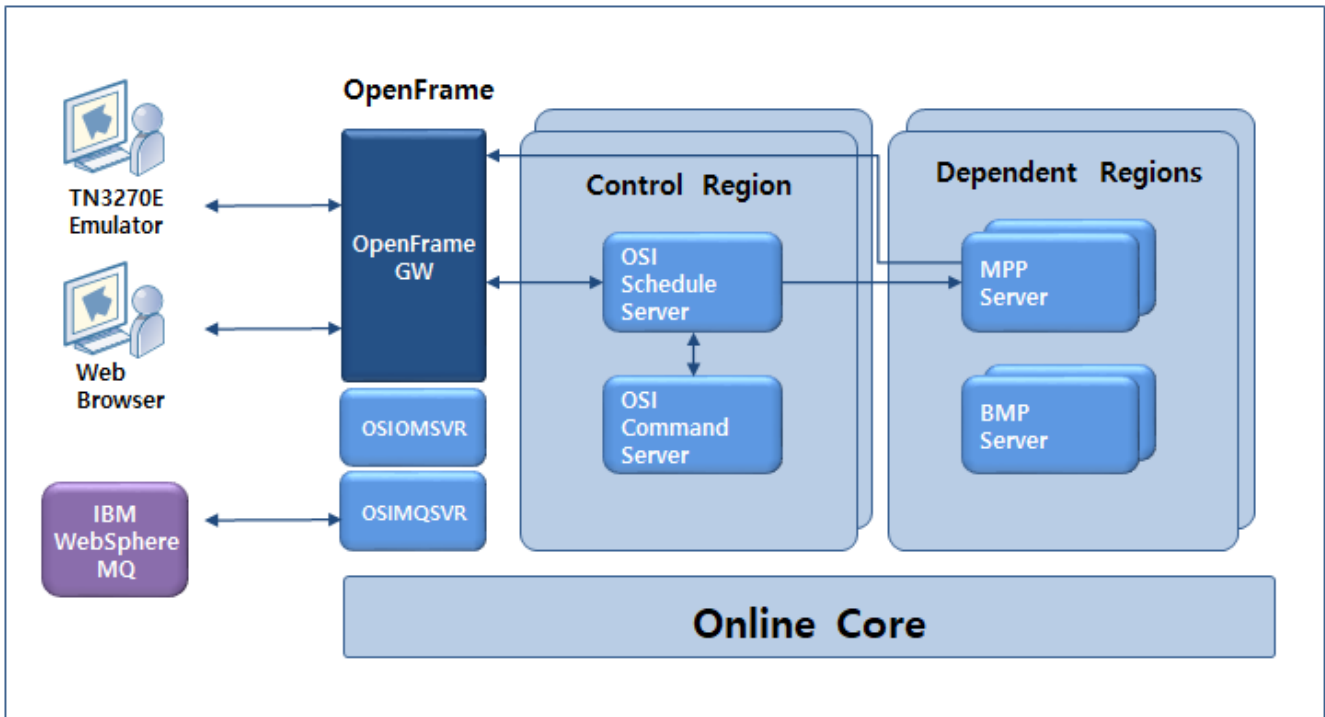
Users run user application programs by starting the processes (in the Tmax server) themselves through JCL or by executing the "/START REGION" command.



1. The MPP server is provided under the name of OSIMPPSVR, and the BMP server is provided as the osibmpsv module.
2. Starting from the current version, TN3270 Gateway is not supported by OSI, but replaced by OpenFrame Gateway, which provides all the existing features. For more information, refer to *OpenFrame Gateway Administrator's Guide*.

1.3. OSI System Structure

Generally, running application programs requires a base system for operating these programs in terms of both system server and user server. The OSI structure for such base system is as follows:



OSI Structure

The system server (Control Region) is a region corresponding to the OSI system module and contains various configurations including the server which is required for running application servers and application programs in the OSI system itself. The user server (Dependent Region) is a part of the OSI user module and divided into two types of application server provided in the OSI system: MPP and BMP. The osiomsvr (Online Manager) server is required for activating both system server and user server. The OSI system also includes the system engine area, Online Core (Tmax).

1.3.1. OSI System Server (Control Region)

The OSI system server (control region) consists of two servers: the schedule server (osisschd) and command server (osicmdsv). Both servers are provided under the type of Tmax UCS server.

- Schedule Server (osisschd)

Determines the integrity of the requested message (determines whether the message is valid), stores (backup) and passes (forwarding) the message at the same time. In the current version, the schedule server only checks for the message's integrity and forwards the message to its destination. In the case of a message received from the terminal (screen), a data transformation through MFS is performed and then the message is sent to the MPP server.

Classification	Description
Message storage and forwarding	Backs up the requested message to the Message Queue (MQ) table, and then forwards the message to the MPP or command server, or to the emulator connecting to OpenFrame Gateway, depending on the type of the destination (command, transaction...).

Classification	Description
MFS transformation	Unifies the data format defined in the screen information and application, between MPP and TN3270 emulator (OpenFrame Gateway is present in between), and converts data with different code pages, from ASCII to EBCDIC or the other way round.

- Command Server (osicmdsv)

Processes all commands required for operating the OSI system. Commands can be entered through a terminal, imscmd tool, or a DL/I CMD call. The result of command processing are recorded in `${OPENFRAME_HOME}/log/cmd/imscmd_{DATE}.log`.



For more information about the commands processed by the command server, refer to *OpenFrame OSI Command Reference Guide*.

1.3.2. OSI User Server (Dependent Region)

Operating user-written application programs requires a user to prepare a dedicated server module during the system configuration phase, in addition to the default system module provided by OSI. A user-prepared server module means, conceptually, the region corresponding to the OSI user server (dependent region) of IMS/DC. Generally, at least one user module is prepared for each of the dependent regions which have been run in the IBM mainframe.

The following describes the user servers used in operating OSI.

Classification	Description
MPP User Server	Corresponds to an MPP region in IMS/DC. OSI runs MPP servers at the level of message classes. Therefore, an MPP user server features up to one-to-four correspondence per MPP region running in the existing IMS/DC.
BMP User Server	Allows user programs running in the BMP region of IMS/DC to be managed.

The process for preparing the user server can be started by using JCL or the OSI command. For more information about relevant user server environment configurations, refer to [User Server Configuration](#).

1.3.3. OSIOMSVR

OSIOMSVR is a server for starting and stopping the OSI system and MPP user servers. If you execute JCL or the `"/START REGION"` command, OSIOMSVR internally stores information about the regions required for startup to the database and then starts servers through `tmboot`. If you execute `"/STOP REGION"` or `"/CHECKPOINT FREEZE"`, OSIOMSVR deletes the regions information that has been

internally stored in the database, and then shuts down servers through tmdown.

OSIOMSVR must be booted ahead of the regions and can be started or stopped along with the OpenFrame core (Tmax/Base/Batch) servers by using the osiboot or osidown tool.



The BMP user server must be directly started/stopped without using OSIOMSVR.

1.3.4. OSI OTMA Server

OSI OTMA is a server allowing connection with IBM MQ and providing the OTMA feature. The MQ feature can be enabled by specifying a certain option but does not affect the system behavior if disabled.

The following describes different MQ interfaces (MQI) used by OTMA servers.

Name	Description
MQCONN	Connects to an IBM MQ queue manager.
MQOPEN	Enables access to an object.
MQGET	Gets messages from a local queue that has been opened by MQOPEN.
MQPUT	Sends messages to a local queue that has been opened by MQOPEN.
MQCLOSE	Disables access to an object.
MQDISC	Disconnects from an IBM MQ queue manager.



For more information about how to configure and use the IBM MQ feature of the OTMA server, refer to *OpenFrame OSI Configuration Guide*.

1.3.5. OSI Administrator Server

OSI Administrator Server interacts with the OpenFrame Manager product and provides OSI management functionality.



For more information about OSI management functionality, refer to OpenFrame Manager "User Guide".

1.3.6. OpenFrame Gateway

OpenFrame Gateway (OpenFrame GW) is found between the OSI system server (control region) and TN3270/TN3270E emulators, or between terminals that are connected through web browsers.

OpenFrame GW is a server that sends transaction requests to the OSI system server (control region)

and manages information about terminal connections and system servers (control region). It is also referred to as VTAM (Virtual Terminal Access Method) Gateway.

The following are the main functions of OpenFrame GW.

- Supporting TN3270 protocol
- Managing the Application Identifier (APPLID) information (APPLID indicates a terminal connecting to OpenFrame GW and the OSI system server (control region).)
- Supporting IP-LU mapping



For more information about how to configure and use OpenFrame GW, refer to *OpenFrame Gateway Administrator's Guide*.

1.3.7. OpenFrame OSC

OpenFrame OSC (OSC) is one of the products of OpenFrame, a mainframe rehosting solution. It facilitates the operation of CICS workloads in an open system environment through a migration process.

OSC can communicate with OSI using the CICS SEND and CICS RECEIVE commands, and messages sent from OSC are sent to the schedule server (osisschd).



For more information about how to configure and use OpenFrame OSC, refer to *OpenFrame OSC Administrator's Guide*.

1.3.8. System and User Server DB Sessions

The OSI system servers and region servers use a database for system metadata management and integration with OpenFrame HiDB. Each server maintains the required number of DB sessions.

- **osiomsvr**

The osiomsvr server establishes the following DB sessions by default:

No.	Session	Type	Purpose
1	System connection	ODBC	OpenFrame meta

- **osiotmasvr**

The osiotmasvr server establishes the following DB sessions by default:

No.	Session	Type	Purpose
1	System connection	ODBC	OpenFrame meta

No.	Session	Type	Purpose
2	OpenFrame HiDB	ESQL	HiDB connection

- **osiofmgr**

The osiofmgr server establishes the following DB sessions by default:

No.	Session	Type	Purpose
1	System connection	ODBC	OpenFrame meta

1.3.8.1. System Servers

- **osisschd**

The osisschd server establishes the following DB sessions by default:

No.	Session	Type	Purpose
1	System connection	ODBC	OpenFrame meta
2	OpenFrame HiDB	ESQL	HiDB connection

- **osicmdsv**

The osicmdsv server establishes the following DB sessions by default:

No.	Session	Type	Purpose
1	System connection	ODBC	OpenFrame meta
2	OpenFrame HiDB	ESQL	HiDB connection

1.3.8.2. User Servers

- **OSIMPPSVR**

The OSIMPPSVR server establishes the following DB sessions by default:

No.	Session	Type	Purpose
1	System connection	ODBC	OpenFrame meta
2	OpenFrame HiDB	ESQL	HiDB connection

- **osibmpsv**

The osibmpsv server establishes the following DB sessions by default:

No.	Session	Type	Purpose
1	System connection	ODBC	OpenFrame meta
2	OpenFrame HiDB	ESQL	HiDB connection



For region servers, the number of database sessions may increase depending on server group settings For more information, refer to [User Server Configuration](#).

2. System Environment Configuration

This chapter describes the environment configuration methods required for OSI system operation.

2.1. Overview

After the installation of the OSI system is complete, users must change the default configuration of items for the OSI system environment, as required by the system.

Configure the following for the OSI system environment.

- [System configuration files](#)
- [Libraries](#)
- [Storage](#)

2.2. Configuring System Configuration Files

Among system configuration values that are required by the OSI system, the items that cannot be dynamically applied are mostly described in the configuration file and the values specified are applied when the system is started.

2.2.1. openframe_osi.conf

To operate the OSI system, configure the following subjects in the openframe_osi.conf file, and then apply them to the OpenFrame system by using the **ofconfig** tool.

Subject	Description
osi	Common reference item for all modules of OSI. The values specified in this item are applied to all modules of OSI.
osi.[<i>IMSID</i>]	Identification for each IMS system in OSI. The values specified in this item are applied to all modules of the respective IMS system. <i>IMSID</i> must be four alphanumeric characters.
osi.[<i>osiotmasvrname</i>]	Setting for OTMA functionality between OSI and IBM MQ. Each OTMA server requires one subject item. <i>osiotmasvrname</i> must match with the server name specified in the MQ user server configuration.
ssm.[<i>IMSID</i>].[<i>SSM</i>]	Identification applied in the Tmax XA server for each IMS system to use MultipleRM in OSI.



1. For more information about subjects and settings for OpenFrame configuration, refer to *OpenFrame OSI Configuration Guide*.

2. For more information about the usage and functions of the ofconfig tool, refer to *OpenFrame Base Tool Reference Guide*.

2.2.2. osi.oftsys.seq

You can choose which Tmax server to run with osiboot in OSI, by specifying the name of Base/Batch/TACF servers including the OTMA server, but except for region servers.

The following is a sample file of osi.oftsys.seq.

```
#BASE
TPFMAGENT
ofrsasvr
ofrlhsrv
ofrdmsvr
ofrdsedt
ofrcmsvr
ofruisvr
ofrsmlog

#BATCH
obmjmsvr
obmj schd
obmjinit
obmjhist
obmjspb
ofrpmsvr
obmtmgr
obmjtimr

#TACF
tmsvr

#OSI
#IMSAOTMA
```

2.2.3. Configuring TCache

For supporting multi-node clustering, OSI manages that information that needs to be shared between regions, including runtime system definition (RTSD), in DB tables. This guarantees enhanced read performance by using TCache to cache the data accessed from DB table into the shared memory. TCache needs to be configured for each node, involving configuration of shared memory key and resource for each region.



For more information about default setting, refer to *Tmax TCache Guide*.

The following is an example of pfmtcache.conf. LOCK_WAITTIME is set based on the TCache version in use.

```

# the configuration file of TCACHE
SHMKEY=0x70005          # the key of shared memory
IPCPerm=0600           # permission of the shared memory
SIZE_LOCAL=32          # L1 cache size in kilo-bytes
LIB_PTHREAD=libpthread.so # the pthread library file name
INVALIDATE_TYPE=0      # Invalidate type 0:multi-node 1:multi-domain
AGENT_SVC=SPFMAGENT    # multi-node sync. svc SPFMAGENT|2 = SPFMAGENT01, SPFMAGENT02

# lock wait time for earlier than r11724
# LOCK_WAITTIME=1000    # read, write lock wait time (default: 1000)

# lock wait time for r11724 and later
# WRITE_LOCK_WAITTIME=1000 # write lock wait time (default: 1000, recommed larger value
than READ_LOCK_WAITTIME)
# READ_LOCK_WAITTIME=1000 # read lock wait time (default: 0)

# cache for OSI Region Status
CACHE_NAME=OFM_OSI_REGION_STATUS # table name
SIZE_MEM=32                      # the total cache memory size in kilo-bytes
SIZE_HASH=32                     # the number of hash key (MAX=65536)
SIZE_KEY=4                       # the number of digits of the index column
SIZE_REC=8                       # the size of a single record in bytes
INV_TIMEOUT=0                   # invalidation timeout in sec

# cache for IMSA Region
CACHE_NAME=IMSA                  # IMSID
SIZE_MEM=65535                  # the total cache memory size in kilo-bytes
SIZE_HASH=32                    # the number of hash key (MAX=65536)
SIZE_KEY=13                     # the number of digits of the index column
SIZE_REC=289                    # the size of a single record in bytes
INV_TIMEOUT=0                   # invalidation timeout in sec

#cache for OFM_OSI_DBD
CACHE_NAME=OFM_OSI_DBD          # the name of cache
SIZE_MEM=65535                  # the total cache memory size in kilo-bytes
SIZE_HASH=32                    # the number of hash key (MAX=65536)
SIZE_KEY=64                     # the number of digits of the index column
SIZE_REC=2048                   # the size of a single record in bytes
INV_TIMEOUT=0                   # invalidation timeout in sec

#cache for OFM_OSI_PSB
CACHE_NAME=OFM_OSI_PSB          # the name of cache
SIZE_MEM=65535                  # the total cache memory size in kilo-bytes
SIZE_HASH=32                    # the number of hash key (MAX=65536)
SIZE_KEY=64                     # the number of digits of the index column
SIZE_REC=1024                   # the size of a single record in bytes
INV_TIMEOUT=0                   # invalidation timeout in sec

```

2.3. Configuring Libraries

All information required for system operation must be prepared before booting the system. This section describes the process of preparing such information by type. Most of the information is saved as records in RDB tables, and others such as MDA or MFS are stored in datasets or Unix files.

The following libraries are required in OSI operation.

Library	Description
DBDLIB	Stores DBD information for using OpenFrame HiDB.
DFSRESLB	Stores information needed for dynamic usage of datasets.
FORMAT	Staging library which stores MFS information for mapping support in OSI.
FORMATA / FORMATB	Stores MFS information for mapping support in OSI and is actually used online.
IMSACB	Staging library which stores ACB information for integrated management of PSB and DBD.
IMSACBA/IMSACBB	Stores ACB information for integrated management of PSB and DBD and is actually used online.
MODBLKS	Staging library which stores system information required for OSI operation.
MODBLKSA / MODBLKSB	Stores system information for OSI operation and is actually used online.
MODSTAT	Stores the current ACTIVE dataset information for the dataset that supports dynamic change.
PSBLIB	Stores PSB information for using application programs.
RESLIB	Stores MDA information needed for dynamic usage of datasets. (Same as DFSRESLB)
STEPLIB	Directory for shared object form execution module management of programs that application developers wrote.



1. For more information about dataset usage and PDS, refer to *OpenFrame Base Dataset Guide*.
2. For more information about the usage of IDCAMS, refer to *OpenFrame Batch Utility Reference Guide*.
3. For more information about registering and managing DBD, PSB, ACB and MDA information as well as creating OpenFrame HiDB, refer to *OpenFrame HiDB Guide*.
4. For more information about tools used during library configuration, refer to *OpenFrame OSI Tool Reference Guide*.

2.3.1. DBDLIB

DBD is a set of macro parameter statements that define the characteristics of OpenFrame HiDB and the relationships between segments, fields, and segment types within the database structure, access methods and databases.

Creating Datasets

Using DBDLIB requires a procedure for creating datasets during the system preparation process. DBDLIB is composed in the PDS form and the individual environment configuration files are stored as DBDLIB members. To create DBDLIB in OSI, a tool named **pdsgen** is used.

The following example creates IMS.DBDLIB in a volume named DEFVOL by using pdsgen.

```
$ pdsgen IMS.DBDLIB DEFVOL -f LB -l 32760
pdsgen version 7.1.0(0) oframe@tmax:ofsrc/base(#1) 2020-06-29 17:30:26
PDS Dataset Generation Program

pdsgen: *** PDS IMS.DBDLIB is created.
```

Registering DBD Information

Using DBD in OSI requires a tool called **dbdgen**, to register DBD information.

The following example registers DBD information with a file named OIVPIDBD which contains the DBD control statement, by using dbdgen.

```
$ dbdgen OIVPIDBD
dbdgen version 7.2.0(0) oframe@tmax:ofsrc/ims(#3) 2020-08-08 16:21:16
Database Description Block Generation Program

dbdgen: force flag on; the tool overwrites existing data
dbdgen: 1 files are requested in total
-----
dbdgen: processing DBD script "OIVPIDBD"
-----

dbdgen: processing DBD "OIVPIDBD"
dbdgen: removing existing DBD OIVPIDBD metadata
dbdgen: successfully processed DBD "OIVPIDBD"
dbdgen: successfully processed for the requested DBDs (total 1)
```



In the current version, executing dbdgen stores the database schema information described in the DBD script to metatables, and DBDLIB stores no actual data.

2.3.2. DFSRESLB

Running a program using PSB resources requires a method of sending the PSB information and the dataset information defined in DBD, to the program to run.

OpenFrame Batch runs an application program by using JCL, at which point the JCL DD statements send the PSB information and DBD dataset information to the program. On the other hand, OSI runs

multiple programs in a single region, and the JCL required to startup the region cannot contain all DD statements. Therefore, OSI is required to use a different method than OpenFrame Batch to send necessary information to the program. Hence, OSI stores information necessary for dynamic usage of datasets, in DFSRESLB.



For more information about dynamic DD allocation, refer to [IBM IMS DFSMDA macro](#).

Creating Datasets

For DFSRESLB, datasets need to be created in the system preparation process. DFSRESLB is composed in the PDS dataset format and individual configuration files are stored in the member format of DFSRESLB. To create DFSRESLB in OSI, use the **pdsgen** tool.

The following example creates IMS.RESLIB in a volume named DEFVOL, by using pdsgen.

```
$ pdsgen IMS.RESLIB DEFVOL -f LB -l 32760
pdsgen version 7.1.0(0) oframe@tmax:ofsrc/base(#1) 2020-06-29 17:30:26
PDS Dataset Generation Program

pdsgen: *** PDS IMS.RESLIB is created.
```

Registering MDA Information

To use MDA in OSI, the MDA information has to be registered using the **imsdalloc** tool.

The following example registers the MDA information to DFSRESLB, with a file named OIVPIDBD which contains the script defining MDA.

<OIVPIDBD>

```
DFSMDA TYPE=DATABASE,DBNAME=OIVPIDBD
DFSMDA TYPE=DATASET,DSNAME=IMS.TEST.OIVPI,DDNAME=OIVPI
END
```

Register the MDA information by using imsdalloc.

```
$ imsdalloc -l IMS.RESLIB -v DEFVOL OIVPIDBD
imsdalloc version 7.2.0(0) oframe@tmax:ofsrc/ims(#3) 2020-08-08 16:21:16
Dynamic Allocation Block Generation Program

IMSDALOC FCOUNT=1,RESLIB=IMS.RESLIB,VOLSER=DEFVOL
-----
*** processing filepath="OIVPIDBD"
-----
mdaparser: *** dfsmda_statement matched!
mdaparser: *** dfsmda_statement matched!
```

```

mdaparser: *** end_statement matched!
mdaparser: *** mda_generation finished!!!
-----
*** ims_parse_mda("OIVPIDBD") success.
-----
MDA TYPE=DATABASE,DBNAME=OIVPIDBD
DATASET TYPE=DATASET,DDNAME=OIVPI,DSNAME=IMS.TEST.OIVPI,DISP=
-----
*** ims_print_mda("OIVPIDBD") success.
-----
PROGRAM COMPLETED SUCCESSFULLY.

```

2.3.3. FORMAT, FORMATA and FORMATB

FORMAT is a library that stores the information supporting the mapping function in OSI.

The mapping function here, refers to connecting fields and applications that are defined in the format by using FORMAT. Prepare three datasets in the same way as in MODBLKS and IMSACB.

Creating Datasets

Using FORMAT requires creating datasets during the system preparation process.

FORMAT is composed in the PDS format and the individual environment configuration files are stored as MFSLIB members. To create FORMAT in OSI, use the **pdsgen** tool.

The following example creates OSI.IMSA.MFSLIB in a volume named DEFVOL, by using pdsgen.

```

$ pdsgen OSI.IMSA.MFSLIB DEFVOL -f LB -l 32760
pdsgen version 7.1.0(0) oframe@tmax:ofsrc/base(#1) 2020-06-29 17:30:26
PDS Dataset Generation Program

pdsgen: *** PDS OSI.IMSA.MFSLIB is created.

```

Registering Map Information

The following example registers map information to OSI.IMSA.MFSLIB, by using a file named OIVP001.TXT.

<OIVP001.TXT>

```

      PRINT ON,NOGEN
*****
      TITLE 'FORMAT SET FOR OPENFRAME ONLINE IVP'
*****
OIVP001  FMT
          DEV  TYPE=(3270,2),                X
              FEAT=IGNORE,                  X
              DSCA=X'00A0',                  X

```

	PFK=(PFKFIELD,	X
	12='/FOR OIVP0060')	
	DIV TYPE=INOUT	
	DPAGE CURSOR=((7,40)),	X
	FILL=PT	
CURDATE	DFLD POS=(1,2),	X
	LTH=8,	X
	ATTR=(PROT,ALPHA,NORM,NOMOD)	
	DFLD '** WELCOME TO OPENFRAME ONLINE **',	X
	POS=(2,24),	X
	ATTR=(PROT,ALPHA,HI,NOMOD),	X
	EATTR=YELLOW	
	DFLD '-----X	
	-----',	X
	POS=(3,2),	X
	ATTR=(PROT,ALPHA,NORM,NOMOD),	X
	EATTR=YELLOW	
	DFLD '** OSI INSTALLATION VERIFICATION PROCEDURE **',	X
	POS=(4,17),	X
	ATTR=(PROT,ALPHA,HI,NOMOD),	X
	EATTR=GREEN	
	DFLD 'CODE ',	X
	POS=(7,34),	X
	ATTR=(PROT,ALPHA,NORM,NOMOD)	
CODE	DFLD POS=(7,40),	X
	LTH=4,	X
	ATTR=(NOPROT,ALPHA,NORM,MOD),	X
	EATTR=HREV	
	DFLD ' ',	X
	POS=(7,45),	X
	ATTR=(PROT,ALPHA,HI,NOMOD)	
	DFLD 'INQR INQUIRY ACCOUNT INFORMATION',	X
	POS=(10,23),	X
	ATTR=(PROT,ALPHA,HI,NOMOD)	
	DFLD 'INSR INSERT NEW ACCOUNT',	X
	POS=(11,23),	X
	ATTR=(PROT,ALPHA,HI,NOMOD)	
	DFLD 'UPDT UPDATE ACCOUNT INFORMATION',	X
	POS=(12,23),	X
	ATTR=(PROT,ALPHA,HI,NOMOD)	
	DFLD 'DELT DELETE ACCOUNT',	X
	POS=(13,23),	X
	ATTR=(PROT,ALPHA,HI,NOMOD)	
	DFLD '-----X	
	-----',	X
	POS=(18,2),	X
	ATTR=(PROT,ALPHA,NORM,NOMOD),	X
	EATTR=YELLOW	
ERRMSG	DFLD POS=(19,2),	X
	LTH=79,	X
	ATTR=(PROT,ALPHA,NORM,NOMOD),	X
	EATTR=TURQ	
	DFLD 'ENTER APPROPRIATE CODE.',	X
	POS=(20,2),	X
	ATTR=(PROT,ALPHA,NORM,NOMOD),	X
	EATTR=PINK	
	DFLD '-----X	
	-----',	X
	POS=(21,2),	X

```

        ATTR=(PROT,ALPHA,NORM,NOMOD),          X
        EATTR=YELLOW
    DFLD  '[SYS INFO] - OSI IVP STARTING NOW',    X
        POS=(22,2),                             X
        ATTR=(PROT,ALPHA,HI,NOMOD),             X
        EATTR=RED
CURTIME DFLD  POS=(22,71),                      X
        LTH=8,                                   X
        ATTR=(PROT,ALPHA,NORM,NOMOD),          X
        EATTR=RED
    DFLD  '-----X
        -----',                              X
        POS=(23,2),                             X
        ATTR=(PROT,ALPHA,NORM,NOMOD),          X
        EATTR=YELLOW
    DFLD  'CopyRight(c) 2007, TmaxSoft, All Rights Reserved.', X
        POS=(24,31),                             X
        ATTR=(PROT,ALPHA,NORM,NOMOD),          X
        EATTR=BLUE
    FMTEND
*****
    EJECT
*****
OIVP001I MSG  TYPE=INPUT,                      X
        SOR=(OIVP001,IGNORE),                  X
        NXT=OIVP0010
    SEG
    MFLD  'OIVMPP1',                            X
        LTH=8
    MFLD  CODE,                                  X
        LTH=4
    MSGEND
*****
OIVP0010 MSG  TYPE=OUTPUT,                    X
        SOR=(OIVP001,IGNORE),                  X
        NXT=OIVP001I
    SEG
    MFLD  (CURDATE,DATE2)
    MFLD  CODE,                                  X
        LTH=4
    MFLD  ERRMSG,                              X
        LTH=79
    MFLD  (CURTIME,TIME)
    MSGEND
*****
    END

```

Register map information by using osimfsgen.

```

$ osimfsgen -m OSI.IMSA.MFSLIB OIVP001.TXT
osimfsgen 7.2.0(0) oframe@tmax:ofsrc/osi(#1) 2020-11-12 19:43:54
[/home/oframe/products/ofsrc/OpenFrame/volume_default/OSI.IMSA.MFSLIB/OIVP001.mfs] create ok.
[/home/oframe/products/ofsrc/OpenFrame/volume_default/OSI.IMSA.MFSLIB/OIVP001I.mfs] create ok.
[/home/oframe/products/ofsrc/OpenFrame/volume_default/OSI.IMSA.MFSLIB/OIVP0010.mfs] create ok.

```

2.3.4. IMSACB, IMSACBA and IMSACBB

In OSI, the PSB and DBD information is not managed and operated individually, but rather managed as a single integrated library. Such library is called **ACB**. Prepare three data sets in the same way as in MODBLKS.

Creating Datasets

Using ACBLIB requires creating datasets during the system preparation process. ACBLIB is composed in the PDS format and the individual environment configuration files are stored as ACBLIB members. To create PSBLIB in OSI, use the **pdsgen** tool.

The following example creates IMS.ACBLIB in a volume named DEFVOL, by using pdsgen.

```
$ pdsgen IMS.ACBLIB DEFVOL -f LB -l 32760
pdsgen version 7.1.0(0) oframe@tmax:ofsrc/base(#1) 2020-06-29 17:30:26
PDS Dataset Generation Program

pdsgen: *** PDS IMS.ACBLIB is created.
```

Registering ACB Information

To use ACB in OSI, the ACB information has to be registered using a tool called **acbgen**.

The following example registers the ACB information to ACBLIB for a PSB named OIVPI002 of PSBLIB.

```
$ acbgen build -p IMS.PSBLIB -d IMS.DBDLIB -l IMS.ACBLIB PSB=OIVPI002
acbgen version 7.2.0(0) oframe@tmax:ofsrc/ims(#3) 2020-08-08 16:21:16
Application Control Block Generation Program

ACBGEN COMMAND=BUILD,OPERAND=(PSB=OIVPI002),ACBLIB=IMS.ACBLIB
-----
*** ACBGEN BUILD PSB=OIVPI002
-----
*** BUILDING PSB BLOCK..... PSBNAME=OIVPI002
*** BUILDING DBD BLOCK..... DBDNAME=OIVPIDBD
*** BUILDING DBD BLOCK..... DBDNAME=OIVPIX1
-----
PROGRAM COMPLETED SUCCESSFULLY.
```



In the current version, executing **acbgen** generates ACB meta information based on the DBD and PSB meta information, and stores the ACB meta information to each metatable. ACBLIB stores no actual data.

2.3.5. MODBLKS, MODBLKSA and MODBLKSB

In OSI, the system definition that registers and manages using macros in IMS/DC is replaced by a structure called Online System Definition (OSD).

OSD is the region used for storing the various system configuration information that can be dynamically operated and is required for using OSI. OSD is composed of modules, where actual datasets are stored and OSD is managed. When the system is started, a separate area called RunTime System Definition (RTSD) is activated and allows easy dynamic management of system information and easy modification of OSD.

Prepare three datasets including a staging library and those suffixed 'A' and 'B' each. Data can be registered directly to A and B, but when applying dynamic changes, apply the information stored in the staging library by using the dynamic change tool and commands.



In the current version, OSD stores and manages data in metatables, not datasets. The three datasets including the staging library are required only when starting up the control region, and store no actual data.

Creating Tables

Using OSD requires creating metatables during the system preparation process. Metatables are created by using a tool called **osiinit**.

The following example creates a metatable to store information about the APPLCTN macro, retrieved from OSD, in a tablespace named DEFVOL, by using the **osiinit** tool.

```
$ osiinit create -t OFM_OSI_SD_APPLCTN -st DEFVOL
osiinit version 7.2.0(0) oframe@tmax:ofsrc/osi(#1) 2020-11-12 19:43:54
Initialize OpenFrame OSI System Tables

Creating OFM_OSI_SD_APPLCTN...
> "OFM_OSI_SD_APPLCTN" created...
```



For more information about metatables, refer to [Resource Tables](#).

Creating Datasets

The JCL for starting up the control region in IMS/DC can also be used as it is in OSI, by creating datasets during the system preparation process. Datasets can be created by using a tool called **idcams**.

The following example creates an OSD dataset in a volume named DEFVOL, by using the **idcams** tool.

```
$ idcams define -t CL -n OSI.IMSA.DEFLIB -o KS -k 10,0 -l 100,32760 -s 1024,128,128 -v DEFVOL
```

```
idcams version 7.1.0(0) oframe@tmax:ofsrc/base(#1) 2020-06-29 17:30:26
Access Method Services for Catalogs
```

```
IDCAMS COMMAND=DEFINE,TYPE=CL,NAME=OSI.IMSA.DEFLIB,RELATE=,CATALOG=
```

```
tbESQL Precompiler 6
```

```
TmaxData Corporation Copyright (c) 2008-. All rights reserved.
```

```
/home/oframe/products/ofsrc/OpenFrame/tsam/temp/OSI_IMSA_DEFLIB.tbc is precompiled successfully!
```

```
COMPLETED SUCCESSFULLY.
```

Registering Resource Information

The method of configuring resource information, used in operating the system in the OSD tables, is similar to the method of defining resources using macros in IMS/DC.

A tool called **osisdgen** provides this function in OSI. **osisdgen** is a tool program which operates in the Unix environment, which is not a batch job environment. **osisdgen** performs tasks with the input file containing the resource information to be configured and IMSID of the region which is used for storing the resource information. The resource information also supports the macro syntax that has been used in the existing IMS/DC.

The following example registers resource information in a region named IMSA, by using the `osi_sdlb.dat` file, which contains certain resource definition scripts.

<osi_sdlb.dat>

```
TYPE
TERMINAL NAME=OIVPTRM1, X
          FEAT=(PFK,CARD,PEN)
NAME     N031E01

APPLCTN PSB=OIVPI001,PGMTYPE=(,,),SCHDTYP=PARALLEL
TRANSACT CODE=OIVPMPP1,MSGTYPE=(SNGLSEG,RESPONSE,1),PRTY=(1,5), X
          MODE=SNGL

APPLCTN PSB=OIVPI002,PGMTYPE=(TP,,1),SCHDTYP=PARALLEL
TRANSACT CODE=OIVPMPP2,MSGTYPE=(SNGLSEG,RESPONSE,1),PRTY=(1,5), X
          MODE=SNGL,MAXRGN=1

APPLCTN PSB=OIVPI003,PGMTYPE=(TP,,1),SCHDTYP=PARALLEL
TRANSACT CODE=OIVPMPP3,MSGTYPE=(SNGLSEG,RESPONSE,1),PRTY=(1,5), X
          MODE=SNGL

APPLCTN PSB=OIVPI004,PGMTYPE=(TP,,1),SCHDTYP=PARALLEL
TRANSACT CODE=OIVPMPP4,MSGTYPE=(SNGLSEG,RESPONSE,1),PRTY=(1,5), X
          SPA=(60,STRUNC),MODE=SNGL

APPLCTN PSB=OIVPI005,PGMTYPE=(TP,,1),SCHDTYP=PARALLEL
TRANSACT CODE=OIVPMPP5,MSGTYPE=(SNGLSEG,RESPONSE,1),PRTY=(1,5), X
          SPA=(60,STRUNC),MODE=SNGL
```

```

APPLCTN PSB=OIVPIL02,PGMTYPE=(BATCH,,1),SCHDTYP=PARALLEL
TRANSACT CODE=OIVPBMP2,MSGTYPE=(SNGLSEG,RESPONSE,1),PRTY=(1,5),      X
      MODE=SNGL

APPLCTN PSB=OIVPIL03,PGMTYPE=BATCH,SCHDTYP=PARALLEL
TRANSACT CODE=OIVPBMP3,MSGTYPE=(SNGLSEG,RESPONSE,1),PRTY=(1,5),      X
      MODE=SNGL

APPLCTN PSB=OIVPIL04,PGMTYPE=BATCH,SCHDTYP=PARALLEL
TRANSACT CODE=OIVPBMP4,MSGTYPE=(SNGLSEG,RESPONSE,1),PRTY=(1,5),      X
      MODE=SNGL

APPLCTN PSB=OIVPIL05,PGMTYPE=BATCH,SCHDTYP=PARALLEL
TRANSACT CODE=OIVPBMP5,MSGTYPE=(SNGLSEG,RESPONSE,1),PRTY=(1,5),      X
      MODE=SNGL

```

Register the resource information by using osisdgen.

```

$ osisdgen osi_sd.dat IMSA
[2020-12-21T16:59:20.510891] [osisdgen(21666)          ] [M] [OSI7203M] Processing result :
Success[20], Ignore[0], Error[0]

```

Execute the following commands to deploy the resources registered in the dataset to the running OSI system.

```

$ dfsuocu0 IMSA MODBLKS
- ACTIVE MODBLKS is MODBLKSA
- Succeeded to copy MODBLKS to MODBLKSB

$ imscmd IMSA /MOD PREPARE MODBLKS
IMS control region : [IMSA]
Requested command  : [MOD PREPARE MODBLKS]
-----
ACTIVE DD: MODBLKSA IMSACBA FORMATA
MODIFY PREPARE COMMAND COMPLETED
*20356/170207*
-----
Command '/MOD PREPARE MODBLKS' execution done

$ imscmd IMSA /MOD COMMIT
IMS control region : [IMSA]
Requested command  : [MOD COMMIT]
-----
ACTIVE DD: MODBLKSB IMSACBA FORMATA
MODIFY COMMIT COMMAND COMPLETED
*20356/170211*
-----
Command '/MOD COMMIT' execution done

```

2.3.6. MODSTAT

MODSTAT is a library that stores information about Active Library for MODBLKS, IMSACB and FORMAT datasets used in OSI.

To create MODSTAT for the first time, copy the contents of the staging library for all three libraries mentioned above, to all data sets suffixed 'A' and 'B', when booting OSI. Then, specify the datasets suffixed 'A' to the active library. OSI reads and uses the active library resources whenever the system is booted.



In the current version, the active library information for MODSTAT is stored and managed as records in metatables, not datasets. These datasets are required only when starting up the control region, and store no actual data.

Creating Tables

Storing and managing MODSTAT resources requires creating metatables during the system preparation process. Metatables are created by using a tool called **osiinit**.

The following example creates a MODSTAT table in a tablespace named DEFVOL, by using the **osiinit** tool.

```
$ osiinit create -t OFM_OSI_MODSTAT -st DEFVOL
osiinit version 7.2.0(0) oframe@tmax:ofsrc/osi(#1) 2020-11-12 19:43:54
Initialize OpenFrame OSI System Tables

Creating OFM_OSI_MODSTAT...
> "OFM_OSI_MODSTAT" created...
```



For more information about metatables, refer to [Resource Tables](#).

Creating Data Sets

The JCL for starting up the control region in IMS/DC can also be used as it is in OSI, by creating datasets during the system preparation process. Datasets can be created by using a tool called **idcams**.

The following example creates a MODSTAT dataset in a volume named DEFVOL, by using the **idcams** tool.

```
$ idcams define -t CL -n OSI.IMSA.MODSTAT -o KS -k 8,0 -l 160,160 -s 1024,128,128 -v DEFVOL
idcams version 7.1.0(0) oframe@tmax:ofsrc/base(#1) 2020-06-29 17:30:26
Access Method Services for Catalogs

IDCAMS COMMAND=DEFINE,TYPE=CL,NAME=OSI.IMSA.MODSTAT,RELATE=,CATALOG=
```

```
tbESQL Precompiler 6
```

```
TmaxData Corporation Copyright (c) 2008-. All rights reserved.
```

```
/home/oframe/products/ofsrc/OpenFrame/tsam/temp/OSI_IMSA_MODSTAT.tbc is precompiled successfully!
```

```
COMPLETED SUCCESSFULLY.
```

2.3.7. PSBLIB

Program Communication Block (PSB) is a collection of PCBs which are control blocks for using the databases or messages used in applications. There is usually one PSB per user, for each program.

PCB is a control block provided by the system to communicate with views, message sources, or message destinations for OpenFrame HiDB in applications. IO PCB/ALT PCB is a resource used for reading and writing data in message queues provided in OSI within PCB and therefore it can only be used when IO PCB/ALT PCB is started. IO PCB can be used only by using IOPCB-MASK, which is provided as the program's first parameter, for programs that run in OSI regardless of PSB's technology. However, multiple ALT PCBs can be used as needed.

Creating Datasets

Using PSBLIB requires creating datasets during the system preparation process.

PSBLIB is composed in the PDS format and the individual environment configuration files are stored as PSBLIB members. To create PSBLIB in OSI, use a tool called **pdsgen**.

The following example creates IMS.PSBLIB in a volume named DEFVOL, by using pdsgen.

```
$ pdsgen IMS.PSBLIB DEFVOL -f LB -l 32760
pdsgen version 7.1.0(0) oframe@tmax:ofsrc/base(#1) 2020-06-29 17:30:26
PDS Dataset Generation Program

pdsgen: *** PDS IMS.PSBLIB is created.
```

Registering PSB Information

To use PSB in OSI, PSB information has to be registered using a tool called **psbgen**.

The following example registers the PSB information by using a file named OIVPI002, which contains the scripts defining PSB.

```
<OIVPI002>
```

```
PCB TYPE=TP,MODIFY=YES
PCB TYPE=DB,DBDNAME=OIVPIDBD,KEYLEN=10,PROCOPT=A
```

```
SENSEG NAME=DBSEG
PSBGEN LANG=COBOL,PSBNAME=OIVPI002
END
```

Register the PSB information by using psbgen.

```
$ psbgen OIVPI002
psbgen version 7.2.0(0) oframe@tmax:ofsrc/ims(#3) 2020-08-08 16:21:16
Program Specification Block Generation Program

psbgen: force flag on; the tool overwrites existing data
psbgen: 1 files are requested in total
-----
psbgen Processing PSB script "OIVPI002"
-----

psbgen: Processing PSB "OIVPI002"
psbgen: removing existing PSB OIVPI002 metadata
psbgen: successfully processed PSB "OIVPI002"

psbgen: successfully processed for the requested PSBs (total 1)
```



In the current version, executing psbgen stores the meta information described in the PSB scripts to metatables. PSBLIB stores no actual data.

2.3.8. RESLIB

RESLIB is a library that stores MDA information, which is required for using datasets dynamically. For more information, refer to [DFSRESLB](#).

2.3.9. STEPLIB

The application program running in the OSI server is developed by using the COBOL programming language.

Application developers have to place the binaries created after compiling their programs, to STEPLIB. Perform preprocessing as needed before compiling. The programs can be run by starting the OSI system after registering program resource definitions in the OSI tables.

Creating Datasets

In the same way as when creating FORMAT, use pdsgen to create datasets.

The following example creates OSI.IMSA.STEPLIB in a volume named DEFVOL, by using pdsgen.

```
$ pdsgen OSI.IMSA.STEPLIB DEFVOL -f LB -l 32760
```

```
pdsgen version 7.1.0(0) oframe@tmax:ofsrc/base(#1) 2020-06-29 17:30:26
PDS Dataset Generation Program
```

```
pdsgen: *** PDS OSI.IMSA.STEPLIB is created.
```

2.4. Configuring Storage

OSI requires system tables to be used as storage, for the system operation.

2.4.1. MQ

OSI basically uses the functions of Tmax, a TP-monitor, to send messages between terminals and applications, while storing all messages in Message Queue (MQ) tables. To operate the OSI system, prepare Master, MPP (terminal), BMP tables and MQ datasets.



In the current version, all messages transferred during operation are stored in MQ tables. MQ datasets are required only when starting up the control region, and store no actual data.

Creating Tables

MQ system tables are created by using the **osiinit** tool.

The following example creates three MQ tables by using the **osiinit** tool.

```
$ osiinit create -t OFM_OSI_MQ -st DEFVOL
osiinit version 7.2.0(0) oframe@tmax:ofsrc/osi(#1) 2020-11-12 19:43:54
Initialize OpenFrame OSI System Tables

Creating OFM_OSI_MQ...
> "OFM_OSI_MQ" created...
```



For more information about MQ system tables, refer to [Resource Tables](#).

Creating Datasets

The JCL for starting up the control region in IMS/DC can also be used as it is in OSI, by creating datasets during the system preparation process. Datasets can be created by using a tool called **idcams**.

The following example creates an MQ dataset in a volume named DEFVOL, by using the **idcams** tool.

```
$ idcams define -t CL -n OSI.IMSA.MQLIB -o KS -k 32,0 -l 10000,32000 -s 1024,128,128 -v DEFVOL
idcams version 7.1.0(0) oframe@tmax:ofsrc/base(#1) 2020-06-29 17:30:26
Access Method Services for Catalogs

IDCAMS COMMAND=DEFINE,TYPE=CL,NAME=OSI.IMSA.MQLIB,RELATE=,CATALOG=

tbESQL Precompiler 6

TmaxData Corporation Copyright (c) 2008-. All rights reserved.

/home/oframe/products/ofsrc/OpenFrame/tsam/temp/OSI_IMSA_MQLIB.tbc is precompiled successfully!

COMPLETED SUCCESSFULLY.
```



For more information about dataset usages, refer to *OpenFrame Base Dataset Guide*.

2.4.2. REGION

Region ID of the user server (dependent region), job name, PSB name, program name and other necessary information for operating the OSI system are stored in REGION tables.

Creating Tables

To create REGION tables, use the **osiinit** tool.

The following example creates a REGION table by using the **osiinit** tool.

```
$ osiinit create -t OFM_OSI_REGION -st DEFVOL
osiinit version 7.2.0(0) oframe@tmax:ofsrc/osi(#1) 2020-11-12 19:43:54
Initialize OpenFrame OSI System Tables

Creating OFM_OSI_REGION...
> "OFM_OSI_REGION" created...

$ osiinit create -t OFM_OSI_REGION_SVR -st DEFVOL
osiinit version 7.2.0(0) oframe@tmax:ofsrc/osi(#1) 2020-11-12 19:43:54
Initialize OpenFrame OSI System Tables

Creating OFM_OSI_REGION_SVR...
> "OFM_OSI_REGION_SVR" created...
```



For more information about REGION tables, refer to [Resource Tables](#).

2.4.3. RTSD

System definition resources are copied and managed in a separate area called RunTime System Definition (RTSD), at the point where each control region is started in OSI. During the system operation, all region servers present in the same IMS system can share the same RTSD resources.

Creating Tables

To create RTSD tables, use the **osiinit** tool.

The following example creates the APPLCTN and TRANSACT tables in a tablespace named DEFVOL, by using the **osiinit** tool.

```
$ osiinit create -t OFM_OSI_RTSD_APPLCTN -st DEFVOL
osiinit version 7.2.0(0) oframe@tmax:ofsrc/osi(#1) 2020-11-12 19:43:54
Initialize OpenFrame OSI System Tables

Creating OFM_OSI_RTSD_APPLCTN...
> "OFM_OSI_RTSD_APPLCTN" created...

$ osiinit create -t OFM_OSI_RTSD_TRANSACT -st DEFVOL
osiinit version 7.2.0(0) oframe@tmax:ofsrc/osi(#1) 2020-11-12 19:43:54
Initialize OpenFrame OSI System Tables

Creating OFM_OSI_RTSD_TRANSACT...
> "OFM_OSI_RTSD_TRANSACT" created...
```



For more information about RTSD tables, refer to [Resource Tables](#).

3. System Server Configuration

This chapter describes how to configure the OSI system server (control region).

3.1. Overview

All OSI servers are managed by Tmax, which is a TP monitor.

The following describes system servers that must be configured to be operated in OSI. Other servers can be used without separate configurations.

Server	Description
Start/Stop Server (osiomsvr)	Starts and stops the OSI system server and MPP user server.
Schedule Server (osisschd)	Schedules all messages requested by one IMSID to be sent to MPP servers.
Command Server (osicmdsv)	Processes OSI commands with a single IMSID.
OTMA Server (osiotmasvr)	Integrates with IBM MQ and provides OTMA functionality.
Administrator Server (osiofmgr)	Handles OpenFrame Manager functions.

Managing the OSI system server requires registering the system server information in Tmax. To register the system server in Tmax, a Tmax environment configuration file needs to be written. Detailed information about the environment configuration is described in each section.

3.2. Start/Stop Server (osiomsvr)

Starts and stops the OSI system server and MPP user server. Commonly applied to all IMS systems, without the need of individual registration for each IMS system.

3.2.1. Tmax Environment Configuration

The following example configures common servers and services in the Tmax configuration file, to use the osiomsvr server in OSI.

- Common Configuration

Configure the [SERVER] and [SERVICE] sections in the Tmax configuration file as follows:

```
*SERVER
```

```

osiomsvr          SVGNAME = svg_domain, MIN = 1, MAX = 1, SVRTYPE = UCS

*SERVICE
OSIOMSVRBOOT     SVRNAME = osiomsvr
OSIOMSVRDOWN     SVRNAME = osiomsvr
OSIOMSVRREL      SVRNAME = osiomsvr

```

3.3. Schedule Server (osisschd)

Schedules OSI messages and provides different functions required for message scheduling. One schedule server is registered for one IMS system.

3.3.1. Tmax Environment Configuration

The following example configures common servers and services as well as individual servers required for each IMS system, in the Tmax configuration file, to use the schedule server in OSI.

- Common Configuration

Configure the [SERVER] and [SERVICE] sections in the Tmax configuration file, as follows. For the service section, you just need to configure all at once, without the need of individual registration for each IMS system.

```

*SERVER
osisschd          SVGNAME = svg_domain, MIN = 0, MAX = 10, SVRTYPE = UCS

*SERVICE
OSISSCHDCTL      SVRNAME = osisschd
OSISSCHDSVC      SVRNAME = osisschd
OSISSCHDREG      SVRNAME = osisschd
OSISSCHDDTP      SVRNAME = osisschd
OSISSCHDOTMA     SVRNAME = osisschd

```

- Individual Configuration for Each IMS System

If the IMSID is IMSA, a server named IMSASCHD must be registered. The bold part in the following example must be modified if the IMSID is different. Specify osisschd, which is the actual binary name of the IMSASCHD server, by setting the TARGET option.

```

*SERVER
IMSASCHD        SVGNAME = svg_domain, MIN = 1, MAX = 1, SVRTYPE = UCS,
                    TARGET = osisschd

```

3.4. Command Server (osicmdsv)

Processes different functions required for operating the OSI system with different commands. Must be individually registered for each IMS system.

3.4.1. Tmax Environment Configuration

The following example registers the command server in the Tmax configuration file, to use it in OSI.

In the same way as the schedule server, there are common servers and services as well as individual servers required for each IMS system.

- Common Configuration

Configure the [SERVER] and [SERVICE] sections in the Tmax configuration file as follows. For the service section, you just need to configure all at once, without the need of individual registration for each IMS system.

```
*SERVER
osicmdsv      SVGNAME = svg_domain, MIN = 0, MAX = 10, SVRTYPE = UCS

*SERVICE
OSICMDSVSV   SVRNAME = osicmdsv
```

- Individual Configuration for Each IMS System

If the IMSID is IMSA, a server named IMSACMMD must be registered. The bold part in the following example must be modified if the IMSID is different. Specify osicmdsv, which is the actual binary name of the IMSACMMD server, by setting the TARGET option.

```
*SERVER
IMSACMMD    SVGNAME = svg_domain, MIN = 1, MAX = 1, SVRTYPE = UCS,
              TARGET = osicmdsv
```

3.5. OTMA Server (osiotmasvr)

Supports integration with IBM MQ and provides OTMA functionality. Must be individually registered for each IMS system.

3.5.1. Tmax Environment Configuration

The following example registers the OTMA server in the Tmax configuration file, to use it in OSI.

In the same way as the schedule server, there are common servers and services as well as individual servers required for each IMS system.

- Common Configuration

Configure the [SERVER] and [SERVICE] sections in the Tmax configuration file, as follows. For the service section, you just need to configure all at once, without the need of individual registration for each IMS system.

```
*SERVER
osiotmasvr      SVGNAME = svg_node1, MIN = 0, MAX = 10, SVRTYPE = UCS
```

- Individual Configuration for Each IMS System

If the IMSID is IMSA, a server named IMSAOTMA must be registered. The bold part in the following example must be modified if the IMSID is different. Specify osiotmasvr, which is the actual binary name of the IMSAOTMA server, by setting the TARGET option.

```
*SERVER
IMSAOTMA      SVGNAME = svg_node1, MIN = 1, MAX = 1, SVRTYPE = UCS,
                TARGET = osiotmasvr
```

3.6. Administrator Server (osiofmgr)

Handles OpenFrame Manager functions. Commonly applied to all IMS systems, without the need of individual registration for each IMS system.

3.6.1. Tmax Environment Configuration

The following example registers the administrator server in the Tmax configuration file, to use it in OSI.

- Common Configuration

Configure the [SERVER] and [SERVICE] sections in the Tmax configuration file, as follows.

```
*SERVER
osiofmgr      SVGNAME = svg_node, MIN = 1, MAX = 10

*SERVICE
OSIOFMGRSVC   SVRNAME = osiomsvr
```

4. User Server Configuration

This chapter describes how to configure the OSI user server (dependent region).

4.1. Overview

In order to run user generated application programs, in the OSI system, a server which corresponds to the user server (dependent region) in IBM Mainframe IMS/DC must be prepared.

The following describes user servers that are used for OSI operation.

User Server	Description
MPP user server	As a part corresponding to the MPP region in IMS/DC, OSI operates the MPP server in units of message class. It is a server corresponding to a maximum of 1:4 per MPP Region operated in the existing IMS/DC.
BMP user server	Server which allows operation of user programs that are run in the BMP region of IMS/DC.

OSI user servers are managed by a TP-monitor (Tmax) and in order to operate them information about user servers must be registered in Tmax. The environment configuration files of Tmax must be created in order to register servers in Tmax.

When preparing user servers, the process of preparing MPP user servers and BMP user servers is different. The BMP region which runs batch jobs uses server modules which are provided in the system and therefore does not separately create servers. It needs to be configured while taking into account the maximum number of BMP jobs that will be performed simultaneously. Detailed explanation of the environment configuration will be provided in each section.

4.2. Server Group Configuration

The following is an example of server group configuration.

4.2.1. General Configuration

The following is an example of general environment settings for OSI.

```
*SVRGROUP
svg_node1
  NODENAME = "NODE1"
```

4.2.2. XA Configuration

The following is an example of an XA configuration where HiDB and standard Tibero are set as Multiple RMs.

```
*SVRGROUP
svg_node1
  NODENAME = "NODE1"
  SVGTYPE = MTMAX,
  SVGLIST = "svg_hidb_xa,svg_tibero_xa"

svg_hidb_xa
  NODENAME = "NODE1"
  SVGTYPE = STMAX,
  TMSNAME = tms_tbr,
  DBNAME = TIBERO,
  OPENINFO = "TIBERO_XA:user=tibero, pwd=tmax, sestm=60,db=tb_fix3,conn_id=db1"

svg_tibero_xa
  NODENAME = "NODE1"
  SVGTYPE = STMAX,
  TMSNAME = tms_tbr,
  DBNAME = TIBERO,
  OPENINFO = "TIBERO_XA:user=tibero, pwd=tmax, sestm=60,db=tb_fix3"
```

Specify the libraries in `$(TAXMDIR)/config/RM` that a server with multiple RMs configured will dynamically load.

```
#TIBERO
TIBERO:tbxa,tbs

# IBM MQ for 64bit
MQ:mqmx64,mqs

# IBM MQ for 32bit
MQ:mqmx32,mqs
```



1. For XA configuration for HiDB, relevant settings must be added to the Tmax configuration file. The value of the **DLI_CONN_ID** key in the **GENERAL** section of the **ssm.{IMSID}{SSM}** subject in OpenFrame configuration must match the `conn_id` specified in the `OPENINFO` of the `SVRGROUP` setting in the Tmax configuration file. For more information, refer to "OpenFrame OSI Configuration" in *OpenFrame OSI Configuration Guide*.
2. For XA configuration, a user server establishes sessions for all databases within its server group.

4.3. MPP User Server

As a part of the MPP Region in IMS/DC, it operates with up to 4 MPP user servers per MPP Region in the existing IMS/DC. On the prepared MPP user server, the MPP application written by the user is scheduled and operated in the same way as it did in IMS/DC.

Just like system servers, MPP user servers are also managed by Tmax, a TP-monitor. To be managed by Tmax, Tmax environment configuration files must be created.

The following is an example of registering the server and service that must be commonly registered in the Tmax configuration file in order to use the MPP user server in OSI and configuring the server that can process transaction classes 1,2,3,4 in the IMS system called IMSA.

- Common configuration

Set the [SERVER] and [SERVICE] clauses in the Tmax configuration file as follows:

(The service does not need to be registered for each IMS system and needs to be set only once.)

```
*SERVER
OSIMPPSVR      SVGNAME = svg_node1, MIN = 0, MAX = 10

*SERVICE
OSIMPPSVRSVC   SVRNAME = OSIMPPSVR, SVCTIME=60
OSIMPPSVRMGR   SVRNAME = OSIMPPSVR, SVCTIME=60
```

- Configuration by IMS system

It is divided into IMSID and message class, and registered by adding prefix and suffix to the server name, respectively. The part marked in bold in the example is the part that needs to be changed if the IMSID and class are different. OSIMPPSVR, which is the actual binary name of the IMSAMPP_TCL 1~4 server, must be specified using the TARGET option.

```
*SERVER
IMSAMPP_TCL1  SVGNAME = svg_node1, MIN = 1, MAX = 10,
                TARGET = OSIMPPSVR
IMSAMPP_TCL2  SVGNAME = svg_node1, MIN = 1, MAX = 10,
                TARGET = OSIMPPSVR
IMSAMPP_TCL3  SVGNAME = svg_node1, MIN = 1, MAX = 10,
                TARGET = OSIMPPSVR
IMSAMPP_TCL4  SVGNAME = svg_node1, MIN = 1, MAX = 10,
                TARGET = OSIMPPSVR
```



When registering an MPP user server in the Tmax environment configuration file, care should be taken when configuring the MAX value. MAX value refers to the number of transactions which can be simultaneously processed in the MPP user server. When the maximum number of MPP to be executed is set, the MPP user server operates in the same way as the MPP Region of IMS/DC.

4.4. BMP User Server

BMP user server is a server which allows the operation of a user program that is run in the MBP Region of IMS/DC. The MPP user server has separate server modules for each user server (dependent region), but the BMP user server uses the BMP user server module for all user servers (dependent region).

The BMP user server uses the OSI server module. Therefore, unlike MPP user server configuration, once the process for registering the entire OSI system has been completed, a separate creation process is not required.

Just like system servers, BMP user servers are managed by a TP-monitor, Tmax. To be managed by Tmax, Tmax environment configuration files must be created.

The following is an example of registering a BMP user server in the Tmax environment configuration file.

- Common configuration

Set the [SERVER] and [SERVICE] clauses in the Tmax configuration file as follows:

```
*SERVER
osibmpsv      SVGNAME = svg_node1, MIN = 0, MAX = 10, SVRTYPE = UCS,
               RESTART = NO

*SERVICE
OSIBMPSVSVC   SVRNAME = osibmpsv
OSIBMPSVSHUTDOWN SVRNAME = osibmpsv
```

- XA configuration

It is identified by `ssm.{IMSID}{SSM}` configuration name and registered by adding suffix to the server name. The bold part in the following example must be modified if the SSM configuration is different. The actual binary name of the `osibmpsv.DB2T` server, `osibmpsv`, must be specified using the `TARGET` option.

```
*SERVER
osibmpsv.DB2T SVGNAME = svg_node1, MIN = 0, MAX = 10, SVRTYPE = UCS,
               TARGET = osibmpsv, RESTART = NO
```



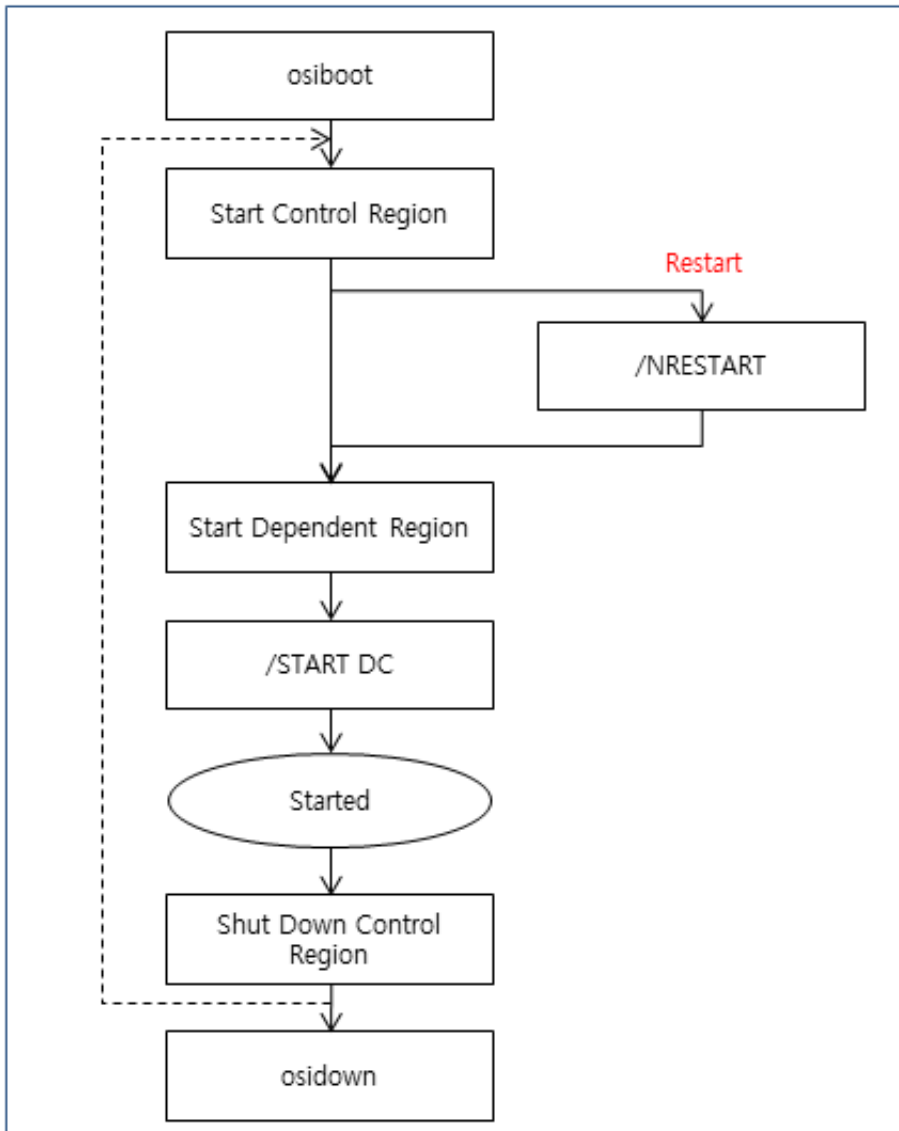
When registering a BMP user server in the Tmax environment configuration file, care should be taken when configuring the MAX value. The MAX value refers to the BMP JOB count which can be simultaneously enabled in OSI.

5. System Operation

This chapter describes how to start and shut down the OSI system and how to use the log files which record operating information about the OSI system.

5.1. Starting and Shutting Down OSI

In order to operate the OSI system, Tmax, which is a TP-monitor, must be running and the modules of OpenFrame Base and OpenFrame Batch must be prepared, to be able to run.



Starting and Shutting Down OSI



TCache must be created before starting up the control region. TCache can be created using the **pfmtcacheadmin** tool.

5.1.1. Starting OSI

The following are the steps for starting OSI.

1. Start osiomsvr and base/batch modules required in Tmax engine and OSI system. (osiboot)

Starting Tmax consists of three internal stages. The stages are performed simultaneously using the **osiboot** tool.

- a. Tmax is started.
- b. The server modules used by other packages of OpenFrame that are needed to start OSI, are also started.

In order to perform the second stage the server modules used in packages other than OSI, must be configured beforehand in a environment configuration file named "osi.ofsys.seq". Consideration should be given to the operating order of each server.

To enable the operation of BMP in OSI, the server modules required in OpenFrame Batch must be started when Tmax is started.

The following example is a general example of this.

```
$ cat osi.ofsys.seq
TPFMAGENT
ofrsasvr
ofrlhsvr
ofrdmsvr
ofrdsedt
ofrcmsvr
ofruisvr
ofrsmlog
obmjmsvr
obmjschd
obmjinit
obmjhist
obmjspb
ofrpmsvr
obmtsmgr
obmjtimr
tmsvr
#IMSAOTMA
```

- c. osiomsvr to start the OSI system server and user server is started.

Start Tmax through osiboot as follows:

```
$ osiboot

TMBOOT for node(NODE1) is starting:
TMBOOT: TMM is starting: Wed Mar 10 19:37:02 2021
TMBOOT: CLL is starting: Wed Mar 10 19:37:02 2021
TMBOOT: CLH is starting: Wed Mar 10 19:37:02 2021
```

```

TMBOOT: TLM(tlm) is starting: Wed Mar 10 19:37:02 2021
[2021-03-10T19:37:02.848282] [osiboot(5670) ] [M] [OSI7061M] System
server(ofrsasvr) booting ok
[2021-03-10T19:37:02.851625] [osiboot(5670) ] [M] [OSI7061M] System
server(ofrlhsvr) booting ok
[2021-03-10T19:37:02.855374] [osiboot(5670) ] [M] [OSI7061M] System
server(ofrdmsvr) booting ok
[2021-03-10T19:37:02.859919] [osiboot(5670) ] [M] [OSI7061M] System
server(ofrdsedt) booting ok
[2021-03-10T19:37:02.864295] [osiboot(5670) ] [M] [OSI7061M] System
server(ofrcmsvr) booting ok
[2021-03-10T19:37:02.868285] [osiboot(5670) ] [M] [OSI7061M] System
server(ofruisvr) booting ok
[2021-03-10T19:37:02.873191] [osiboot(5670) ] [M] [OSI7061M] System
server(ofrsmlog) booting ok
[2021-03-10T19:37:02.877533] [osiboot(5670) ] [M] [OSI7061M] System
server(obmjmsvr) booting ok
[2021-03-10T19:37:02.881853] [osiboot(5670) ] [M] [OSI7061M] System
server(obmj sched) booting ok
[2021-03-10T19:37:02.886434] [osiboot(5670) ] [M] [OSI7061M] System
server(obmjinit) booting ok
[2021-03-10T19:37:02.890673] [osiboot(5670) ] [M] [OSI7061M] System
server(obmjhist) booting ok
[2021-03-10T19:37:02.897808] [osiboot(5670) ] [M] [OSI7061M] System
server(obmjspb) booting ok
[2021-03-10T19:37:02.905868] [osiboot(5670) ] [M] [OSI7061M] System
server(ofrpmsvr) booting ok
[2021-03-10T19:37:02.920575] [osiboot(5670) ] [M] [OSI7061M] System
server(obmtmgr) booting ok
[2021-03-10T19:37:02.937001] [osiboot(5670) ] [M] [OSI7061M] System server(tmsvr)
booting ok
[2021-03-10T19:37:02.943765] [osiboot(5670) ] [M] [OSI7061M] System
server(osiomsvr) booting ok
[2021-03-10T19:37:02.943791] [osiboot(5670) ] [M] [OSI7051M] Booting process
complete.

```

2. Start the system server (control region) by IMSID using JCL.

Start the OSI system server (control region) after starting Tmax.

One OSI system is started by each IMSID. When using a 3 OSI system configuration, start three OSI systems per IMSID. The OSI system is started by running JOB in JCL, using the tjesmgr tool.

The following is an example of starting the system server (control region) whose IMSID is IMSA, with JCL.

```

//IMSACTL JOB
//STEP1 EXEC PGM=DFSMVRC0,
// PARM='CTL,IMS,IMSID=IMSA'
//MODBLKSA DD DISP=SHR,DSN=OSI.IMSA.DEFLIBA
//MODBLKSB DD DISP=SHR,DSN=OSI.IMSA.DEFLIBB
//IMSACBA DD DISP=SHR,DSN=IMS.ACBLIBA
//IMSACBB DD DISP=SHR,DSN=IMS.ACBLIBB
//DFSRESLB DD DISP=SHR,DSN=IMS.RESLIB
//FORMATA DD DISP=SHR,DSN=OSI.IMSA.MFSLIBA
//FORMATB DD DISP=SHR,DSN=OSI.IMSA.MFSLIBB

```

```

//STEPLIB DD DISP=SHR,DSN=OSI.IMSA.STEPLIB
//QBLKS DD DISP=SHR,DSN=OSI.IMSA.MQLIB
//MODSTAT DD DISP=SHR,DSN=OSI.IMSA.MODSTAT
//SYSOUT DD SYSOUT=*
//

$ tjesmgr boot
>
Command : [boot]
Node name : A L L
NODE1 is booted.

$ tjesmgr run IMSACTL
>
Command : [run IMSACTL]
Node name : A N Y
(JOB00001) /root/OF71/OHOME/volume_DEFVOL/SYS1.JCLLIB/IMSACTL is submitted as IMSACTL(JOB00001).

```

In the PARM parameter of JCL which starts the IMS system server, the following two items can be configured.

Item	Description
IMSID	Unique IMS ID of the system server. (4 bytes)
AUTO	Option to automatically execute "/NRESTART" or "/ERESTART" command after starting the server. (YES NO)

3. Enable OSI system by OSI commands

If it is an initial startup of the OSI system after starting the system server (control region) with JCL, execute the "/NRE CHKPT 0" command. If it is not the initial OSI startup, execute the "/NRE" or "/ERE" command. The "/NRE" command is used when the server was normally shut down through the "/CHE" command. And when it was not shut down normally, "/ERE" command is used. Otherwise, an error occurs and the command is not executed. After that, execute "/START DC" command to log on to the terminal.

In the following example, the OSI command, /NRE and /STA DC of the system server whose IMSID is IMSA are input in order.

```

$ imscmd IMSA /NRE
IMS control region : [IMSA]
Requested command : [NRE]
-----
NRESTART COMMAND IN PROGRESS
*21069/194013*
-----
Command '/NRE' execution done
$ imscmd IMSA /STA DC
IMS control region : [IMSA]
Requested command : [STA DC]
-----
START COMMAND COMPLETED
*21069/194018*

```

```
-----  
Command '/STA DC' execution done
```

4. Start the user server per IMSID (dependent region) using JCL.

After the system server (control region) is started, execute the MPP start JCL command, according to the class to be processed. MPP can be started by running jobs as needed. It can also be started by using "/START REGION" which is an OSI command.

The following is an example of JCL that starts the user server (dependent region) whose IMSID 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=DFSRR00,  
// PARM='MSG,001002003004,W00099000,,,R,,,,IMSA,,,,'  
//STEPLIB DD DISP=SHR,DSN=OSI.IMSA.STEPLIB  
//SYSPRINT DD SYSOUT=*  
//SYSOUT DD SYSOUT=*  
//SYSDBOUT DD SYSOUT=*  
//
```

The following is an example of the "/START REGION" command, which starts the user server (dependent region) whose JCL name is IMSAMSG.

```
$ imscmd IMSA /START REGION IMSAMSG  
IMS control region : [IMSA]  
Requested command : [START REGION IMSAMSG]  
-----  
START COMMAND COMPLETED  
*21069/194050*  
-----  
Command '/START REGION IMSAMSG' execution done
```



For detailed information about "/START REGION", refer to *OpenFrame OSI Command Reference Guide*.

5.1.2. Shutting Down OSI

The following are the steps for shutting down OSI.

1. Shut down the system server (control region) and the user server (dependent region).

Use the OSI command to terminate the system server (control region) and the user server (dependent region). The OSI command is CHECKPOINT, and shut down the system server after terminating the user server.

The following is an example of shutting down the system server and the user server whose IMSID is IMSA, using the OSI command.

```
$ imscmd IMSA /CHE FREEZE
IMS control region : [IMSA]
Requested command  : [CHE FREEZE]
-----
CHECKPOINT COMMAND IN PROGRESS
*21069/194108*
-----
Command '/CHE FREEZE' execution done
```

2. Shut down Tmax.

If the osidown tool is used, after the server modules which are used in the packages of OpenFrame are shut down, Tmax is shutdown.

The following is an example of shutting down Tmax using osidown.

```
$ osidown
[2021-03-10T19:41:27.956075] [osidown(5969) ] [M] [OSI7141M] System server(osiomsvr)
shutdown ok
[2021-03-10T19:41:27.960864] [osidown(5969) ] [M] [OSI7141M] System server(tmsvr)
shutdown ok
[2021-03-10T19:41:27.983167] [osidown(5969) ] [M] [OSI7141M] System server(obtmsmgr)
shutdown ok
[2021-03-10T19:41:27.995447] [osidown(5969) ] [M] [OSI7141M] System server(ofrpsvr)
shutdown ok
[2021-03-10T19:41:27.000641] [osidown(5969) ] [M] [OSI7141M] System server(obmjspb)
shutdown ok
[2021-03-10T19:41:28.003942] [osidown(5969) ] [M] [OSI7141M] System server(obmjhist)
shutdown ok
[2021-03-10T19:41:28.033647] [osidown(5969) ] [M] [OSI7141M] System server(obmjinit)
shutdown ok
[2021-03-10T19:41:28.040265] [osidown(5969) ] [M] [OSI7141M] System server(obmjschd)
shutdown ok
[2021-03-10T19:41:28.047434] [osidown(5969) ] [M] [OSI7141M] System server(obmjmsvr)
shutdown ok
[2021-03-10T19:41:28.076810] [osidown(5969) ] [M] [OSI7141M] System server(ofrsmlog)
shutdown ok
[2021-03-10T19:41:28.086320] [osidown(5969) ] [M] [OSI7141M] System server(ofruisvr)
shutdown ok
[2021-03-10T19:41:28.093782] [osidown(5969) ] [M] [OSI7141M] System server(ofrcmsvr)
shutdown ok
[2021-03-10T19:41:28.101464] [osidown(5969) ] [M] [OSI7141M] System server(ofrdsedt)
shutdown ok
[2021-03-10T19:41:28.108771] [osidown(5969) ] [M] [OSI7141M] System server(ofrdmsvr)
shutdown ok
[2021-03-10T19:41:28.116562] [osidown(5969) ] [M] [OSI7141M] System server(ofrlhsrv)
shutdown ok
[2021-03-10T19:41:28.124770] [osidown(5969) ] [M] [OSI7141M] System server(ofrsasvr)
shutdown ok

TMDOWN for node(NODE1) is starting:
TMDOWN: CLH downed: Wed Mar 10 19:41:28 2021
```

```
TMDOWN: CLL downed: Wed Mar 10 19:41:28 2021
TMDOWN: TLM downed: Wed Mar 10 19:41:28 2021
TMDOWN: TMM downed: Wed Mar 10 19:41:28 2021
TMDOWN: TMAX is down
```

5.1.3. Restarting OSI

The following are the steps for restarting OSI.

1. Start Tmax. (osiboot)
2. Start a system server (control region) for each IMSID, using JCL.
3. Restart the system server (control region) for each IMSID using the command. After shutting down, using the OSI command, the system server needs to be restarted in order to get the previous system usage information.

The following is an example of restarting the system server (control region) whose IMSID is IMSA using the OSI command.

```
$ imscmd IMSA /NRE
IMS control region : [IMSA]
Requested command  : [NRE]
-----
NRESTART COMMAND IN PROGRESS
*21069/195025*
-----
Command '/NRE' execution done
```

4. To allow a terminal to be logged on use "/START DC".
5. Start a user server (dependent region) per IMSID using JCL or by using "/START REGION".



For more information on "/START REGION", refer to *OpenFrame OSI Command Reference Guide*.

5.2. Log Management

In OpenFrame OSI system, the server logs are managed by dividing it into a system server, MPP user server, and BMP user server. This section describes the management method for each server.

5.2.1. System Server and MPP User Server

The logs generated by OSI system server and MPP user servers are OUT log and ERR log. Both logs are stored in ULOGDIR described in the NODE clause of the Tmax configuration file. Users can freely set ULOGDIR, but it is recommended to set it to the directory under `${OPENFRAME_HOME}/log`.

• **OUT log**

If a file name is specified with the [-o] option in the CLOPT clause, all contents written to stdout by the OSI system server and the MPP user server are printed to the file. The output log follows the common output format of OpenFrame.

The following is the output format of the OUT log.

```
[YYYY-MM-DDTHH:MI:SS.FFFFFFF] [MODULE(PID)] [LEVEL] [MSGCODE] MESSAGE-CONTENTS
```

Field	Description
[YYYY-MM-DDTHH:MI:SS.FFFFFFF]	YYYY-MM-DDTHH:MI:SS.FFFFFFF type timestamp which includes dates.
MODULE	Module name of the OSI system which prints messages.
PID	Process ID of the application server.
LEVEL	Prints one of the following. <ul style="list-style-type: none"> • M (Message): A message in which the system gives information to users. Various information about the server is printed. • E (Error): A message which is printed when an error occurs in the system. It is displayed when users give incorrect information to the system or there is an internal error in the system. • W (Warning): A message which is printed when an error occurs in the system, but does not affect the operation of the system. • D (Debug): A message which is output for system debugging.
MSGCODE	8-character OSI message code.
MESSAGE-CONTENTS	Field where log messages are printed.

The following is an example of the OUT log file of the MPP user server whose message class is 1 in the IMS system named IMSA.

```
[2021-03-10T19:59:51.194767] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP2D] [GENERAL]LOG_LEVEL = D
[2021-03-10T19:59:51.217509] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP1D] AGN =
[2021-03-10T19:59:51.217539] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP2D] [SECURITY]TYPE = TACF
[2021-03-10T19:59:51.217564] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP2D] security type is TACF[2]
[2021-03-10T19:59:51.217591] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP3D]
[GENERAL]SCHEDULE_RECOVER_MAXCNT = 5
[2021-03-10T19:59:51.227545] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP3D] [MQ]USE_MQ = NO
[2021-03-10T19:59:51.227821] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP1D] CTL's JOBID : JOB02316
[2021-03-10T19:59:51.227844] [IMSAMPP_TCL1(6839) ] [M] [OSI0291M] IMSA server boots -
resource manager initialization starts
[2021-03-10T19:59:51.229970] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP2D] [CPM]REGION_CCSID =
[2021-03-10T19:59:51.229992] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP2D] [CPM]ASCII_TO_EBCDIC =
ASCEBCUS.cpm
[2021-03-10T19:59:51.230114] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP2D] [CPM]EBCDIC_TO_ASCII =
```

```

EBCASCUS.cpm
[2021-03-10T19:59:51.230214] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP0D] cpm version
[2021-03-10T19:59:51.230228] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP2D] [SCREEN]3270_TYPE = 3270-
A2
[2021-03-10T19:59:51.230239] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP0D] SCREEN TYPE : A2
[2021-03-10T19:59:51.230255] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP3D] [CPM]CONVERT_TO_SPACE =
X'00'
[2021-03-10T19:59:51.230270] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP2D] X'00'(in ASCII) will be
converted X'20'
[2021-03-10T19:59:51.230280] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP1D] null character=1a
[2021-03-10T19:59:51.230289] [IMSAMPP_TCL1(6839) ] [M] [OSI0101M] osimfs version: 7.2.0(4)
sbaek@tmax:of7_1_mvs_dev/osi(#1) 2020-11-12 19:43:54
[2021-03-10T19:59:51.231791] [IMSAMPP_TCL1(6839) ] [D] [DEBUGP2D] [GENERAL]SVCLLOG = N
[2021-03-10T19:59:51.231812] [IMSAMPP_TCL1(6839) ] [M] [OSI0291M] IMSA server boots -
resource manager initialization completed

```

- **ERR log**

If a file name is specified with the [-e] option in the CLOPT clause, all contents written to stderr in the OSI system server and the MPP user server are printed to the file. The output log follows the common output format of OpenFrame in the same way as the OUT log.

5.2.2. BMP User Server

The BMP user server creates logs in the SPOOL log directory as the application program is executed in the form of a batch job using JCL. Unlike the MPP user server, the BMP user server manages logs through a directory separated by JOBID by using the same server module in one IMS system. It also follows the common output format of OpenFrame.



For more information, refer to *OpenFrame Batch Guide*.

Appendix A: IMSBATCH Procedure

This appendix describes how to use the OSI IMSBATCH procedure, which is provided in order to use BMP in IMS/DC.

BMP in IMS/DC is started by using a utility named DFSRRC00 and usually uses the IMSBATCH procedure. In OSI, the IMSBATCH procedure is not provided. But just as in IMS/DC, it starts BMP by using a utility named DFSRRC00 provided in the HiDB package.

The following is an example of the IMSBATCH procedure. The bold text in the example are the parameter items supported in OSI.

```
// PROC MBR=TEMPNAME,PSB=,IN=,OUT=,
// OPT=N,SPIE=0,TEST=0,DIRCA=000,
// PRLD=,STIMER=,CKPTID=,PARDLI=,
// CPUTIME=,NBA=,OBA=,IMSID=,AGN=,
// SSM=,PREINIT=,RGN=56K,SOUT=A,
// SYS2=,ALTID=,APARM=,LOCKMAX=
//*
//G EXEC PGM=DFSRRC00,REGION=&RGN,
// PARM=(BMP,&MBR,&PSB,&IN,&OUT,
// &OPT&SPIE&TEST&DIRCA,&PRLD,
// &STIMER,&CKPTID,&PARDLI,&CPUTIME,
// &NBA,&OBA,&IMSID,&AGN,&SSM,
// &PREINIT,&ALTID,
// '&APARM',&LOCKMAX)
//STEPLIB DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
// DD DSN=IMS.&SYS2.PGMLIB,DISP=SHR
//PROCLIB DD DSN=IMS.&SYS2.PROCLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=&SOUT,
// DCB=(LRECL=121,RECFM=VBA,BLKSIZE=3129),
// SPACE=(125,(2500,100),RLSE,,ROUND)
```

OSI supports the following parameters out of those provided to DFSRRC00, to run BMP.

Parameter	Description
MBR	Application name.
PSB	PSB name specified differently from the application name.
IN	Input transaction code. If omitted, processed as batch-oriented BMP.
OUT	Output transaction code. Ignored if IN is specified.
IMSID	IMS system identifier used in an OS.

Appendix B: Printer Facilities

This appendix describes the printing functions provided by the OSI system.

B.1. Overview

OSI supports the following two types of printer facility.

- Display HardCopy
- SCS-DATA Printer



Using those printer facilities requires connection to OpenFrame Gateway. For more information about OpenFrame Gateway, refer to *OpenFrame Gateway Web Terminal Guide*.

B.2. Display HardCopy

If a terminal connecting to OpenFrame Gateway is in Display mode (the device type for the emulator is set to IBM-3278-2-E), and the LU of this terminal is defined as printer in the OSI system, all screens transferred to this terminal will be printed by the default printer of the computer.

1. Set the terminal in Display mode (options vary depending on the emulator), and then connect to OpenFrame Gateway.
2. Check whether the LU (Netname) of the terminal is set as printer, in the system server (control region) to log in.
3. Send a message to the terminal from another terminal or program, to check for a successful hard copy printing.



OSI does not send other data than user data through the printer terminal, to prevent unnecessary printing. Therefore, the terminal set as printer is marked as X SYSTEM, by default.

B.3. SCS-DATA Printer

If a terminal connecting to OpenFrame Gateway is in Printer mode (the device type for the emulator is set to IBM-3287-1), and the LU of this terminal is defined as printer in the OSI system, all data sent to this terminal must be SCS type. Therefore, if a terminal sending SCS type data from a user application program is not set in SCS-DATA Printer mode, it cannot operate as intended.

1. Set the terminal in Printer mode (options vary depending on the emulator), and then connect to

OpenFrame Gateway.

2. Check whether the LU (Netname) of the terminal is set as printer, in the system server (control region) to log in.
3. Since a terminal set in Printer mode cannot receive input, use the IP-LU mapping function for auto-login, or run the OSI command `"/OPNDST"`, to log in from outside.
4. Send some SCS data to the terminal to check for a successful printing.

Appendix C: Resource Tables

This appendix describes RDB tables managed in the OpenFrame OSI system.

You can manage resources used in OSI with RDB tables. Use the **osiinit** tool to create and delete tables. For more information, refer to *OpenFrame OSI Tool Reference Guide*.

- **SD and RTSD Information**

The following tables manage SD and RTSD resources.

Table Name	Description
OFM_OSI_SD_APPLCTN	Stores program SD resources.
OFM_OSI_SD_DATABASE	Stores database SD resources.
OFM_OSI_SD_LTERM	Stores logical terminal SD resources.
OFM_OSI_SD_TERMINAL	Stores terminal SD resources.
OFM_OSI_SD_TRANSACT	Stores transaction SD resources.
OFM_OSI_RTSD_APPLCTN	Stores program RTSD resources.
OFM_OSI_RTSD_DATABASE	Stores database RTSD resources.
OFM_OSI_RTSD_LTERM	Stores logical terminal RTSD resources.
OFM_OSI_RTSD_TERMINAL	Stores terminal RTSD resources.
OFM_OSI_RTSD_TRANSACT	Stores transaction RTSD resources.
OFM_OSI_RTSD_MODS	Stores the RTSD resources to be modified by the MODIFY command.

- **CI Information**

The following table manages CI resources (terminal session information).

Table Name	Description
OFM_OSI_CI	Stores terminal session information.

- **MODSTAT Information**

The following table manages MODSTAT resources.

Table Name	Description
OFM_OSI_MODSTAT	Stores MODSTAT information.

- **Message Queue Information**

The following tables manage MQ information.

Table Name	Description
OFM_OSI_MQ	Stores and manages all messages.

- **Region Information**

The following table manages region information.

Table Name	Description
OFM_OSI_REGION	Stores the JOB information of the DR region.
OFM_OSI_REGION_SVR	Stores information about transactions and programs currently running in the DR region.
OFM_OSI_REGION_STATUS	Stores JOBID, STEPSEQ, STATUS, and CHKPT information of the CTL region.

- **Event Information**

The following table manages event information.

Table Name	Description
OFM_OSI_EVENT	Stores event information that occurs in the OSI system.

- **LOG Information**

The following table manages log information.

Table Name	Description
OFM_OSI_LOG	Stores logs recorded through DL/I LOG calls.