Progress Orbix

Orbix® Mainframe

Installation Guide

Version 6.3, July 2009

PROGRESS SOFTWARE

Third Party Acknowledgments:

1. The Product incorporates IBM-ICU 2.6 (LIC-255) technology from IBM. Such technology is subject to the following terms and conditions: Copyright (c) 1995-2009 International Business Machines Corporation and others. All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder. All trademarks and registered trademarks mentioned herein are the property of their respective owners.

2. The Product incorporates IDL Compiler Front End Technology from Sun Microsystems, Inc. Such technology is subject to the following terms and conditions: Copyright 1992, 1993, 1994 Sun Microsystems, Inc. Printed in the United States of America. All Rights Reserved. This product is protected by copyright and distributed under the following license restricting its use. The Interface Definition Language Compiler Front End (CFE) is made available for your use provided that you include this license and copyright notice on all media and documentation and the software program in which this product is incorporated in whole or part. You may copy and extend functionality (but may not remove functionality) of the Interface Definition Language CFE without charge, but you are not authorized to license or distribute it to anyone else except as part of a product or program developed by you or with the express written consent of Sun Microsystems, Inc. ("Sun"). The names of Sun Microsystems, Inc. and any of its subsidiaries or affiliates may not be used in advertising or publicity pertaining to distribution of Interface Definition Language CFE as permitted herein. This license is effective until terminated by Sun for failure to comply with this license. Upon termination, you shall destroy or return all code and documentation for the Interface Definition Language CFE. The Interface Definition Language CFE may not be exported outside of the United States without first obtaining the appropriate government approvals.

INTERFACE DEFINITION LANGUAGE CFE IS PROVIDED AS IS WITH NO WARRANTIES OF ANY KIND INCLUDING THE WARRANTIES OF DESIGN, MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE. INTERFACE DEFINITION LANGUAGE CFE IS PROVIDED WITH NO SUPPORT AND WITHOUT ANY OBLIGATION ON THE PART OF SUN OR ANY OF ITS SUBSIDIARIES OR AFFILIATES TO ASSIST IN ITS USE, CORRECTION, MODIFICATION OR ENHANCEMENT. SUN OR ANY OF ITS SUBSIDIARIES OR AFFILIATES SHALL HAVE NO LIABILITY WITH RESPECT TO THE INFRINGEMENT OF COPYRIGHTS, TRADE SECRETS OR ANY PATENTS BY INTERFACE DEFINITION LANGUAGE CFE OR ANY PART THEREOF. IN NO EVENT WILL SUN OR ANY OF ITS SUBSIDIARIES OR AFFILIATES BE LIABLE FOR ANY LOST REVENUE OR PROFITS OR OTHER SPECIAL, INDIRECT AND CONSEQUENTIAL DAMAGES, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Updated: July 24, 2009

Contents

CHAPTER 1

Installation Prerequisites

Before you install Orbix Mainframe 6.3, check the system requirements, and familiarize yourself with the steps involved in installing the product.

In this chapter

This chapter contains the following sections:

Before You Begin

Overview

This guide describes how to install Orbix Mainframe. Before you begin, visit the Orbix Mainframe 6.3 documentation web page:

http://www.iona.com/support/docs/orbix/mainframe/6.3/index.xml

There you can read the Mainframe Release Notes and check for updates to this guide.[1]

Also, before you install, check the requirements for your installation, as described in "System Requirements" on page 9, and familiarize yourself with the steps involved in installing the product.

Note for existing customers

Orbix Mainframe 6.3 represents a binary compatible upgrade from Orbix Mainframe 6.2. However, even if you are upgrading from 6.2 (or a 6.2 service pack), you must still complete in full the installation tasks described in Chapter 2, as appropriate for your setup.

If you are planning to migrate an existing Orbix 6.2-based domain to this new version, you should review the Mainframe Migration and Upgrade Guide before proceeding with any of the customization tasks described in Chapter 3.

License codes

You must have valid license codes to be able to install and use Orbix Mainframe. You also need additional license keys if you plan to use the following optional features of Orbix Mainframe:

- Artix Transport—to Web service enable existing Orbix applications.
- Cross Memory Transport—to use with the IMS/CICS client adapter, as an alternative to the default APPC transport.
- Enterprise Performance Logging—to enable integration with third-party management/monitoring tools.

For more details, see "Installing an Optional License Key" on page 74.

If you do not have the required licenses, please contact technical support or your account representative before proceeding:
http://www.progress.com/support

1. A date beside a document on the documentation web pages indicates that the document was last updated on that date.

System Requirements

Overview

This section describes the system requirements for installing Orbix Mainframe.

Supported platforms

The supported platforms are:

- IBM z/OS V1R7
- IBM z/OS V1R8
- IBM z/OS V1R9
- IBM z/OS V1R10

Supported compilers

The supported compilers are:

- IBM z/OS ANSI C++ Compiler (as delivered with the supported platform)
- IBM Enterprise COBOL V3.4 and V4.1
- IBM Enterprise PL/I for z/OS V3.6 and V3.7

Supported IMS releases

The supported IMS releases are:

- IMS V9.1
- IMS V10.1

Supported CICS releases

The supported CICS releases are:

- CICS TS V3.1
- CICS TS V3.2

z/OS system requirements

The following basic program temporary fixes (PTFs) are required:

Note: Check http://www.iona.com/support/docs/apars/index.xml for details of PTFs, and for an up-to-date list of IBM maintenance requirements for Orbix products.

Operating System	Required Patches
z/OS 1.7	UK09695, UK10244, UK08059, UA23848
z/OS 1.8	UK19837, UK21780, UA36419

The following PTF is also required if you want to use TLS with Orbix Mainframe:

Operating System	Required TLS Patch
z/OS 1.7	UA23758

IMS requirements

The following PTFs are required for Open Transaction Manager Access (OTMA) if you want to use IMS with Orbix Mainframe:

IMS Version	Required OTMA Patches
IMS V9.1	UK03271, UQ91993

The following PTFs are required for Resource Recovery Service (RRS) if you want to use IMS with Orbix Mainframe:

IMS Version	Required RRS Patches
IMS V9.1	UQ91845, UK09099

CICS requirements

There are currently no PTF requirements for Customer Information Control System (CICS).

Disk space requirements

The approximate amount of disk space required to install Orbix Mainframe on z/OS is:

Files	Space
Work space for installation	312 3390-3 cylinders
Product as installed	608 3390-3 cylinders

The approximate amount of disk space required to install Orbix Mainframe on the optional z/OS UNIX System Services (USS) is:

Files	Space
Work space for installation	3 MB
Product as installed	16 MB

Installation requirements

The following installation requirements apply:

Prerequisite	Notes
C++ runtime libraries	The IBM Language Environment (SCEERUN) and C++ runtime libraries (SCLBDLL) must be available when installing your Orbix Mainframe licenses.
USS privileges	To install the optional z/OS UNIX System Services portion of the product in the default location, you must have root privileges. To install in a non-default location, you must have permission to create files and directories in that location.

Runtime environment requirements

The following runtime environment requirements apply:

Prerequisite	Notes
C++ runtime libraries	The IBM Language Environment (SCEERUN) and C++ runtime libraries (SCLBDLL) must be available when running any Orbix Mainframe program.
Security product	To use the optional SAF plug-in in Orbix Mainframe, an associated profile class must be added to the installed security product. Instructions for doing this are provided in *orbixhlq*.DOC(SAF) which is uploaded as part of the installation process.
USS privileges	User IDs associated with Orbix services, and all client and server user IDs running on z/OS or the optional z/OS UNIX System Services, require an OMVS segment. This does not apply to servers running inside IMS or CICS.
XML Toolkit V1.7	If you plan to use the iSF (IONA Security Framework) feature of Orbix Mainframe, you must make the IBM XML Toolkit for z/OS V1.7 runtime libraries (SIXMLOD1) available to your Orbix application. The XML parser delivered with this version of the toolkit is also referred to as XML4C V5.4.

Development environment requirements

The following development environment requirements apply:

Prerequisite	Notes
C++ compiler	IBM z/OS ANSI C++ Compiler (as delivered with the supported platform)
COBOL compiler	IBM Enterprise COBOL V3.4 and V4.1
PL/I compiler	IBM Enterprise PL/I for z/OS V3.6 and V3.7

Prerequisite	Notes
Region size	The IBM z/OS ANSI C++ compiler requires at least 48 MB of virtual memory to run. It is recommended that at least 192 MB is available for compiles. For telnet or rlogin users, this can be done by adjusting the MAXASSIZE parameter in BPXPRMxx. Users of the TSO OMVS shell must also ensure their region size is large enough in their RACF TSO segment.

TLS requirements

The following requirements apply if you plan to run services or programs with TLS enabled:

- To run the supplied GENCERT JCL, which sets up the various keyrings, you must be authorized to issue the RACDCERT CERTAUTH command. The authority to issue this command is controlled by having CONTROL access to the IRR.DIGTCERT.function resource in the FACILITY class.

  Note: Although having READ and UPDATE access to the IRR.DIGTCERT.function resource grants authority to issue the RACDCERT command within certain limits, you must have CONTROL access to the IRR.DIGTCERT.function, because the supplied GENCERT and DELCERT JCL members respectively create and delete sample CERTAUTH certificates.

  For detailed information about the RACDCERT command, and the authority required to execute each operand, see the IBM publication *OS/390 Security Server (RACF) Command Language Reference*.

- Ensure that the RACF DIGTCERT and DIGTRING general resource classes have been activated. If not, ask your RACF administrator to issue the following commands:

```
SETROPTS CLASSACT(DIGTCERT)
SETROPTS CLASSACT(DIGTRING)
```

- IBM strongly recommends that you issue the RACLIST command on the DIGTCERT class, to improve performance when using digital certificates. If you do not issue the RACLIST command on the DIGTCERT class, digital certificates can still be used, but performance might be affected. For best performance, issue the following command:

```
SETROPTS RACLIST(DIGTCERT)
```

- After creating a new digital certificate, you should refresh the DIGTCERT class by issuing the following command:

```
SETROPTS RACLIST(DIGTCERT) REFRESH
```

  If you do not refresh the DIGTCERT profiles on which the RACLIST command has been issued, RACF still uses the new digital certificate, but performance might be affected.

For more information about creating keyrings and storing digital certificates in RACF, see the IBM publication *OS/390 Security Server (RACF) Security Administrator's Guide*.

Kerberos Authentication Requirements

The Artix Transport component of Orbix Mainframe supports the validation of Kerberos tokens sent to it from off-host Web services clients using either RACF or an off-host iS2 server.

Before Kerberos authentication can be used with Orbix Mainframe a number of steps to enable the Network Authentication Service are required on your z/OS system. Network Authentication Service is a component of IBM's z/OS Security Server and is IBM's implementation of Kerberos Version 5 from the Massachusetts Institute of Technology.

To configure Network Authentication Service on your z/OS system follow the instructions in the section “Making the program operational” in the IBM publication *z/OS Security Server Network Authentication Service Administration - SC24-5926*. Depending on your installation, one or all of these tasks might already have been completed. When complete, you will have the SKRBKDC started task running on your z/OS system with a registry database defined and the required RACF definitions in place.

CHAPTER 2

Installing Orbix Mainframe

This chapter explains how to install Orbix Mainframe. Please read each step in full before proceeding with it, because the text might contain important recommendations or requirements that you should be aware of before proceeding.

In this chapter

This chapter discusses the following topics:

Before You Begin Installing

Overview

The primary Orbix Mainframe distribution is shipped as an IEBCOPY backup file that has been compressed using the TSO `XMIT` command.

Optional installer

When you have installed the primary distribution in the classic MVS environment, you then have the option to install the Unix System Services add-on installer. This is distributed in a TAR file format.

Customizing the product

After you have successfully installed the product on z/OS (and on the optional z/OS UNIX System Services, if you wish), you must perform some customization tasks before you can use the product. These customization tasks are described in "Customizing Orbix Mainframe" on page 33.

Sequence of tasks

You must successfully complete installation before you begin customization. Perform all installation and customization tasks in the order in which they are described in this guide.

Installing on z/OS

Overview

This section describes how to install Orbix Mainframe on z/OS.

Note: You must complete all the steps in this section in the order in which they are presented.

Step 1—Preallocate a data set

Preallocate a z/OS sequential data set with the following information:

Space Units	Tracks
PRIMARY	4700
SECONDARY	100
RECORD FORMAT	FB
RECORD LENGTH	80
BLOCK SIZE	3120

Step 2—Copy the ORBIX.SEQ file

Copy the `ORBIX.SEQ` file from your product CD into the z/OS data set that you preallocated in the preceding step. How you copy the file depends on the type of machine the CD-ROM drive is on. The most convenient way is to use FTP.

The following is an example of the FTP command sequence to transmit the `ORBIX.SEQ` file into the preallocated data set, where the CD drive letter is `d:` and *`XXXX.XXXX`* represents the name of the data set:

```
d:
ftp hostname
ftp> binary
ftp> put ORBIX.SEQ 'XXXX.XXXX'
```

Step 3—Unpack the PDS

After the ORBIX.SEQ file has been copied to z/OS, use the TSO RECEIVE command to unpack the PDS (where *XXXX.XXXX* represents the exact name of the PDS data set that is to be received):

```
RECEIVE INDSN('XXXX.XXXX')
```

Because the preceding command is a TSO command, you must enter it on an ISPF command screen.

You are prompted with restore parameters similar to the following:

```
To receive the Orbix PDS, please specify the following:
DA('HLQ.ORBIX63.PDS') SPACE(5222,100) REL
replacing the ORB as appropriate.
INMR901I Dataset HLQ.ORBIX63.PDS from JOE on NODENAME
INMR906A Enter restore parameters or 'DELETE' or 'END' +
```

You must choose between one of the following:

- Press **Enter**, to restore *XXXX.XXXX* into the default data set, HLQ.ORBIX63.PDS.
- Restore *XXXX.XXXX* into an alternative data set, by entering the command that appears on your screen, and substituting HLQ.ORBIX63.PDS with the dataset name you want to use.

The sequential data set, *XXXX.XXXX*, can now be deleted.

Step 4—Expand the PDS

The *orbixhlq*.PDS($FIRST) member contains JCL to expand the other PDS members into the full Orbix Mainframe installation. The default high-level qualifier for installation data sets is HLQ.ORBIX63. If you want to change the default high-level qualifier to your installation standard, you can use a command as follows in ISPF:

```
C 'HLQ.ORBIX63'   'orbixhlq' ALL
```

In the preceding example, *orbixhlq* represents your high-level qualifier, which can be up to 19 characters, including one or more periods.

Now submit *orbixhlq*.PDS($FIRST) to install Orbix Mainframe.

Note: This step might take several minutes to complete.

Step 5—Customize the Orbix HLQ and job card accounting info

The default Orbix Mainframe installation can be customized as follows:

- Customize the default high-level qualifier
- Customize the JCL job card accounting information

Customize the default high-level qualifier

The default high-level qualifier used in Orbix Mainframe (in JCL members, PROCS, readmes, and configuration files) is `HLQ.ORBIX63`. In each case, this high-level qualifier must be changed to match the high-level qualifier that you used for your installation when you submitted the `$FIRST` job. The *`orbixhlq`*`.PDS($SECOND)` member contains JCL to convert all the references of `HLQ.ORBIX63` in Orbix Mainframe to match your high-level qualifier.

Customize the JCL job card accounting information

The default JCL job card accounting information used in Orbix Mainframe is `(ACCOUNTING-INFO)`. If your installation requires specific job card accounting information, the *`orbixhlq`*`.PDS($SECOND)` member contains JCL to convert all references of `(ACCOUNTING-INFO)` in Orbix Mainframe to match your job card accounting information.

To enable the `$SECOND` job to do this, perform the following steps:

1. Edit the *`orbixhlq`*`.PDS($SECOND)` member, using the following command in ISPF:

```
C 'INSTALHLQ' 'orbixhlq' ALL
```

   In the preceding command, *`orbixhlq`* must match the high-level qualifier you specified in the `$FIRST` job in "Step 4—Expand the PDS" on page 18.

2. Edit the *`orbixhlq`*`.PDS($SECOND)` member, using the following command in ISPF:

```
C 'INSTALACCT' 'acctinfo' ALL
```

   In the preceding command, `acctinfo` is your installation specific job card accounting information.

Please note the following:

- The value for `acctinfo` must not exceed 53 bytes.
- The ISPF editor has limited space to enter a change all command. This may pose a challenge when `acctinfo` is a large value. One way to approach this problem is to issue multiple change all commands, where the large `acctinfo` value is broken up into two smaller values. For example:

```
C 'INSTALACCT' 'acctinfo1_suffix' ALL
C ' _suffix' 'acctinfo2' ALL
```

- If accounting information is ignored by your JES system, you can skip this step. Or, if you prefer, you can specify a blank setting as follows:

```
C 'INSTALACCT' '' ALL
```

3. Submit `$SECOND` to convert all the references of `HLQ.ORBIX63` in Orbix Mainframe to match your high-level qualifier, and to convert all the references of `(ACCOUNTING-INFO)` to match your installation-specific job card accounting information.

Step 6—Customize your locale (if necessary

This is only relevant if you want to run Orbix Mainframe in a locale other than the default IBM-1047 locale, and your system and compiler is also running in a locale other than IBM-1047.

Orbix Mainframe include files and demonstration sources are coded by default in the IBM-1047 locale. Follow these steps if you do not want to run Orbix Mainframe in the default IBM-1047 locale, and your system and compiler are also running in a locale other than IBM-1047:

1. In *`orbixhlq`*`.PDS($THIRD)`, use the following command in ISPF to change the default high-level qualifier to make it match your installation value:

```
C 'HLQ.ORBIX63' 'orbixhlq' ALL
```

In this example, *`orbixhlq`* represents your high-level qualifier, which can be up to 19 characters, including one or more periods.

2. In *orbixhlq*.PDS($THIRD), use the following command in ISPF to change the value of the TO variable, to make it match the locale codeset you want to use (where *IBM-xxx* represents your codeset):

```
C 'IBM-500' 'IBM-xxx' ALL
```

   This command enables you to simultaneously change all occurrences of the default to match your codeset.

3. Submit $SECOND to convert the files to match your installation.

Step 7—Check installed data sets

Note: All datasets with the name *.BD.LOADLIB are Partitioned Dataset Extended (PDSE) files. These are used when you build your CICS/IMS programs using the binder. If you use the pre-linker and linker approach to build your CICS/IMS programs, the PDSE files are not required and may be deleted.

Compare your list of installed data sets with the list shown in Table 1:

Table 1: *List of Installed Data Sets (Sheet 1 of 7)*

Data Set	Description
orbixhlq.ADMIN.GRAMMAR	Contains itadmin grammar files.
orbixhlq.ADMIN.HELP	Contains itadmin help files.
orbixhlq.ADMIN.LOADLIB	Contains Orbix Mainframe administration programs.
orbixhlq.CBL.OBJLIB	Contains programs for Orbix Mainframe COBOL support.
orbixhlq.CONFIG	Contains Orbix Mainframe configuration information.
orbixhlq.DEMO.ARTIX.BLD.JCLLIB	Contains jobs to build the Artix Transport demonstrations.
orbixhlq.DEMO.ARTIX.README	Contains documentation for the Artix Transport demonstrations.

Table 1: *List of Installed Data Sets (Sheet 2 of 7)*

Data Set	Description
orbixhlq.DEMO.CBL.BLD.JCLLIB	Contains jobs to build the COBOL demonstrations.
orbixhlq.DEMO.CBL.COPYLIB	Used to store generated files for the COBOL demonstrations.
orbixhlq.DEMO.CBL.LOADLIB	Used to store programs for the COBOL demonstrations.
orbixhlq.DEMO.CBL.MAP	Used to store name substitution maps for the COBOL demonstrations.
orbixhlq.DEMO.CBL.README	Contains documentation for the COBOL demonstrations.
orbixhlq.DEMO.CBL.RUN.JCLLIB	Contains jobs to run the COBOL demonstrations.
orbixhlq.DEMO.CBL.SRC	Contains program source for the COBOL demonstrations.
orbixhlq.DEMO.CICS.CBL.BD.LOADLIB	Used to store programs built with the binder for the CICS COBOL demonstrations.
orbixhlq.DEMO.CICS.CBL.BLD.JCLLIB	Contains jobs to build the CICS COBOL demonstrations.
orbixhlq.DEMO.CICS.CBL.COPYLIB	Used to store generated files for the CICS COBOL demonstrations.
orbixhlq.DEMO.CICS.CBL.LOADLIB	Used to store programs for the CICS COBOL demonstrations.
orbixhlq.DEMO.CICS.CBL.README	Contains documentation for the CICS COBOL demonstrations.
orbixhlq.DEMO.CICS.CBL.SRC	Contains program source for the CICS COBOL demonstrations.

Table 1: *List of Installed Data Sets (Sheet 3 of 7)*

Data Set	Description
orbixhlq.DEMO.CICS.MFAMAP	Used to store CICS server adapter mapping member information for demonstrations.
orbixhlq.DEMO.CICS.PLI.BD.LOADLIB	Used to store programs built with the binder for the CICS PL/I demonstrations.
orbixhlq.DEMO.CICS.PLI.BLD.JCLLIB	Contains jobs to build the CICS PL/I demonstrations.
orbixhlq.DEMO.CICS.PLI.LOADLIB	Used to store programs for the CICS PL/I demonstrations.
orbixhlq.DEMO.CICS.PLI.PLINCL	Used to store generated files for the CICS PL/I demonstrations.
orbixhlq.DEMO.CICS.PLI.README	Contains documentation for the CICS PL/I demonstrations.
orbixhlq.DEMO.CICS.PLI.SRC	Contains program source for the CICS PL/I demonstrations.
orbixhlq.DEMO.CPP.BLD.JCLLIB	Contains jobs to build the C++ demonstrations.
orbixhlq.DEMO.CPP.GEN	Used to store generated code for the C++ demonstrations.
orbixhlq.DEMO.CPP.H	Contains header files for the C++ demonstrations.
orbixhlq.DEMO.CPP.HH	Contains header files for the C++ demonstrations.
orbixhlq.DEMO.CPP.LOADLIB	Used to store programs for the C++ demonstrations.
orbixhlq.DEMO.CPP.README	Contains documentation for the C++ demonstrations.
orbixhlq.DEMO.CPP.RUN.JCLLIB	Contains jobs to run the C++ demonstrations.

Table 1: *List of Installed Data Sets (Sheet 4 of 7)*

Data Set	Description
orbixhlq.DEMO.CPP.SRC	Contains program source for the C++ demonstrations.
orbixhlq.DEMO.CPP.TWOPCA	Data store for the two-phase commit demonstration server.
orbixhlq.DEMO.CPP.TWOPCB	Data store for the two-phase commit demonstration server.
orbixhlq.DEMO.IDL	Contains IDL for demonstrations.
orbixhlq.DEMO.IMS.CBL.BD.LOADLIB	Used to store programs built with the binder for the IMS COBOL demonstrations.
orbixhlq.DEMO.IMS.CBL.BLD.JCLLIB	Contains jobs to build the IMS COBOL demonstrations.
orbixhlq.DEMO.IMS.CBL.COPYLIB	Used to store generated files for the IMS COBOL demonstrations.
orbixhlq.DEMO.IMS.CBL.LOADLIB	Used to store programs for the IMS COBOL demonstrations.
orbixhlq.DEMO.IMS.CBL.README	Contains documentation for the IMS COBOL demonstrations.
orbixhlq.DEMO.IMS.CBL.SRC	Contains program source for the IMS COBOL demonstrations.
orbixhlq.DEMO.IMS.MFAMAP	Used to store IMS server adapter mapping member information for demonstrations.
orbixhlq.DEMO.IMS.PLI.BD.LOADLIB	Used to store programs built with the binder for the IMS PL/I demonstrations.
orbixhlq.DEMO.IMS.PLI.BLD.JCLLIB	Contains jobs to build the IMS PL/I demonstrations.
orbixhlq.DEMO.IMS.PLI.LOADLIB	Used to store programs for the IMS PL/I demonstrations.

Table 1: *List of Installed Data Sets (Sheet 5 of 7)*

Data Set	Description
orbixhlq.DEMO.IMS.PLI.PLINCL	Used to store generated files for the IMS PL/I demonstrations.
orbixhlq.DEMO.IMS.PLI.README	Contains documentation for the IMS PL/I demonstrations.
orbixhlq.DEMO.IMS.PLI.SRC	Contains program source for the IMS PL/I demonstrations.
orbixhlq.DEMO.IORS	Used to store IORs for demonstrations.
orbixhlq.DEMO.PLI.BLD.JCLLIB	Contains jobs to build the PL/I demonstrations.
orbixhlq.DEMO.PLI.LOADLIB	Used to store programs for the PL/I demonstrations.
orbixhlq.DEMO.PLI.MAP	Used to store name substitution maps for the PL/I demonstrations.
orbixhlq.DEMO.PLI.PLINCL	Used to store generated files for the PL/I demonstrations.
orbixhlq.DEMO.PLI.README	Contains documentation for the PL/I demonstrations.
orbixhlq.DEMO.PLI.RUN.JCLLIB	Contains jobs to run the PL/I demonstrations.
orbixhlq.DEMO.PLI.SRC	Contains program source for the PL/I demonstrations.
orbixhlq.DEMO.TYPEINFO	Optional type information store.
orbixhlq.DOC	Contains miscellaneous documentation.
orbixhlq.DOC.IMAGES	Contains miscellaneous documentation images.
orbixhlq.DOMAINS	Contains Orbix Mainframe configuration information.

Table 1: *List of Installed Data Sets (Sheet 6 of 7)*

Data Set	Description
orbixhlq.INCLUDE.COPYLIB	Contains include file for COBOL programs.
orbixhlq.INCLUDE.H	Contains C++ header files.
orbixhlq.INCLUDE.IT@CAL.H	Contains C++ header files.
orbixhlq.INCLUDE.IT@DSA.CXX	Contains C++ template implementation files.
orbixhlq.INCLUDE.IT@DSA.H	Contains C++ header files.
orbixhlq.INCLUDE.IT@ERR.H	Contains C++ header files.
orbixhlq.INCLUDE.IT@ITL.CXX	Contains C++ template implementation files.
orbixhlq.INCLUDE.IT@ITL.H	Contains C++ header files.
orbixhlq.INCLUDE.IT@MFA.H	Contains C++ header files.
orbixhlq.INCLUDE.IT@OSS.H	Contains C++ header files.
orbixhlq.INCLUDE.IT@TS.H	Contains C++ header files.
orbixhlq.INCLUDE.IT@TSDSA.H	Contains C++ header files.
orbixhlq.INCLUDE.OMG.H	Contains C++ header files.
orbixhlq.INCLUDE.OMG.HH	Contains C++ header files.
orbixhlq.INCLUDE.OMG.IDL	Contains IDL files.
orbixhlq.INCLUDE.ORBIX.H	Contains C++ header files.
orbixhlq.INCLUDE.ORBIX.HH	Contains C++ header files.
orbixhlq.INCLUDE.ORBIX.IDL	Contains IDL files.
orbixhlq.INCLUDE.ORBIX@PD.H	Contains C++ header files.
orbixhlq.INCLUDE.ORBIX@PD.HH	Contains C++ header files.
orbixhlq.INCLUDE.ORBIX@PD.IDL	Contains IDL files.

Table 1: *List of Installed Data Sets (Sheet 7 of 7)*

Data Set	Description
orbixhlq.INCLUDE.ORBIX@SY.CXX	Contains template implementation files.
orbixhlq.INCLUDE.ORBIX@SY.H	Contains C++ header files.
orbixhlq.INCLUDE.ORBIX@XT.HH	Contains C++ header files.
orbixhlq.INCLUDE.ORBIX@XT.IDL	Contains IDL files.
orbixhlq.INCLUDE.PLINCL	Contains include files for PL/I demonstrations.
orbixhlq.JCLLIB	Contains jobs to run Orbix Mainframe.
orbixhlq.LKED	Contains side-decks for the DLLs.
orbixhlq.LOADLIB	Contains binaries & DLLs.
orbixhlq.LPALIB	Contains LPA eligible programs.
orbixhlq.MFA.BD.LOADLIB	Contains DLLS built with the binder required for deployment of Orbix programs in CICS and IMS. If you do not build your CICS/IMS programs with the binder, use *orbixhlq*.MFA.LOADLIB.
orbixhlq.MFA.LOADLIB	Contains DLLS required for deployment of Orbix programs in IMS.
orbixhlq.PLI.OBJLIB	Contains programs for Orbix Mainframe PL/I support.
orbixhlq.PROCLIB	Contains JCL procedures.
orbixhlq.REXX	Contains REXX execs for Orbix Mainframe customization.

Installing on z/OS UNIX System Services

Overview

This section describes how to install the optional Unix System Services distribution for Orbix Mainframe.

This step is only required if you plan to develop and/or deploy Orbix applications in the Unix System Services command-line shell.

Note: If you need to perform the tasks in this section, perform them in the order in which they are presented. Before you proceed ensure that the tasks in "Installing on z/OS" on page 17 have already been completed.

Step 1—Create installation directory

From the UNIX System Services shell on your z/OS system, create a directory for use during the installation. Ensure the file system has the required space for the installation, as specified in "Disk space requirements" on page 11.

Step 2—Transfer tar file to installation directory

Transfer the `orbix_uss.tar` file on the product CD into the installation directory that you created in the preceding step. Ensure the file is transferred without undergoing any conversions. Example 1 shows a sample FTP session from z/OS.

Example 1: *Sample FTP Session from z/OS (Sheet 1 of 2)*

```
$ ftp hostname

IBM FTP CS V1R5

Connecting to: hostname ip-address port: 21.
220-FTPD1 IBM FTP CS V1R5 at hostname, 06:11:21 on 2001-10-22.
220 Connection will close if idle for more than 5 minutes.

NAME (hostname:user):
joe

>>>USER joe
331 Send password please.
PASSWORD:
```

Example 1: *Sample FTP Session from z/OS (Sheet 2 of 2)*

```
>>>PASS
230 joe is logged on.  Working directory is "JOE.".
Command:

cd /home/joe/orbix63
>>>CWD /home/joe/orbix63
250 HFS directory /home/joe/orbix63 is the current working
directory
Command:

bin

>>>TYPE I
200 Representation type is Image
Command:

put /<dir>/orbix_uss.tar /home/joe/orbix63/orbix_uss.tar

>>>PORT ip-address,port
200 Port request OK.
>>>STOR /home/joe/orbix63/orbix_uss.tar
125 Storing data set /home/joe/orbix63/orbix_uss.tar
1658880 bytes transferred.
250 Transfer completed successfully.
1884160 bytes transferred in 12.510 seconds.  Transfer rate
   150.61 Kbytes/sec.
Command:

quit

>>>QUIT
221 Quit command received. Goodbye.
$
```

Step 3—Unpack the tar file

The compressed tar file contains a number of other tar files and an installation script. Unpack the tar file as follows:

```
$ tar -xvopf orbix_uss.tar
```

Step 4—Run the installation script

Run the installation script as follows:

```
$ sh install.sh
```

Note: To use a locale other than IBM-1047, convert the install script before running it, by using the following commands:

```
$ cp install.sh install.sh.orig
$ iconv -f ibm-1047 -t <codeset> install.sh.orig >install.sh
```

Step 5—Accept license agreement

The license agreement dialog appears. Read the license agreement and, if you agree with the conditions, enter `y`.

Step 6—Specify high-level qualifier

You are asked to specify the high-level qualifier where you have installed the product data sets on z/OS. This must be the same as the high-level qualifier that you specified in "Step 3—Unpack the PDS" on page 18. If you chose to accept the default high-level qualifier, `HLQ.ORBIX63`, when you installed on z/OS, press **Enter** to accept the default now. Otherwise, specify the alternative high-level qualifier that you specified in "Step 3—Unpack the PDS" on page 18.

Step 7—Specify UNIX System Services installation directory

You are next asked to specify a directory where the product is to be installed on z/OS UNIX System Services. The location you specify is referred to later in this guide as *`OrbixInstallDir`*. The default is `/opt/iona` on UNIX. Specify your own directory choice or press **Enter** to accept the default.

Step 8—Specify codeset

You are now asked what codeset the product should use. The default is based on the current `LC_ALL` setting. Specify the codeset you wish to use or press **Enter** to accept the default.

Note: If you choose a codeset other than IBM-1047, there is a slight delay while the script converts all the relevant files.

At this point, the installation script unpacks the tar files into *`OrbixInstallDir`* and deletes each tar file.

Step 9—Delete original tar file

When the installation is complete under *OrbixInstallDir* you can delete the original tar file and the installation script.

Step 10—Connect to configuration domain

Issue the following command to connect to the existing configuration domain:

```
. OrbixInstallDir/etc/bin/default-domain_env.sh
```

Step 11—Include SSL load library in STEPLIB (if necessary)

This is only relevant if you want to use TLS from z/OS UNIX System Services. If so, you must include the IBM System SSL load library in your STEPLIB. Use the following command to do this (where *GSK-LOAD-LIBRARY* represents the name of your System SSL load library):

```
export STEPLIB=GSK-LOAD-LIBRARY:$STEPLIB
```

CHAPTER 3

Customizing Orbix Mainframe

This section describes the customization tasks to be performed after installing Orbix Mainframe before you can use it.

In this chapter

This chapter discusses the following topics:

Standard Customization Tasks

Overview

This section describes standard customization tasks that you must perform before you can use Orbix Mainframe. You must perform these customization tasks in the order in which they are presented.

Note: If you are not using SSL, all the steps in this section are relevant. If you are using SSL, only steps 1–5 are relevant and further customization tasks are described in "SSL/TLS Customization" on page 45.

Step 1—Change dataset name defaults in ORXVARS

Verify that the following variables in the `ORXVARS` member, which represent system data set high-level qualifiers, match those installed on your z/OS system:

`TCPIP` This is the high-level qualifier for the IBM TCP/IP SEZARNT1 and SEZACMTX libraries. For example:

```
SET TCPIP=TCPIP
```

`TCPIPCFG` This is the TCP/IP configuration file to be used by Orbix programs. It is the file referred to as the TCPIP.DATA file in the IBM TCP/IP publications. For example:

```
SET TCPIPCFG=SYS1.TCPPARMS(TCPDATA)
```

`CEE` This is the high-level qualifier for the IBM Language Environment (L/E) C data sets, such as the SCEELKED library needed to link the sample demonstrations. For example:

```
SET CEE=CEE
```

`CBC` This is the high-level qualifier for the IBM C++ compiler data sets, such as the SCLBDLL library. For example:

```
SET CBC=CBC
```

`CICSHLQ` If you are using CICS, `CICSHLQ` should be set to the high-level qualifier where CICS is installed.

`IMSHLQ` If you are using IMS, `IMSHLQ` should be set to the high-level qualifier where IMS is installed.

`CBLPRFX` If you are building COBOL applications, `CBLPRFX` should be set to the high-level qualifier where the COBOL compiler is installed.

PLIPRFX	If you are building PL/I applications, PLIPRFX should be set to the high-level qualifier where the PL/I compiler is installed.
SSLHLQ	If you are deploying secure Orbix applications using SSL/TLS, SSLHLQ should be set to the high-level qualifier of your System SSL installation

If the supplied defaults do not match those in use at your site, change them where appropriate.

Step 2—Additional customizations in ORXVARS

The locale settings are only relevant if you want to override the system locale when developing and deploying Orbix applications in a locale other than IBM-1047. The time zone setting is only relevant if you wish to deploy Orbix applications in a time zone other than the default system time zone.

Locale customization

If you plan to run Orbix Mainframe in a locale other than IBM-1047, and your system and compiler are running in a locale other than the locale in which you want to run Orbix Mainframe, set the following variables in *orbixhlq*.PROCLIB(ORXVARS):

ITLOCALE	This is the locale in which you want to run Orbix Mainframe. For example, to have Orbix Mainframe run in the Swiss German locale, set ITLOCALE as follows: `SET ITLOCALE='LC_ALL=DE_CH.IBM-500'`
CPPLCALE	This is the locale in which you want to run the C++ compiler. For example, to have the C++ compiler run in the Swiss German locale, set CPPLCALE as follows: `SET CPPLCALE='LOCALE(DE_CH.IBM-500)'`

Time zone customization

If you plan to run Orbix Mainframe applications using a time zone that differs to your system's default time zone setting, you may set the `ITTIMEZ` setting.

For example, to run using GMT-1 Daylight Savings Time, set `ITTIMEZ` as follows:

```
SET  ITTIMEZ='TZ="GMT-1GDT"'
```

JCL arguments

In JCL, the parameter length (that is, the length of the PARM field) can be up to 100 bytes. The RPARM JCL symbolic and PPARM JCL symbolic often comprise the data that is passed in the PARM field. This might pose problems when passing `-ORB` arguments along with any locale arguments, because the total length of the PARM field might then exceed 100 bytes.

To avoid this potential problem, an optional DD name is supplied in the JCL components in your Orbix Mainframe installation, as follows:

```
//ORBARGS DD *
```

When the preceding DD name is coded in the JCL, arguments of the form `-ORB`*`xxx yyy`* can be specified here rather than in the PARM field. For example:

```
//ORBARGS DD *
-ORBname iona_utilities.imsa
```

The `ORBname` is supplied using the `ORBARGS` DD name rather than on the RPARM symbolic. This yields a saving of 27 bytes of the 100 that are available on the PARM field.

The following rules apply when using the `ORBARGS` DD name:

- Use it only for arguments of the form `-ORB`*`xxx yyy`*. Do not use it for other arguments.
- Code only one `-ORB`*`xxx`* argument per line.
- Up to a maximum of 16 lines can be coded.
- Each line must be of the form `-ORB`*`xxx yyy`*, where *`xxx`* represents the -ORB argument, and *`yyy`* represents the value for that argument.
- If multiple lines are coded, an invalidly coded line invalidates all others.

- If the same argument is coded both on the RPARM and in ORBARGS, the RPARM takes precedence.
- ORBARGS can be used with `DD *` or, alternatively, with `DD DSN=` pointing to a fixed block data set with a logical record length of 80 bytes.

Step 3—Choose a configuration domain name

The *orbixhlq*`.CONFIG(ORBARGS)` PDS contains the following setting, which specifies the default configuration domain name:

```
-ORBdomain_name DEFAULT@
```

If you wish, you can specify an alternative configuration domain name other than `DEFAULT@`. The name can be up to eight characters long.

When running Orbix Mainframe clients, servers, or services, you can specify the configuration domain name in JCL in either of the following ways:

- Use the `ORBARGS DD` statement, which allows a `-ORBdomain_name` to be specified inside the file that is pointed to by the `ORBARGS` DD statement. For example:

  ```
  //ORBARGS DD *
  -ORBdomain_name DEFAULT@
  /*
  ```

- Use the `ITDOMAIN DD` statement, which points to *orbixhlq*`.CONFIG(`*domname*`)`, where *domname* represents the configuration domain name. For example:

  ```
  //ITDOMAIN DD DSN=orbixhlq.CONFIG(DEFAULT@),DISP-SHR
  ```

  If the `ITDOMAIN DD` statement specifies a PDS with a non-existent member name, a `CORBA::INITIALIZE` exception with a minor code of `ERROR_IN_DOMAIN` is thrown.

  Note: The `ITDOMAIN DD` statement cannot be used in JCL that updates settings in the configuration, because it might conflict with a service that is currently running and using this `ITDOMAIN DD` statement. If you do this, an error occurs on opening the configuration file. In this case, the `ORBARGS DD` statement should be used instead.

If you do not take either of the preceding approaches to specify a configuration domain name, the default name of `DEFAULT@` is used.

Note: You can also specify the configuration domain name in the `PARM` field. However, because the `PARM` field is limited to 100 characters, this can cause JCL errors if other items are also specified. It is therefore recommended that, if you want to specify an alternative configuration domain name, you should use either of the preceding approaches instead of using JCL `PARM`.

Step 4—Set up your license file

The product license information that you have received by e-mail needs to be transferred to the mainframe and formatted before it can be used by Orbix Mainframe. Follow these steps:

1. Preallocate a small data set on the host with the following information:

Space Units	Tracks
PRIMARY	1
SECONDARY	1
RECORD FORMAT	VB
RECORD LENGTH	500 (or greater)
BLOCK SIZE	0

2. Use FTP to transfer the license as a text file into the newly created data set. The following is an example of the FTP command sequence, where the drive letter is `C:` and *`XXXX.XXXX`* represents the name of the data set you have just allocated:

```
C:
ftp hostname
ftp> asc
ftp> put license.txt 'XXXX.XXXX'
```

3. After the license text file has been copied to z/OS, edit the JCL *`orbixhlq`*`.JCLLIB(ORXCOPY)`, as follows:

   In the `IN DD` statement, replace where it says *`<your VB dataset here>`* with the name of the data set that contains your license file.

4. Submit ORXCOPY to copy the license file to *orbixhlq*.CONFIG(LICENSES). The ORXCOPY job copies the license file from a variable-length record file into the fixed-length record license file used by Orbix Mainframe. It splits long lines across records, delimiting them with a backslash in column 72.

Step 5—Convert your license file

This is only relevant if you want to run Orbix Mainframe in a locale other than the default locale IBM-1047.

If so, the steps are:

1. In *orbixhlq*.PDS($FOURTH), use the following command in ISPF to change the default high-level qualifier, to make it match your installation value (where *orbixhlq* represents your high-level qualifier, which can be up to 19 characters, including one or more periods):

   ```
   C 'HLQ.ORBIX63'   'orbixhlq' ALL
   ```

2. In *orbixhlq*.PDS($FOURTH), use the following command in ISPF to change the value of the TO variable, to make it match the locale codeset in which you want to run Orbix Mainframe (where IBM-*xxx* represents the codeset):

   ```
   C 'IBM-500'   'IBM-xxx' ALL
   ```

   The preceding command lets you simultaneously change all occurrences of the default to make it match your codeset.

   Note: If your system and compiler are installed in IBM-1047, make a copy of your original license file at this point and keep it. This is necessary for running the Orbix IDL compiler.

3. Submit *orbixhlq*.PDS($FOURTH) to convert your license file.

4. *This is only relevant if your system and compiler are not installed in IBM-1047, and you want to run Orbix Mainframe in a different locale to these.*
   i. Make a copy of the license file that you converted in point 2, and keep it. This is necessary for running Orbix Mainframe in the locale that you specified in point 2.
   ii. In *orbixhlq*.PDS($FOURTH), use the following command in ISPF to change the value of the TO variable, to make it match the locale codeset in which you want to run the Orbix IDL compiler (that is, the locale in which your system and compiler are installed):

   ```
   C 'IBM-xxx'   'IBM-yyy' ALL
   ```

   In the preceding example, IBM-*xxx* represents the locale codeset (that you specified in point 2) in which you want to run Orbix Mainframe, and IBM-*yyy* represents the locale codeset in which you want to run the Orbix IDL compiler.
   iii. In *orbixhlq*.PDS($FOURTH), use the following command in ISPF to change the value of the FROM variable from IBM-1047, to make it match the locale codeset (that you specified in point 2) in which you want to run Orbix Mainframe:

   ```
   C 'IBM-1047'   'IBM-xxx' ALL
   ```

   iv. Submit *orbixhlq*.PDS($FOURTH) to convert your license file to match the locale where you want to run the Orbix IDL compiler.

Step 6—Create a configuration file

Before you can use any of the supplied Orbix Mainframe services, values must be given to some configuration variables and the services must be run in prepare mode. JCL is provided in *orbixhlq*.JCLLIB(DEPLOY1) to allow you to do this.

Note: Before updating the configuration file, you should read at least part 1 of the CORBA Administrator's Guide.

Follow these steps to customize the configuration variables:

1. In the MAKECON step of *orbixhlq*.JCLLIB(DEPLOY1), customize each of the following configuration items:

   LOCAL_HOSTNAME="";

   Specify the fully qualified local hostname.

   LOCAL_HFS_ROOT="";

   Specify the HFS path of the optional z/OS UNIX System Services directory to be used by the Orbix services for databases and logs. For example, "/opt/iona/orbix63";

   When you start any of the Orbix services, log files and persistent data are stored in the z/OS UNIX System Services directory that you specify via this setting.

   Note: You must have write access to the HFS at this location.

   LOCAL_LOCATOR_PORT="5001";

   Specify a unique TCP/IP port to be used by the locator.

   LOCAL_NODE_DAEMON_PORT="5002";

   Specify a unique TCP/IP port to be used by the node daemon.

2. Still in the MAKECON step of *orbixhlq*.JCLLIB(DEPLOY1), go to the following line:

   ```
   //SYSUT2 DD DISP=SHR,DSN=&ORBIXCFG(DEFAULT@)
   ```

   Ensure that the member name for the //SYSUT2 PDS (DEFAULT@) matches the configuration domain name specified in *orbixhlq*.CONFIG(ORBARGS) in "Step 3—Choose a configuration domain name" on page 37.

3. In the MAKEDOM step of *orbixhlq*.JCLLIB(DEPLOY1), change FILEDOMA in the SELECT MEMBER=((BASETMPL,FILEDOMA)) line to the value specified in the include statement of the MAKECON step. (FILEDOMA is the default value. If it was not changed in the MAKECON step, you need not change it here).

If you are deploying to the same domain a second time, and you want to overlay the file domain member, you can modify the `SELECT` line as follows (with the appropriate changes made to `FILEDOMA`, if necessary):

```
SELECT MEMBER=((BASETMPL,FILEDOMA,R))
```

Step 7—Update configuration and prepare to run daemons

Now submit *orbixhlq*`.JCLLIB(DEPLOY1)`. This does all the following:

- It creates a configuration domain in *orbixhlq*`.CONFIG`. By default, the configuration domain is created in the `DEFAULT@` member.
- It copies the appropriate configuration file template to *orbixhlq*`.DOMAINS(FILEDOMA)`.

  Note: The default is `FILEDOMA`. This might have been customized to an alternative name in "Step 6—Create a configuration file" on page 40. If so, the configuration file template is copied to that member name instead.

- It runs the locator and node daemon in prepare mode.

  Note: The locator and node daemon must be run in prepare mode before you can start Orbix Mainframe. Running the locator and node daemon in prepare mode generates stringified IORs for them.

- It copies the IORs generated for the locator and node daemon to the `LOCAL_LOCATOR_REFERENCE` and `LOCAL_NODE_DAEMON_REFERENCE` configuration variables in *orbixhlq*`.CONFIG(DEFAULT@)`.

  Note: The *orbixhlq*`.CONFIG(IORLCT)` member contains two IORs—`IT_Locator` and `IT_SingleLocator`. The IOR for `IT_Locator` is used.

The `LOCATOR` step produces a message, as shown in the following example. This message can be safely ignored, because it is merely informational:

```
Wed, 11 May 2005 16:57:36.0000000 [host:DEPLOY1,A=004A]
    (IT_LOCATOR:150) I - EndpointCache setup called
```

The NODEDAEM step produces a message, as shown in the following example. This message can be safely ignored, because there is no native activator supplied in this release of Orbix Mainframe:

```
Wed, 11 May 2005 16:57:36.0000000 [host:DEPLOY1,A=0016]
   (IT_ACTIVATOR:0) W - Activation feature not supported in the
   batch environment
```

When running the prepare jobs, the permissions set for the HFS files and directories that are created are based on a default umask of 022. If you require other permissions (for example, to allow multiple users in the same group to run Orbix services (not at the same time)), specify a umask of 002. To do this, add an RPARM to each prepare step. For example, update the locator prepare step in the *orbixhlq*.JCLLIB(DEPLOY1) JCL as follows:

```
//*
//* Prepare the locator
//*
//PREPLCT   EXEC PROC=ORXG,
//          PROGRAM=ORXLOCAT,
//          RPARM='ENVAR(_EDC_UMASK_DFLT=002)',
//          PPARM='prepare -publish_to_file=DD:ITCONFIG(IORLCT)'
//*
```

If you are not running in the default locale, add the locale to the RPARM, as follows:

```
//*
//* Prepare the locator
//*
//PREPLCT  EXEC PROC=ORXG,
//       PROGRAM=ORXLOCAT,
//      RPARM='ENVAR(_EDC_UMASK_DFLT=002,LC_ALL=DE_CH.IBM-500)',
//       PPARM='prepare -publish_to_file=DD:ITCONFIG(IORLCT)'
//*
```

You might wish to set a umask for the locator, node daemon, IFR, and Naming Service, in which case you must update the JCL in *orbixhlq*.JCLLIB(DEPLOY1) and *orbixhlq*.JCLLIB(DEPLOY2).

Step 8—Run daemons in run mode

You are now ready to start the locator and node daemon. Follow these steps:

1. Edit the JCL in *orbixhlq*.JCLLIB(LOCATOR) and *orbixhlq*.JCLLIB(NODEDAEM), to change the default high-level qualifier, so that it reflects the proper value for your installation.
2. Submit the *orbixhlq*.JCLLIB(LOCATOR) job. After submitting it, wait until you see the following message:

```
+ORX2001I ORB iona_services.locator STARTED
   (hostname:LOCATOR,A=nnnn)
```

3. Submit the *orbixhlq*.JCLLIB(NODEDAEM) job. After submitting it, wait until you see the following message:

```
+ORX2001I ORB iona_services.node_daemon STARTED
   (hostname:NODEDAEM,A=nnnn)
```

SSL/TLS Customization

Overview

This section is only relevant if you want to run the services (for example, the locator daemon, node daemon, CICS or IMS adapters) or the supplied demonstrations, with SSL enabled.

Note: If you need to perform the tasks in this section, perform them in the order in which they are presented. Before you proceed ensure that steps 1–5 in "Standard Customization Tasks" on page 34 have already been completed.

Step 1—Create SSL certificates

To run the services (for example, the locator daemon, node daemon, CICS or IMS adapters) or the supplied demonstrations, with SSL enabled, you must generate some sample certificates for these services and programs to use. A job is provided in *orbixhlq*.JCLLIB(GENCERT) to do this.

The GENCERT JCL contains the default high-level qualifier, so first change it to reflect the proper value for your installation. You must also change the user ID to make it match the user ID that the Orbix services use. Then submit *orbixhlq*.JCLLIB(GENCERT).

Step 2—Add System SSL load library

The Orbix SSL runtime uses IBM System SSL modules. Therefore, when running with SSL enabled, you must ensure that the System SSL load library is in the MVS search path for your Orbix application. If this library is not in the system search path by default, you must include it in the STEPLIB for your application.

For example, if you are using *orbixhlq*.PROCLIB(ORXG) to run your application, you need to uncomment the appropriate line in the DD concatenation so that the SSLLOAD library is included in the search path:

```
//         DD  DISP=SHR,DSN=&SSLLOAD
```

Step 3—Create a configuration file

Before you can use any of the supplied Orbix Mainframe services, values must be given to some configuration variables and the services must be run in prepare mode. JCL is provided in *orbixhlq*.JCLLIB(DEPLOYT) to allow you to do this.

Note: Before updating the configuration file, you should read at least part 1 of the CORBA Administrator's Guide.

Follow these steps to customize the configuration variables:

1. In the MAKECON step of *orbixhlq*.JCLLIB(DEPLOYT), customize each of the following configuration items:

   ```
   LOCAL_HOSTNAME="";
   ```

   Specify the fully qualified local hostname.

   ```
   LOCAL_HFS_ROOT="";
   ```

   Specify the HFS path of the z/OS UNIX System Services directory to be used by the Orbix services for databases and logs. For example:

   ```
   "/opt/iona/orbix63";
   ```

   When you start any of the Orbix services, log files and persistent data are stored in the z/OS UNIX System Services directory that you specify via this setting.

   Note: You must have write access to the HFS at this location.

   ```
   LOCAL_LOCATOR_PORT="5001";
   ```

   Specify the TCP/IP port to be used by the locator for non-secure conversations.

   ```
   LOCAL_NODE_DAEMON_PORT="5002";
   ```

   Specify a unique TCP/IP port to be used by the node daemon for non-secure conversations.

   ```
   LOCAL_TLS_LOCATOR_PORT="5101";
   ```

   Specify a unique TCP/IP port to be used by the locator for secure conversations.

```
LOCAL_TLS_NODE_DAEMON_PORT="5102";
```

Specify a unique TCP/IP port to be used by the node daemon for secure conversations.

```
LOCAL_SSL_USER_SAF_KEYRING="ORBXRING";
```

Specify the name of the RACF keyring that contains your certificates.

2. Still in the MAKECON step of *orbixhlq*.JCLLIB(DEPLOYT), go to the following line

```
//SYSUT2 DD DISP=SHR,DSN=&ORBIXCFG(DEFAULT@
```

Ensure that the member name for the //SYSUT2 PDS (DEFAULT@) matches the configuration domain name specified in *orbixhlq*.CONFIG(ORBARGS) in "Step 3—Choose a configuration domain name" on page 37.

3. In the MAKEDOM step of *orbixhlq*.JCLLIB(DEPLOYT), change TLSBASE and TLSDOMA in the following lines

```
SELECT MEMBER=((BASETMPL,TLSBASE))
SELECT MEMBER=((TLSTMPL,TLSDOMA))
```

to the value specified in the include statement of the MAKECON step. (TLSBASE and TLSDOMA are the default values. If they were not changed in the MAKECON step, you need not change it here).

If you are deploying to the same domain a second time, and you want to overlay the file domain member, you can modify the SELECT lines as follows:

```
SELECT MEMBER=((BASETMPL,TLSBASE,R))
SELECT MEMBER=((TLSTMPL,TLSDOMA,R))
```

After you have set the preceding variables in *orbixhlq*.JCLLIB(DEPLOYT), change the default high-level qualifier in DEPLOYT, to reflect the proper value for your installation.

Step 4—Update configuration and prepare to run daemons

Now submit *orbixhlq*.JCLLIB(DEPLOYT). This does all the following:

- It creates a configuration domain in *orbixhlq*.CONFIG. By default, the configuration domain is created in the DEFAULT@ member.
- It copies the appropriate configuration file templates to *orbixhlq*.DOMAINS(TLSBASE) and *orbixhlq*.DOMAINS(TLSDOMA). The TLSBASE member contains the common configuration items that are used in both insecure and secure domains, while the TLSDOMA member contains only TLS-specific configuration items. Both of these are included by default in the DEFAULT@ member.

  Note: The defaults are TLSBASE & TLSDOMA These might have been customized to alternative names in "Step 3—Create a configuration file" on page 46. If so, the configuration file templates are copied to those member names instead.

- It runs the locator and node daemon in prepare mode.

  Note: The locator and node daemon must be run in prepare mode before you can start Orbix Mainframe. Running the locator and node daemon in prepare mode generates stringified IORs for them.

- It copies the IORs generated for the locator and node daemon to the LOCAL_LOCATOR_REFERENCE and LOCAL_NODE_DAEMON_REFERENCE configuration variables in *orbixhlq*.CONFIG(DEFAULT@).

  Note: The *orbixhlq*.CONFIG(IORLCT) member contains two IORs—IT_Locator and IT_SingleLocator. The IOR for IT_Locator is used.

The LOCATOR step produces a message, as shown in the following example. This message can be safely ignored, because it is merely informational:

```
Wed, 11 May 2005 16:57:36.0000000 [host:DEPLOY1,A=004A]
   (IT_LOCATOR:150) I - EndpointCache setup called
```

The NODEDAEM step produces a message, as shown in the following example. This message can be safely ignored, because there is no native activator supplied in this release of Orbix Mainframe:.

```
Wed, 11 May 2005 16:57:36.0000000 [host:DEPLOYT,A=0016]
   (IT_ACTIVATOR:0) W - Activation feature not supported in the
   batch environment
```

When running the prepare jobs, the permissions set for the HFS files and directories that are created are based on a default umask of 022. If you require other permissions (for example, to allow multiple users in the same group to run Orbix services (not at the same time)), specify a umask of 002. To do this, add an RPARM to each prepare step. For example, update the locator prepare step in the *orbixhlq*.JCLLIB(DEPLOYT) JCL as follows:

```
//*
//* Prepare the locator
//*
//PREPLCT   EXEC PROC=ORXG,
//          PROGRAM=ORXLOCAT,
//          RPARM='ENVAR(_EDC_UMASK_DFLT=002)',
//          PPARM='prepare -publish_to_file=DD:ITCONFIG(IORLCT)'
//*
```

If you are not running in the default locale, add the locale to the RPARM, as follows:

```
//*
//* Prepare the locator
//*
//PREPLCT  EXEC PROC=ORXG,
//       PROGRAM=ORXLOCAT,
//      RPARM='ENVAR(_EDC_UMASK_DFLT=002,LC_ALL=DE_CH.IBM-500)',
//       PPARM='prepare -publish_to_file=DD:ITCONFIG(IORLCT)'
//*
```

You might wish to set a umask for the locator, node daemon, IFR, and Naming Service, in which case you must update the JCL in *orbixhlq*.JCLLIB(DEPLOYT).

Step 5—Run daemons in run mode

You are now ready to start the locator and node daemon. Follow these steps:

1. Edit the JCL in *orbixhlq*.JCLLIB(LOCATOR) and *orbixhlq*.JCLLIB(NODEDAEM), to change the default high-level qualifier, so that it reflects the proper value for your installation.
2. Submit the *orbixhlq*.JCLLIB(LOCATOR) job. After submitting it, wait until you see the following message:

```
+ORX2001I ORB iona_services.locator STARTED
   (hostname:LOCATOR,A=nnnn)
```

3. Submit the *orbixhlq*.JCLLIB(NODEDAEM) job. After submitting it, wait until you see the following message:

```
+ORX2001I ORB iona_services.node_daemon STARTED
   (hostname:NODEDAEM,A=nnnn)
```

Naming Service and IFR Customization

Overview

This section is only relevant if you want to use the Naming Service or Interface Repository (IFR) components of Orbix Mainframe. It describes the customization tasks to be performed before using them.

Note: If you need to perform the tasks in this section, perform them in the order in which they are presented. Before you proceed ensure that the tasks in "Standard Customization Tasks" on page 34 and "SSL/TLS Customization" on page 45 have already been completed, as appropriate.

Step 1—Prepare to run the naming service and IFR

Before proceeding with this step ensure that the locator and node daemon are running.

If you want to use the Naming Service or Interface Repository (IFR) components of Orbix Mainframe, you must run them first in prepare mode. A job is provided to do this in *orbixhlq*.JCLLIB(DEPLOY2). This JCL contains the default high-level qualifier, so first change it to reflect the proper value for your installation before you submit it.

Running the Naming Service and Interface Repository in prepare mode generates stringified IORS for them. The DEPLOY2 JCL automatically writes the IORs for the Naming Service and IFR to *orbixhlq*.CONFIG(IORNAM) and *orbixhlq*.CONFIG(IORIFR) respectively. It then copies these IORs into the LOCAL_NAMING_REFERENCE and LOCAL_IFR_REFERENCE variables in *orbixhlq*.CONFIG(DEFAULT@).

Note: The *orbixhlq*.CONFIG(IORNAM) member contains two IORs—NameService and IT_SingleNameService. The IOR for NameService is used.

Step 2—Run the naming service and IFR in run mode

You are now ready to start the Naming Service and/or IFR. The following sample JCL may be submitted to start these CORBA services:

- *orbixhlq*.JCLLIB(NAMING)
- *orbixhlq*.JCLLIB(IFR)

IMS Server Adapter Customization

Overview

This section is only relevant if you want to use the IMS server adapter component of Orbix Mainframe. It describes the customization tasks to be performed before using the adapter.

Note: If you need to perform the tasks in this section, perform them in the order in which they are presented. Before you proceed ensure that the tasks in "Naming Service and IFR Customization" on page 51 have already been completed, if you intend to use the IFR as the type repository for the IMS server adapter.

Step 1—Avoid known problems

To avoid known problems, it is recommended that the PTFs listed in "System Requirements" on page 9 are applied.

Step 2—Configure OTMA or APPC for IMS

To use the IMS server adapter, either of the following must be enabled for IMS:

- OTMA and the OTMA Callable Interface
- APPC

For details of how to configure OTMA for IMS see the IBM publication *Open Transaction Manager Access Guide and Reference, SC26-8743*.

For details of how to configure APPC for IMS see the IBM publication *MVS Planning: APPC/MVS Management, GC28-1807*. Additionally, for specific details on the use of APPC by IMS, see the chapter on administration of APPC/IMS and LU devices in the IBM publication *IMS/ESA Administration Guide: Transaction Manager, SC26-8104*.

Step 3—Verify adapter configuration prerequisites

Verify that the configuration variables in the `imsa` scope of your configuration file have been changed to match those specified in the IMS control region that you are connecting to. In particular, ensure that you have specified the location of the adapter mapping member that is to be used. For details of how to do this, and the defaults used when the entries are not specified via configuration, see the IMS Adapters Administrator's Guide.

Step 4—Customize IMS JCL

The following libraries should be added to the IMS message region's STEPLIB concatenation:

```
DD DSN=orbixhlq.MFA.LOADLIB,DISP=SHR
DD DSN=orbixhlq.DEMO.IMS.CBL.LOADLIB,DISP=SHR
DD DSN=orbixhlq.DEMO.IMS.PLI.LOADLIB,DISP=SHR
```

If you build your IMS programs with the binder, add the following libraries to the IMS message region's STEPLIB concatenation instead:

```
DD DSN=orbixhlq.MFA.BD.LOADLIB,DISP=SHR
DD DSN=orbixhlq.DEMO.IMS.CBL.BD.LOADLIB,DISP=SHR
DD DSN=orbixhlq.DEMO.IMS.PLI.BD.LOADLIB,DISP=SHR
```

Check if the following entries are already defined in the IMS message region's JCL. If not, they should be added, to ensure you receive all output from your IMS servers (recycle the message regions to pick up these libraries):

```
SYSPRINT DD SYSOUT=*
CEEDUMP DD SYSOUT=*
CEEOUT DD SYSOUT=*
SYSOUT DD SYSOUT=*
```

Step 5—Run the IMS server adapter in prepare mode

Before proceeding with this step ensure that the locator daemon and node daemon are all running. Also ensure that the relevant IMS region is active.

If you want to use the IMS server adapter, you must run it first in prepare mode. Submit the JCL in *orbixhlq*.JCLLIB(PREPIMSA) to run the IMS server adapter in prepare mode.

Running the IMS server adapter in prepare mode generates a stringified IOR for it and writes this IOR to *orbixhlq*.CONFIG(IORIMSA). An IOR is also generated for imsraw. The IT_MFA and imsraw IORs are automatically added to the configuration file by the prepare step.

Step 6—Run the IMS server adapter in run mode

You are now ready to start the IMS server adapter. Submit the JCL in *orbixhlq*.JCLLIB(IMSA) to run the IMS server adapter.

CICS Server Adapter Customization

Overview

This section is only relevant if you want to use the CICS server adapter component of Orbix Mainframe. It describes the customization tasks to be performed before using the adapter.

Note: If you need to perform the tasks in this section, perform them in the order in which they are presented. Before you proceed ensure that the tasks in "Naming Service and IFR Customization" on page 51 have already been completed, if you intend to use the IFR as the type repository for the CICS server adapter.

Step 1—Avoid known problems

It is recommended that the PTFs listed in "System Requirements" on page 9 are applied, to avoid known problems.

Step 2—Configure IRC for CICS

To use the CICS server adapter, support for Inter Region Communication (IRC) must be enabled in CICS. In general, IRC can be enabled by specifying the CICS parameter `IRC=YES` or `IRCSTRT=YES` (depending on the version), and by using the default CICS definitions in the CSD group `DFH$EXCI` that are delivered with CICS by default. These definitions are sufficient to get started and they can be used as models for any future requirements you might have. The following message is issued if this support is active and installed correctly within CICS:

```
DFHSI1519I CICS The inter-region communication session was
   successfully started.
```

If this message is not issued, you cannot use the CICS server adapter to communicate with that CICS region.

Step 3—Configure EXCI or APPC for CICS

To use the CICS server adapter, you must enable either of the following for CICS:

- EXCI
- APPC

For details of how to configure EXCI for CICS see the IBM publication *CICS External Interfaces Guide, SC33-1944*.

For details of how to configure APPC for CICS see the IBM publication *MVS Planning: APPC/MVS Management, GC28-1807*. Additionally, for specific details on the use of APPC by CICS, see the chapter on defining APPC links in the IBM publication *CICS Intercommunication Guide, SC33-1695*.

Step 4—Define required resources to CICS

Before you can run Orbix Mainframe CICS applications in your CICS region, you must perform a number of additional steps to enable CICS to support Orbix Mainframe servers. Depending on your installation, one or all of these tasks might already have been completed (you must verify this with the systems programmer responsible for CICS at your site; see the CICS Adapters Administrator's Guide for more details of these tasks):

- Check if the latest CICS Language Environment (LE) support is installed in your CICS region. See the IBM publication *Language Environment for OS/390 Customization* for details on installing LE support in CICS.
- Check if support for the C++ standard classes is explicitly defined to CICS. See the IBM publication *OS/390 C/C++ Programming Guide* for details of the steps required to run C++ application programs under CICS.

A sample job is provided in `orbixhlq.JCLLIB(ORBIXCSD)` to run DFHCSDUP (which is the CICS offline resource definition utility) to define the CICS resources used by the sample jobs and demonstrations. You can run this job, or just use it as a reference when defining the resources online with the CEDA transaction.

When the resources have been defined, use CEDA to install the whole group. If you decide to run the job, first change the JCL to reflect the proper CICS high-level qualifier in use at your site.

Step 5—Customize CICS JCL

Follow these steps to customize the CICS JCL:

1. Add the following load libraries to the DFHRPL concatenation in the CICS region, as follows:

```
DD DSN=orbixhlq.MFA.LOADLIB,DISP=SHR
DD DSN=orbixhlq.DEMO.CICS.CBL.LOADLIB,DISP=SHR
DD DSN=orbixhlq.DEMO.CICS.PLI.LOADLIB,DISP=SHR
```

   If you build your CICS programs with the binder, add the following libraries to the DFHRPL concatenation in the CICS region instead:

```
DD DSN=orbixhlq.MFA.BD.LOADLIB,DISP=SHR
DD DSN=orbixhlq.DEMO.CICS.CBL.BD.LOADLIB,DISP=SHR
DD DSN=orbixhlq.DEMO.CICS.PLI.BD.LOADLIB,DISP=SHR
```

2. Check if the `CEE.SCEERUN` and `CBC.SCLBDLL` libraries are already in the DFHRPL concatenation for the CICS region. If not, add them as follows:

```
DD DSN=CEE.SCEERUN,DISP=SHR
DD DSN=CBC.SCLBDLL,DISP=SHR
```

3. Check if the `CEE.SCEERUN` library is already in the STEPLIB concatenation for the CICS region. If not, add it as follows:

```
DD DSN=CEE.SCEERUN,DISP=SHR
```

4. Check if `CEEMSG` and `CEEOUT` entries are already defined in the JCL for the CICS region. If not, they should be added as follows, to ensure you receive all output from your CICS servers:

```
CEEMSG DD SYSOUT=*
CEEOUT DD SYSOUT=*
```

You must recycle CICS to pick up these changes.

Step 6—CICS security

The CICS server adapter uses standard CICS security mechanisms to communicate with the CICS regions. See the CICS Adapters Administrator's Guide for a detailed description of security considerations involved in using the adapter, and a review of general Orbix and CICS security implications.

To use the CICS server adapter with a secured CICS region, a number of RACF definitions must be added or changed. The following are some examples of RACF commands that are needed to establish the necessary permissions. Depending on what security options are enabled in your CICS region, or if the region uses `SECPRFX=YES`, or if you use group instead of member RACF classes, the commands for your region might differ.

The CICS server adapter requires access to the EXCI connection, the CICS region, and the EXCI mirror transaction (the names of which are all specified as arguments to the server adapter when it starts). The following is an example of the commands for the default mode:

```
RDEFINE FACILITY (DFHAPPL.ORXPIPE1) UACC(NONE)
PERMIT DFHAPPL.ORXPIPE1 CLASS(FACILITY) ID(server)
   ACCESS(UPDATE)

RDEFINE FACILITY (DFHAPPL.CICS) UACC(NONE)
PERMIT DFHAPPL.CICS CLASS(FACILITY) ID(server) ACCESS(READ)

REDEFINE TCICSTRN ORX1 UACC(NONE)
PERMIT ORX1 CLASS(TCICSTRN) ID(server) ACCESS(READ)
```

With CICS TS, the default setting of the `SURROGCHK` parameter in the DFHXCOPT options table has changed from `NO` to `YES`. To avoid a `423` error from EXCI, set `SURROGHCK=NO` in the DFHXCOPT options table or give the client user ID's READ authority to a profile named `userid.DFHEXCI` in the RACF SURROGAT general resource class. See the chapter on security in the IBM publication *CICS External Interfaces Guide, SC33-1944* for more details of how to do this.

Step 7—Verify adapter configuration prerequisites

Verify that the configuration variables in the `cicsa` scope of your configuration file have been changed to match those specified in the CICS control region that you are connecting to. In particular, ensure that you have specified the location of the adapter mapping member that is to be used. For details of how to do this, and the defaults used when the entries are not specified using configuration, see the CICS Adapters Administrator's Guide.

Step 8—Run the CICS server adapter in prepare mode

Before proceeding with this step ensure that the locator daemon and node daemon are all running. Also ensure that the relevant CICS region is active.

If you want to use the CICS server adapter, you must run it first in prepare mode. Submit the JCL in *orbixhlq*.JCLLIB(PREPCICA) to run the CICS server adapter in prepare mode.

Running the CICS server adapter in prepare mode generates a stringified IOR for it and writes this IOR to *orbixhlq*.CONFIG(IORCICSA). The IT_MFA IOR is automatically added to the configuration file by the prepare step.

If the CICS server adapter is configured for EXCI communications, you can generate an IOR for cicsraw by running step ITCFG2 in the JCL.

If the CICS server adapter is configured for APPC communications, you should comment out step ITCFG2 in the JCL, as APPC does not support cicsraw.

Step 9—Run the CICS server adapter in run mode

You are now ready to start the CICS server adapter. Submit the JCL in *orbixhlq*.JCLLIB(CICSA) to run the CICS server adapter.

Client Adapter Customization

Overview

This section is only relevant if you want to use the IMS/CICS client adapter component of Orbix Mainframe. It describes the customization tasks to be performed before using the client adapter.

Note: If you need to perform the tasks in this section, perform them in the order in which they are presented. Before you proceed ensure that the tasks in "Naming Service and IFR Customization" on page 51 have already been completed, if you intend to use the IFR as the type repository for the IMS/CICS client adapter.

Step 1—Avoid known problems

It is recommended that the PTFs listed in "System Requirements" on page 9 are applied, to avoid known problems.

Step 2—Cross memory runtime license

The cross memory transport runtime is a separately licensed component. If you plan to use cross memory communication with the client adapter, please refer to "Installing an Optional License Key" on page 74 for details.

Step 3—Configure APPC or cross memory communication for IMS

If you plan to use the IMS client adapter with APPC, you must enable APPC communication for IMS.

For details of how to configure APPC for IMS see the IBM publication *MVS Planning: APPC/MVS Management, GC28-1807*. Additionally, for specific details on the use of APPC by IMS, see the chapter on administration of APPC/IMS and LU devices in the IBM publication *IMS/ESA Administration Guide: Transaction Manager, SC26-8104*.

Alternatively, if you plan to use the IMS client adapter with cross memory communication, the client adapter must be APF-authorized, and the client adapter must run in a non-swappable address space. See the *IMS Adapters Administrator's Guide* for details on performing these tasks.

Step 4—Configure APPC or cross memory communication for CICS

If you plan to use the CICS client adapter with APPC, you must enable APPC communication for CICS.

For details of how to configure APPC for CICS see the IBM publication *MVS Planning: APPC/MVS Management, GC28-1807*. Additionally, for specific details on the use of APPC by CICS, see the chapter on defining APPC links in the IBM publication *CICS Intercommunication Guide, SC33-1695*.

Alternatively, if you plan to use the CICS client adapter with cross memory communication, the client adapter must be APF-authorized, and the client adapter must run in a non-swappable address space. See the *CICS Adapters Administrator's Guide* for details on performing these tasks.

Step 5—Define client adapter APPC/MVS side information

If you plan to use the client adapter with APPC, you need to define a symbolic destination name in the APPC/MVS side information data set. Although JCL is not provided to do this in your product installation, the IMS Adapters Administrator's Guide provides an example of how to do this using a symbolic destination name of `ORXCLNT1`.

Step 6—Verify client adapter configuration

Verify client adapter configuration with APPC

Follow these steps to verify client adapter configuration:

1. Verify that the configuration variables in the `ims_client` and `cics_client` scopes of your configuration member are valid for your installation. In particular, verify that the following configuration variable matches the client adapter APPC/MVS Side Information DESTNAME you specified in "Step 5—Define client adapter APPC/MVS side information" on page 60. For example:

   ```
   plugins:amtp_appc:symbolic_destination = "ORXCLNT1";
   ```

   For details of how to change configuration, and the defaults used when the entries are not specified in configuration, see the IMS Adapters Administrator's Guide.

2. Review the following client configuration parameters shipped in *orbixhlq*.JCLLIB(MFACLINK), and make any changes that are required:

   LOGLVL — Determines the level of event logging that is enabled. Valid values are numbers in the range 0–6:

   - 0—no logging is performed (LOG_NONE)
   - 1—log errors only (LOG_ERROR)
   - 2—log warnings and errors (LOG_WARNING)
   - 3—log high priority informational messages, warnings and errors (LOG_INFO_HIGH)
   - 4—log medium and high priority informational messages, warnings and errors (LOG_INFO_MED)
   - 5—log low, medium and high priority informational messages, warnings and errors (LOG_INFO_LOW)
   - 6—log all messages (LOG_INFO_ALL)

   MAXSEG — The Orbix runtime in CICS/IMS builds up APPC segments of this size. For APPC, multiple segments of this size are used to transmit data. The specified value must be a multiple of 8. The minimum allowed value is 32 bytes. The maximum allowed value is 32760. The default is 32760.

   TIMEOUT — Applies to IMS only. The value specified determines the length of time (in minutes) that the Orbix runtime in IMS allows an APPC receive call to wait to receive data from the client adapter, before it is timed out. The specified value must be in the range 0–1440. A value of 0 means no timeout. The default is 5 minutes.

   SYMBDST — The value specified must match the value in the client adapter APPC/MVS Side Information DESTNAME you specified in "Step 5—Define client adapter APPC/MVS side information" on page 60.

   LOCALLU — Applies to IMS only. The APPC LU IMS uses to communicate with the client adapter. The default is IMSLU01.

If you need to change any of the shipped values, you must assemble and relink the new configuration into *orbixhlq*.MFA.LOADLIB(ORXMFAC1). Edit the JCL in *orbixhlq*.JCLLIB(MFACLINK) to change the default high-level qualifier, so that it reflects the proper value for your installation and then submit the JCL.

Verify client adapter configuration with cross memory communication

Follow these steps to verify client adapter configuration:

1. Verify that the configuration variables in the `ims_client.cross_memory` and `cics_client.cross_memory` scopes of your configuration member are valid for your installation. In particular, verify that the following configuration variable matches the SYMBDST client configuration parameter defined *orbixhlq*.JCLLIB(MFACLINK). For example:

   ```
   plugins:amtp_xmem:symbolic_destination = "ORXCLNT1";
   ```

   For details of how to change configuration, and the defaults used when entries are not specified in configuration, see the *CICS Adapters Administrator's* Guide or the *IMS Adapters Administrator's* Guide.
2. Review the following client configuration parameters shipped in *orbixhlq*.JCLLIB(MFACLINK), and make any changes that are required:

   `LOGLVL` Determines the level of event logging that is enabled. Valid values are numbers in the range `0–6`:

   - `0`—no logging is performed (`LOG_NONE`)
   - `1`—log errors only (`LOG_ERROR`)
   - `2`—log warnings and errors (`LOG_WARNING`)
   - `3`—log high priority informational messages, warnings and errors (`LOG_INFO_HIGH`)
   - `4`—log medium and high priority informational messages, warnings and errors (`LOG_INFO_MED`)
   - `5`—log low, medium and high priority informational messages, warnings and errors (`LOG_INFO_LOW`)
   - `6`—log all messages (`LOG_INFO_ALL`)

MAXSEG	The Orbix runtime in CICS/IMS builds up buffers of this size. If the data being transported is greater than this size, multiple buffers of this size are used to transmit data. The specified value must be a multiple of 8. The minimum allowed value is 64 bytes. The maximum allowed value is 32760. The default is 32760.
TIMEOUT	The Orbix runtime in CICS/IMS uses the PROGRAM CALL (PC) assembler instruction to invoke a PC routine to move data between CICS/IMS and the client adapter. Three PC calls are made when processing a client invocation (send data, receive reply buffer count, and receive reply). The timeout value governs how long it takes to make these three PC calls. If the three calls cannot be made within the configured timeout value, a COMM_FAILURE exception is raised. The timeout value is specified in seconds. (This differs from APPC where the timeout is specified in minutes.) The default is 5 seconds.
SYMBDST	The value specified must match the value in the client adapter configuration item: plugins:amtp_xmem:symbolic_destination
LOCALLU	This value must be set to IT_XMEM. This setting causes the Orbix runtime in CICS/IMS to use cross memory communication for interacting with the client adapter.

Step 7—Customize IMS JCL

To use the client adapter with IMS, add the following libraries to the IMS message region's STEPLIB concatenation:

```
DD DSN=orbixhlq.MFA.LOADLIB.DISP=SHR
DD DSN=orbixhlq.DEMO.IMS.CBL.LOADLIB,DISP=SHR
DD DSN=orbixhlq.DEMO.IMS.PLI.LOADLIB,DISP=SHR
```

If you build your IMS programs with the binder, add the following libraries to the IMS message region's STEPLIB concatenation instead:

```
DD DSN=orbixhlq.MFA.BD.LOADLIB,DISP=SHR
DD DSN=orbixhlq.DEMO.IMS.CBL.BD.LOADLIB,DISP=SHR
DD DSN=orbixhlq.DEMO.IMS.PLI.BD.LOADLIB,DISP=SHR
```

Check if the following entries are already defined in the IMS message region's JCL. If not, they should be added, to ensure that you receive all output from your IMS clients (recycle the message regions to pick up these libraries):

```
SYSPRINT DD SYSOUT=*
CEEDUMP DD SYSOUT=*
CEEOUT DD SYSOUT=*
SYSOUT DD SYSOUT=*
```

Check if the `CEE.SCEERUN` library is already in the STEPLIB concatenation for the CICS region. If not, add it as follows:

```
DD DSN=CEE.SCEERUN,DISP=SHR
```

Step 8—Customize CICS JCL

To use the client adapter with CICS, add the following libraries to the CICS region's DFHRPL concatenation, as follows:

```
DD DSN=orbixhlq.MFA.LOADLIB,DISP=SHR
DD DSN=orbixhlq.DEMO.CICS.CBL.LOADLIB,DISP=SHR
DD DSN=orbixhlq.DEMO.CICS.PLI.LOADLIB,DISP=SHR
```

If you build your CICS programs with the binder, add the following libraries to the DFHRPL concatenation in the CICS region instead:

```
DD DSN=orbixhlq.MFA.BD.LOADLIB,DISP=SHR
DD DSN=orbixhlq.DEMO.CICS.CBL.BD.LOADLIB,DISP=SHR
DD DSN=orbixhlq.DEMO.CICS.PLI.BD.LOADLIB,DISP=SHR
```

Check if the `CEE.SCEERUN` and `CBC.SCLBDLL` libraries are already in the DFHRPL concatenation for the CICS region. If not, add them as follows:

```
DD DSN=CEE.SCEERUN,DISP=SHR
DD DSN=CBC.SCLBDLL,DISP=SHR
```

Step 9—Define required resources to CICS

Before you can run Orbix Mainframe CICS applications in your CICS region, you must perform a number of additional steps to enable CICS to support Orbix Mainframe clients. Depending on your installation, one or all of these tasks might already have been completed. (You must verify with the systems programmer responsible for CICS at your site.) See the CICS Adapters Administrator's Guide for more details of these tasks:

- Check if the latest CICS Language Environment (LE) support is installed in your CICS region. See the IBM publication *Language Environment for OS/390 Customization* for details on installing LE support in CICS.
- Check if support for the C++ standard classes is explicitly defined to CICS. See the IBM publication *OS/390 C/C++ Programming Guide* for details of the steps required to run C++ application programs under CICS.
- A sample job is provided in *orbixhlq*.JCLLIB(ORBIXCSD) to run DFHCSDUP (the CICS offline resource definition utility) to define the CICS resources used by the sample jobs and demonstrations. You can run this job, or just use it as a reference when defining the resources online with the CEDA transaction. When the resources have been defined, use CEDA to install the whole group. If you decide to run the job, first change the JCL to reflect the proper CICS high-level qualifier in use at your site.

Step 10—Start the client adapter

You are now ready to start the client adapter:

1. Review the JCL in *orbixhlq*.JCLLIB(IMSCA) or *orbixhlq*.JCLLIB(CICSCA).
2. Depending on which transport you want the client adapter to support, do one of the following:
   - If you wish to use the APPC transport, ensure that the PPARM JCL symbolic points to the following configuration scope:

     ```
     PPARM='-ORBname iona_services.cics_client'
     ```

     or

     ```
     PPARM='-ORBname iona_services.ims_client'
     ```

- If you wish to use the cross memory communication transport, ensure that the PPARM JCL symbolic points to the following configuration scope:

  ```
  PPARM='-ORBname iona_services.cics_client.cross_memory'
  ```

  or

  ```
  PPARM='-ORBname iona_services.ims_client.cross_memory'
  ```

3. Submit the relevant JCL to start the client adapter.

RRS OTSTM Customization

Overview

This section is only relevant if you want to use the RRS OTSTM component of Orbix Mainframe. It describes the customization tasks to be performed before using RRS OTSTM.

The RRS OTSTM component of Orbix provides transaction coordination services. This allows the following types of clients to perform two-phase commit processing:

- COBOL and PL/I clients running in CICS
- COBOL and PL/I clients running in IMS
- C++ clients

Step 1—Avoid known problems

It is recommended that the PTFs listed in "System Requirements" on page 9 are applied, to avoid known problems.

Step 2—Ensure Orbix loadlibs are APF-authorized

The RRS OTSTM component must run APF-authorized. All the load libraries in the STEPLIB concatenation of *orbixhlq*.PROCLIB(ORXG) must be APF-authorized. These usually include:

- *orbixhlq*.ADMIN.LOADLIB
- *orbixhlq*.LOADLIB
- *orbixhlq*.LPALIB
- *libprfx*.SCEERUN
- *clbprfx*.SCLBDLL

If you are using TLS, you must ensure that the System SSL load library is also APF-authorized.

The SETPROG command can be used to temporarily APF-authorize a data set. You must have authority to run this command. To APF-authorize the Orbix administration load library, issue a command similar to the following:

```
SETPROG APF,ADD,DSNAME=orbixhlq.ADMIN.LOADLIB,SMS
```

To verify that the load library is APF-authorized, issue the following command:

```
D PROG,APF
```

Your systems programmer can assist you in permanently setting the load libraries as authorized.

Step 3—Prepare to run the RRS OTSTM service

Before proceeding with this step ensure that the locator and node daemon are running.

If you want to use the RRS OTSTM service in Orbix Mainframe, you must first run it in prepare mode. Submit the job provided in *orbixhlq*.JCLLIB(DEPLOY3) to do this.

Running the RRS OTSTM service in prepare mode generates stringified IORs for the service. The DEPLOY3 JCL automatically writes the IORs for the RRS OTSTM service to *orbixhlq*.CONFIG(IOROTSTM). It then copies these IORs into the LOCAL_OTSTM_REFERENCE and LOCAL_OTSTM_ADM_REFERENCE variables in *orbixhlq*.CONFIG(DEFAULT@).

Step 4—Run the RRS OTSTM service in run mode

You are now ready to start the RRS OTSTM service. Submit the job in *orbixhlq*.JCLLIB(OTSTM) to run the service.

Artix Transport Customization

Overview

This section describes the customization tasks to be performed on z/OS before you can use the Orbix Mainframe Artix Transport. It is only relevant if you plan to expose your Orbix application endpoints as Web services.

Note: You should read each step in full before proceeding with it. The text might contain important recommendations or requirements that you should be aware of before proceeding.

Step 1—Runtime license

This feature is a separately licensed component. Please refer to "Installing an Optional License Key" on page 74 for details.

Step 2—Update the Orbix Mainframe configuration file

The *orbixhlq*.CONFIG(ARTIX) configuration file contains the extra configuration variables required to expose your Orbix Mainframe server as a Web service. The *orbixhlq*.CONFIG(ARTIX) configuration file must be included in your Orbix Mainframe configuration file. To do this, edit *orbixhlq*.CONFIG(DEFAULT@) as follows, to uncomment the include statement, as follows:

```
include "//HLQ.ORBIX63.DOMAINS(FILEDOMA)";
include "//HLQ.ORBIX63.CONFIG(ORXINTRL)";
include "//HLQ.ORBIX63.CONFIG(ARTIX)";
```

That is, ensure the hash sign (#) is removed from the start of the `include "//HLQ.ORBIX63.CONFIG(ARTIX)";` line.

Step 3—Create SOAP descriptor files for imsraw/cicsraw

If you plan to use the `imsraw` and/or `cicsraw` proprietary interfaces, you should submit the job in *orbixhlq*.JCLLIB(PREPSOAP) to create the requisite type information for access to these interfaces over SOAP.

Step 4—Running the supplied demonstrations

To ensure that all installation and configuration has been completed successfully so far, see the Getting Started chapter of the Orbix Mainframe Artix Transport User's Guide for details of how to run the supplied batch, CICS and IMS demonstrations.

Configuration Items Set During Customization

Overview

This section provides a summary and recap of the configuration items that are set during the customization tasks already described in this section.

Items set during standard and SSL/TLS customization

Table 2 summarizes the configuration items that are set during the standard customization tasks. See "Step 6—Create a configuration file" on page 40 and "Step 7—Update configuration and prepare to run daemons" on page 42 for more details of how these are set.

Table 2: *Items Set During Standard Customization Tasks*

Configuration Item	Description
LOCAL_HOSTNAME	Fully qualified local hostname.
LOCAL_HFS_ROOT	HFS path to be used by Orbix services for databases and logs.
LOCAL_NODE_DAEMON_PORT	TCP/IP port to be used by the node daemon. (This should be unique.)
LOCAL_TLS_NODE_DAEMON_PORT	TCP/IP port to be used by the node daemon for secure conversations. (This should be unique.)
LOCAL_LOCATOR_PORT	TCP/IP port to be used by the locator. (This should be unique.)
LOCAL_TLS_LOCATOR_PORT	TCP/IP port to be used by the locator for secure conversations. (This should be unique.)
LOCAL_NODE_DAEMON_REFERENCE	Stringified IOR for the node daemon.
LOCAL_LOCATOR_REFERENCE	Stringified IOR for the locator.

Items set during naming service and IFR customization

Table 3 summarizes the additional configuration items that are set if you choose to use the Naming Service and IFR. See "Step 1—Prepare to run the naming service and IFR" on page 51 for more details of how these are set.

Table 3: *Items Set During Naming Service and IFR Customization*

Configuration Item	Description
LOCAL_NAMING_REFERENCE	Stringified IOR for the Naming Service.
LOCAL_IFR_REFERENCE	Stringified IOR for the IFR.

Items set during IMS or CICS server adapter customization

Table 4 summarizes the additional configuration items that are set if you choose to use the IMS or CICS server adapter. Some configuration items must be manually set.

Table 4: *Items Set During IMS or CICS Server Adapter Customization*

Configuration Item	Description
LOCAL_MFA_IMS_REFERENCE	Stringified IOR for the IMS server adapter.
LOCAL_MFA_CICS_REFERENCE	Stringified IOR for the CICS server adapter.
plugins:imsa:iiop:port	TCP/IP port to be used by the IMS server adapter. (This should be unique.) *This is only required if running the adapter in direct persistent mode*. The default is to run it in indirect persistent mode.
plugins:cicsa:iiop:port	TCP/IP port to be used by the CICS server adapter. (This should be unique.) *This is only required if running the adapter in direct persistent mode*. The default is to run it in indirect persistent mode.

Note: Table 4 does not list all the configuration items that the CICS and IMS server adapters require. As stated in "Step 3—Verify adapter configuration prerequisites" on page 52, for full details of all the configuration items that the adapters require see the IMS Adapters Administrator's Guide or CICS Adapters Administrator's Guide.

Items set during client adapter customization

Table 5 summarizes the additional configuration items that is set if you choose to use the IMS/CICS client adapter. See "Step 6—Verify client adapter configuration" on page 60 for more details of how these are set.

Table 5: *Items Set During IMS/CICS Client Adapter Customization*

Configuration Item	Description
`plugins:amtp_appc:symbolic_destination`	Client adapter APPC/MVS-side information DESTNAME.
`plugins:amtp_xmem:symbolic_destination`	Client adapter symbolic destination name used for cross memory communication.

Items set during RRS OTSTM customization

Table 6 summarizes the additional configuration items that are set if you choose to use the RRS OTSTM component.

Table 6: *Items Set During RRS OTSTM Customization*

Configuration Item	Description
`LOCAL_OTSTM_REFERENCE`	Stringified IOR for the RRS OTSTM service.
`LOCAL_OTSTM_ADM_REFERENCE`	Stringified IOR for sending administration commands to the RRS OTSTM service.

Items set during Artix Transport customization

Table 7 shows the additional configuration items that are set if you choose to use the Artix Transport component.

Table 7: *Items Set During Artix Transport Customization*

Configuration Item	Description
`policies:well_known_addressing_policy:http:addr_list`	Specifies the port on which the server is listening for client requests when running in insecure mode.
`policies:well_known_addressing_policy:https:addr_list`	Specifies the port on which the server is listening for client requests when running in secure mode.

Installing an Optional License Key

Overview

Some features of Orbix Mainframe make use of runtime components that are licensed on a separate basis to the core Orbix functionality (see "License codes" on page 8).

Typically, any additional optional licenses are delivered as part of a separate product order. Follow these steps to add any additional license keys to the core Orbix Mainframe license file already installed on z/OS.

Step 1—Set up your license file

The product license attachment that you have received by e-mail needs to be transferred to the mainframe, formatted, and appended to your existing Orbix Mainframe license file before you can use the feature in question. Follow these steps:

1. Preallocate a small data set on the host with the following information:

Space Units	Tracks
PRIMARY	1
SECONDARY	1
RECORD FORMAT	VB
RECORD LENGTH	500 (or greater)
BLOCK SIZE	0

2. Use FTP to transfer the license as a text file into the newly created data set. The following is an example of the FTP command sequence, where the drive letter is `C:` and *XXXX.XXXX* represents the name of the data set you have just allocated:

```
C:
ftp hostname
ftp> asc
ftp> put license.txt 'XXXX.XXXX'
```

3. After the license text file has been copied to z/OS, edit the JCL in *orbixhlq*.JCLLIB(UPDLICEN), as follows:

   In the IN DD statement, replace where it says <*your VB dataset here*> with the name of the data set that contains your license file.

Step 2—Update locale

This is only relevant if you want to run in a locale other than the default locale IBM-1047.

If you want to run in a locale other than the default locale IBM-1047:

1. Use the following command in ISPF to change the value of the TO variable, to make it match the locale codeset in which you want to run (where IBM-*xxx* represents the codeset):

   ```
   C 'IBM-500' 'IBM-xxx' ALL
   ```

   The preceding command lets you simultaneously change all occurrences of the default to make it match your codeset.

2. Uncomment the iconv step as follows:

   ```
   //ICONV    EXEC PROC=ORXICONV,P=&ORBIX..CONFIG,M=NEWLICEN
   ```

   (That is, ensure the asterisk (*) is removed from the start of the line.)

Step 3—Submit UPDLICEN

Submit *orbixhlq*.JCLLIB(UPDLICEN). This job backs up your existing license file, copies the license you transmitted to the mainframe, converts the new license to your local code page if needed, and appends the new license to your existing license file. It splits long lines across records, delimiting them with a backslash in column 72.

CHAPTER 4

Testing the Installation

Orbix Mainframe is installed with a number of demonstration programs that illustrate some features of the product. This section describes how to run the supplied demonstrations to test your installation.

In this chapter

This chapter discusses the following topics:

Before You Begin Testing

Overview

This section points out some important information and prerequisites before you begin testing the installation.

Test prerequisites

Before you run any demonstration, ensure that:

- The locator and node daemon are running.
- The proper high-level qualifier for your installation is reflected in the corresponding demonstration library members.

z/OS readme information

On z/OS, various `README` libraries are supplied for the available demonstrations, as follows:

orbixhlq`.DEMO.CBL.README`	COBOL batch demonstrations
orbixhlq`.DEMO.CICS.CBL.README`	COBOL CICS demonstrations
orbixhlq`.DEMO.IMS.CBL.README`	COBOL IMS demonstration
orbixhlq`.DEMO.PLI.README`	PL/I batch demonstrations
orbixhlq`.DEMO.CICS.PLI.README`	PL/I CICS demonstrations
orbixhlq`.DEMO.IMS.PLI.README`	PL/I IMS demonstrations
orbixhlq`.DEMO.CPP.README`	C++ batch demonstrations
orbixhlq`.DEMO.ARTIX.README`	Artix Transport demonstrations

Each `README` library has a separate member for each demonstration that explains the feature(s) being demonstrated and how to run the programs.

z/OS UNIX System Services readme information

On the optional z/OS UNIX System Services, each demonstration directory contains a `README.txt` file, for C++ developers, that explains what feature of the product is being demonstrated and how to run the programs.

Note for existing customers

If you plan to reuse applications that were originally developed with a previous release of Orbix Mainframe, please review the Mainframe Migration and Upgrade Guide for more details.

For more information

For more details on getting started with the supplied COBOL and PL/I demonstrations see the COBOL Programmer's Guide and Reference and PL/I Programmer's Guide and Reference.

C++ Installation Tests

Overview

This section describes the following:

- "Testing a C++ installation on z/OS" on page 80
- "Testing a C++ installation on z/OS UNIX System Services" on page 81

Note: You must use the ANSI C++ compiler to compile the C++ demonstrations.

Testing a C++ installation on z/OS

To ensure that your Orbix Mainframe installation is fully operational, run the simple demonstration, as follows:

1. Build the client executable by submitting

   ```
   orbixhlq.DEMO.CPP.BLD.JCLLIB(SIMPLECL)
   ```

   This creates the client load module, which is automatically stored in the *orbixhlq*.DEMO.CPP.LOADLIB PDS.
2. Build the server executable by submitting

   ```
   orbixhlq.DEMO.CPP.BLD.JCLLIB(SIMPLESV)
   ```

   This creates the server load module, which is automatically stored in the *orbixhlq*.DEMO.CPP.LOADLIB PDS.
3. Register the server with the locator daemon, by submitting

   ```
   orbixhlq.DEMO.CPP.RUN.JCLLIB(SIMPLERG)
   ```

4. Run the server by submitting

   ```
   orbixhlq.DEMO.CPP.RUN.JCLLIB(SIMPLESV)
   ```

5. Run the client by submitting

   ```
   orbixhlq.DEMO.CPP.RUN.JCLLIB(SIMPLECL)
   ```

The output should look as follows:

```
Initializing ORB
Invoking method
Reading object reference from DD:IORS(SIMPLE)
Done
```

Testing a C++ installation on z/OS UNIX System Services

To ensure that your Orbix Mainframe installation is fully operational on z/OS UNIX System Services, run the simple demonstration, as follows:

1. Set the default configuration domain, as follows:

   ```
   . OrbixInstallDir/etc/bin/default-domain_env.sh
   ```

2. Change to the simple directory:

   ```
   cd OrbixInstallDir/asp/6.3/demos/corba/orb/simple
   ```

3. Build the C++ programs:

   ```
   make -e
   ```

4. Start the server:

   ```
   cd cxx_server
   ./server
   ```

5. Open another command prompt, set the same environment variables as in the other one, and start the client:

   ```
   cd cxx_client
   ./client
   ```

Note: The client should return `Done` and stop. The server must be manually stopped.

COBOL Installation Tests

Overview

This section describes the following:

Checking setting for CBLOPTS L/E runtime option

When running Orbix Mainframe applications, L/E run-time parameters are required to ensure the successful execution of the program. The specification of these parameters might need to be altered for COBOL applications, depending on how the CBLOPTS L/E runtime option has been set on your operating system.

CBLOPTS specifies the format of the parameter string on application invocation when the main program is written in COBOL (that is, whether runtime options or program arguments appear first in the parameter string). The procedures shipped with Orbix Mainframe expect that the default setting for the CBLOPTS runtime option is in use (that is, `CBLOPTS(ON)`). If you have changed the default setting to `CBLOPTS(OFF)`, you must change the supplied JCL in *`orbixhlq`*`.DEMO.CBL.JCLLIB` to execute the `ORXG` procedure instead of the `ORXGCBL` procedure. Check with your systems programmer, if you are not certain which value CBLOPTS is set to.

Testing a COBOL installation on z/OS

To ensure that your Orbix Mainframe installation is fully operational, run the simple demonstration, as follows:

Note: The source code for the demonstration is already supplied in the *`orbixhlq`*`.DEMO.CBL.SRC` PDS, so the options to generate it are disabled in the `SIMPLIDL` JCL, to avoid overwriting the shipped code.

1. Run the Orbix IDL compiler by submitting

```
orbixhlq.DEMO.CBL.BLD.JCLLIB(SIMPLIDL)
```

This takes as input the sample IDL in *orbixhlq*.DEMO.IDL(SIMPLE), and subsequently generates the relevant COBOL copybooks, which are stored in the *orbixhlq*.DEMO.CBL.COPYLIB PDS.

2. Build the server executable by submitting

```
orbixhlq.DEMO.CBL.BLD.JCLLIB(SIMPLESB)
```

This creates the server load module, which is automatically stored in the *orbixhlq*.DEMO.CBL.LOADLIB PDS.

3. Build the client executable by submitting

```
orbixhlq.DEMO.CBL.BLD.JCLLIB(SIMPLECB)
```

This creates the client load module, which is automatically stored in the *orbixhlq*.DEMO.CBL.LOADLIB PDS.

4. Run the server by submitting

```
orbixhlq.DEMO.CBL.RUN.JCLLIB(SIMPLESV)
```

This writes an object reference for the server to *orbixhlq*.DEMO.IOR(SIMPLE)

5. Run the client by submitting

```
orbixhlq.DEMO.CBL.RUN.JCLLIB(SIMPLECL)
```

The output should look as follows:

```
Initializing the ORB
Registering the Interface
Reading object reference from file
Invoking Simple::call_me:IDL:Simple/SimpleObject:1.0
Simple demo complete.
```

Testing a COBOL installation with the IMS server adapter

To ensure that the IMS server adapter component of your Orbix Mainframe installation is fully operational, run the IMS simple server demonstration as follows against the simple batch client:

Note: The IMS server implementation code is already supplied in *orbixhlq*.DEMO.IMS.CBL.SRC(SIMPLES), so the option to generate it is disabled in the SIMPLIDL JCL, to avoid overwriting the shipped code.

1. Run the Orbix IDL compiler by submitting

   ```
   orbixhlq.DEMO.IMS.CBL.BLD.JCLLIB(SIMPLIDL)
   ```

   This takes as input the sample IDL in *orbixhlq*.DEMO.IDL(SIMPLE), and subsequently generates:

   - The relevant COBOL copybooks for the IMS server, which are stored in the *orbixhlq*.DEMO.IMS.CBL.COPYLIB PDS.
   - The source code for the IMS server mainline program, which is stored in *orbixhlq*.DEMO.IMS.CBL.SRC(SIMPLESV).
   - The IMS adapter mapping file, which is stored in the *orbixhlq*.DEMO.IMS.MFAMAP PDS.

2. Build the server executable by submitting

   ```
   orbixhlq.DEMO.IMS.CBL.BLD.JCLLIB(SIMPLESB)
   ```

   This creates the IMS server load module, which is stored in the *orbixhlq*.DEMO.IMS.CBL.LOADLIB PDS.

   If you use the binder to build your IMS programs, build the server executable by submitting:

   ```
   orbixhlq.DEMO.IMS.CBL.BLD.JCLLIB(SIMPBDSB)
   ```

   This creates the IMS server load module, which is stored in the orbixhlq.DEMO.IMS.CBL.BD.LOADLIB PDSE.

3. Define a transaction definition for the server, to allow it to run in IMS. For example, the following transaction definition is already defined for the supplied demonstration:

```
APPLCTN GPSB=SIMPLESV,                          x
      PGMTYPE=(TP,,2),                          x
      SCHDTYP=PARALLEL
TRANSACT CODE=SIMPLESV,
      EDIT=(ULC)                                x
```

4. Provide the server load module to the IMS region that is to run the transaction, by adding *orbixhlq*.DEMO.IMS.CBL.LOADLIB and *orbixhlq*.MFA.LOADLIB to the STEPLIB for that IMS region.

   If you use the binder to build your IMS programs, add *orbixhlq*.DEMO.IMS.CBL.BD.LOADLIB and *orbixhlq*.MFA.BD.LOADLIB to the STEPLIB for that IMS region.

5. Build the client executable by submitting:
   - *orbixhlq*.DEMO.CBL.BLD.JCLLIB(SIMPLIDL) to create the copybooks needed by the client program, from the IDL.
   - *orbixhlq*.DEMO.CBL.BLD.JCLLIB(SIMPLECB) to create the client load module, which is then stored in the *orbixhlq*.DEMO.CBL.LOADLIB PDS.
6. Ensure that the full path to the mapping file that contains the relevant mapping entries is specified in the plugins:imsa:mapping_file configuration item. If you are using the shipped configuration, you should update the MFAMAPS DD statement in the *orbixhlq*.JCLLIB(IMSA) JCL to point to the sample mapping entries in *orbixhlq*.DEMO.IMS.MFAMAP(SIMPLEA).
7. Ensure that the full path to the type information file that contains the sample type information is specified in the plugins:imsa:type_info:source configuration item. If you are using the shipped configuration, you can just update the TYPEINFO DD statement in the *orbixhlq*.JCLLIB(IMSA) JCL to point to the sample type information in *orbixhlq*.DEMO.TYPEINFO.

8. Start the IMS server adapter. See the IMS Adapters Administrator's Guide for details of how to do this, or ask your systems administrator to do it for you.

   Note: IMS must be running, with the server load module and the server transaction definitions available at this stage.

9. Retrieve the IMS server adapter's IOR by submitting

   ```
   orbixhlq.DEMO.IMS.CBL.BLD.JCLLIB(SIMPLIOR)
   ```

   This retrieves the IOR for the `simple` interface and places it in *orbixhlq*`.DEMO.IORS(SIMPLE)`.

10. Run the client by submitting

    ```
    orbixhlq.DEMO.CBL.RUN.JCLLIB(SIMPLECL)
    ```

    The client contacts the IMS server adapter, to get it to run the transaction in IMS. The client subsequently displays that it has completed after it receives a response back from the adapter.

    The client output should appear as follows:

    ```
    Initializing the ORB
    Registering the Interface
    Reading object reference from file
    invoking Simple::call_me:IDL:Simple/SimpleObject:1.0
    Simple demo complete.
    ```

Note: To test a COBOL installation with the CICS server adapter, see "Testing a PL/I installation with the CICS server adapter" on page 94 for guidelines, and simply substitute `PLI` with `CBL`, and substitute `PLINCL` with `COPYLIB`, in the dataset names. Generated source member names and client output are, however, the same as when testing a COBOL installation with the IMS server adapter.

Testing a COBOL installation with the client adapter

To ensure that the client adapter component of your Orbix Mainframe installation is fully operational, run the IMS simple COBOL client demonstration as follows against the simple batch server:

Note: The batch server implementation code is already supplied in *orbixhlq*.DEMO.CBL.SRC(SIMPLES), so the option to generate it is disabled in the SIMPLIDL JCL, to avoid overwriting the shipped code.

1. Run the Orbix IDL compiler by submitting

   ```
   orbixhlq.DEMO.CBL.BLD.JCLLIB(SIMPLIDL)
   ```

   This takes as input the sample IDL in *orbixhlq*.DEMO.IDL(SIMPLE), and subsequently generates:

   - The relevant COBOL copybooks for the batch server, which are stored in the *orbixhlq*.DEMO.CBL.COPYLIB PDS.
   - The source code for the batch server mainline program, which is stored in *orbixhlq*.DEMO.CBL.SRC(SIMPLESV).

2. Build the server executable by submitting

   ```
   orbixhlq.DEMO.CBL.BLD.JCLLIB(SIMPLESB)
   ```

   This creates the batch server load module, which is stored in the *orbixhlq*.DEMO.CBL.LOADLIB PDS.

3. Run the Orbix IDL compiler again by submitting

   ```
   orbixhlq.DEMO.IMS.CBL.BLD.JCLLIB(SIMPLIDL)
   ```

   First you must edit the JCL to change the IDLPARM to be as follows, to ensure that the line IDLPARM='-cobol' is commented out with an asterisk:

   ```
   //     IDLPARM='-cobol:-S:-TIMS -mfa:-tSIMPLESV:-inf'
   //*    IDLPARM='-cobol'
   ```

   This JCL takes as input the sample IDL in *orbixhlq*.DEMO.IDL(SIMPLE), and subsequently generates the relevant COBOL copybooks for the IMS client, which are stored in the *orbixhlq*.DEMO.IMS.CBL.COPYLIB PDS.

4. Build the client executable by submitting

```
orbixhlq.DEMO.IMS.CBL.BLD.JCLLIB(SIMPLECB)
```

   This creates the IMS client load module, which is stored in *orbixhlq*.DEMO.IMS.CBL.LOADLIB(SIMPLECL).

   If you use the binder to build your IMS programs, build the client executable by submitting:

```
orbixhlq.DEMO.IMS.CBL.BLD.JCLLIB(SIMPBDCB)
```

   This creates the IMS client load module, which is stored in the *orbixhlq*.DEMO.IMS.CBL.BD.LOADLIB PDSE.

5. Define a transaction definition for the client, to allow it to run in IMS. For example, the following transaction definition is already defined for the supplied demonstration:

```
APPLCTN GPSB=SIMPLECL,                          x
      PGMTYPE=(TP,,2),                          x
      SCHDTYP=PARALLEL
TRANSACT CODE=SIMPLECL,                         x
      EDIT=(ULC)
```

6. Provide the client load module to the IMS region that is to run the transaction, by adding *orbixhlq*.DEMO.IMS.CBL.LOADLIB to the STEPLIB for that IMS region.

   If you use the binder to build your IMS programs, add *orbixhlq*.DEMO.IMS.CBL.BD.LOADLIB to the STEPLIB for that IMS region.

7. Start the locator and the node daemon on the batch server host by submitting the following:

```
orbixhlq.JCLLIB(LOCATOR)
orbixhlq.JCLLIB(NODEDAEM)
```

8. Start the batch server by submitting

```
orbixhlq.DEMO.CBL.RUN.JCLLIB(SIMPLESV)
```

   This places the IOR for the batch server in `orbixhlq.DEMO.IORS(SIMPLE)`.

9. Enable the IMS client to obtain the batch server's IOR by submitting

```
orbixhlq.DEMO.IMS.CBL.BLD.JCLLIB(UPDTCONF)
```

   This writes a configuration entry to the configuration member to enable the IMS client to contact the batch server.

10. Configure the client adapter. See the IMS Adapters Administrator's Guide for more details.

11. Ensure that the full path to the type information file that contains the sample type information is specified in the `plugins:client_adapter:type_info:source` configuration item. If you are using the shipped configuration, you can just update the `TYPEINFO` DD statement in the `orbixhlq.JCLLIB(IMSCA)` JCL to point to the sample type information in `orbixhlq.DEMO.TYPEINFO`.

12. Run the client adapter by submitting

```
orbixhlq.JCLLIB(IMSCA)
```

13. Run the IMS client by entering the transaction name, `SIMPLECL`, in the relevant IMS region.

Note: To test a CICS COBOL installation with the client adapter, see "Testing a PL/I installation for two-phase commit" on page 99 for guidelines, and simply substitute `PLI` with `CBL`, and substitute `PLINCL` with `COPYLIB`, in the dataset names. Generated source member names and client output are, however, the same as when testing an IMS COBOL installation with the client adapter.

Testing a COBOL installation for two-phase commit

To ensure that two-phase commit is operational for your Orbix Mainframe installation, run the CICS COBOL two-phase commit client demonstration as follows:

Note: Two-phase commit client support is available for C++ batch clients, and for COBOL and PL/I clients that are running in CICS or IMS. Two-phase commit client support is not currently available for COBOL and PL/I batch clients.

1. Build the server executable by submitting

   ```
   orbixhlq.DEMO.CPP.BLD.JCLLIB(DATASV)
   ```

   This:

   - Runs the Orbix C++ IDL compiler on the IDL in *orbixhlq*.DEMO.IDL(DATA).
   - Compiles the generated stub code and C++ server code.
   - Links the C++ server code to generate the server executable in *orbixhlq*.DEMO.CPP.LOADLIB(DATASV).

2. Run the Orbix IDL compiler by submitting

   ```
   orbixhlq.DEMO.CICS.CBL.BLD.JCLLIB(DATAIDL)
   ```

   This takes as input the sample IDL in *orbixhlq*.DEMO.IDL(DATA), and subsequently generates:

   - The relevant COBOL copybooks for the CICS client, which are stored in the *orbixhlq*.DEMO.CICS.CBL.COPYLIB PDS.
   - Typeinfo data which is stored in the *orbixhlq*.DEMO.TYPEINFO(DATAB) PDS.

3. Build the client executable by submitting

   ```
   orbixhlq.DEMO.CICS.CBL.BLD.JCLLIB(DATACB)
   ```

   This creates the CICS client load module, which is stored in *orbixhlq*.DEMO.CICS.CBL.LOADLIB(DATACL).

4. Define a transaction definition for the client, to allow it to run in CICS. See *orbixhlq*.JCLLIB(ORBIXCSD) for an example of the transaction definition for the supplied demonstration.
5. Provide the client load module to the CICS region that is to run the transaction, by adding *orbixhlq*.DEMO.CICS.CBL.LOADLIB to the DFHRPL for that CICS region.
6. Start the locator, node daemon and RRS OTSTM on the batch server host by submitting the following

```
orbixhlq.JCLLIB(LOCATOR)
orbixhlq.JCLLIB(NODEDAEM)
orbixhlq.JCLLIB(OTSTM)
```

7. Start the two batch servers by submitting the following:

```
orbixhlq.DEMO.CPP.RUN.JCLLIB(DATAA)
orbixhlq.DEMO.CPP.RUN.JCLLIB(DATAB)
```

   This places the IOR for each batch server in *orbixhlq*.DEMO.IORS(DATAA) and *orbixhlq*.DEMO.IORS(DATAB) respectively.
8. Enable the CICS client to obtain the batch servers' IORs by submitting

```
orbixhlq.DEMO.CICS.CBL.BLD.JCLLIB(DATAIORS)
```

   This writes configuration entries to the configuration member to enable the CICS client to contact each batch server.
9. Configure the client adapter. See the CICS Adapters Administrator's Guide for more details.

   In particular, for this demonstration, ensure that you define the following in the iona_services.cics_client configuration scope:

```
plugins:amtp_appc:maximum_sync_level  = "2";
initial_references:TransactionFactory:reference =
       "%{LOCAL_OTSTM_REFERENCE}";
```

10. Run the client adapter by submitting *orbixhlq*.JCLLIB(CICSCA).
11. Run the CICS client by entering the transaction name, DATC, in the relevant CICS region.

Note: To test an IMS installation for two-phase commit with the client adapter, see "Testing a PL/I installation for two-phase commit" on page 99 for guidelines, and simply substitute `PLI` with `CBL`, and substitute `PLINCL` with `COPYLIB`, in the dataset names. Generated source member names and client output are, however, the same as when testing a CICS COBOL two-phase commit client.

PL/I Installation Tests

Overview

This section describes the following:

Testing a PL/I installation on z/OS

To ensure that your Orbix Mainframe installation is fully operational, run the simple demonstration, as follows:

Note: The source code for the demonstration is already supplied in the *orbixhlq*.DEMO.PLI.SRC PDS, so the options to generate it are disabled in the SIMPLIDL JCL, to avoid overwriting the shipped code.

1. Run the Orbix IDL compiler by submitting

   ```
   orbixhlq.DEMO.PLI.BLD.JCLLIB(SIMPLIDL)
   ```

   This takes as input the sample IDL in *orbixhlq*.DEMO.IDL(SIMPLE), and subsequently generates the relevant PL/I include members, which are stored in the *orbixhlq*.DEMO.PLI.PLINCL PDS.
2. Build the client executable by submitting

   ```
   orbixhlq.DEMO.PLI.BLD.JCLLIB(SIMPLECB)
   ```

   This creates the client load module, which is automatically stored in the *orbixhlq*.DEMO.PLI.LOADLIB PDS.
3. Build the server executable by submitting

   ```
   orbixhlq.DEMO.PLI.BLD.JCLLIB(SIMPLESB)
   ```

   This creates the server load module, which is automatically stored in the *orbixhlq*.DEMO.PLI.LOADLIB PDS.

4. Run the server by submitting

```
orbixhlq.DEMO.PLI.RUN.JCLLIB(SIMPLESV)
```

This writes an object reference for the server to *orbixhlq*.DEMO.IOR(SIMPLE).

5. Run the client by submitting

```
orbixhlq.DEMO.PLI.RUN.JCLLIB(SIMPLECL)
```

The output should look as follows:

```
simple_persistent demo
==============
Calling operation call_me…
Operation call_me completed (no results to display)

End of the simple_persistent demo
```

Testing a PL/I installation with the CICS server adapter

To ensure that the CICS server adapter component of your Orbix Mainframe installation is fully operational, run the CICS simple demonstration, as follows:

Note: The server implementation code is already supplied in *orbixhlq*.DEMO.CICS.PLI.SRC(SIMPLEI), so the option to generate it is disabled in the SIMPLIDL JCL, to avoid overwriting the shipped code.

1. Run the Orbix IDL compiler by submitting

```
orbixhlq.DEMO.CICS.PLI.BLD.JCLLIB(SIMPLIDL)
```

This takes as input the sample IDL in *orbixhlq*.DEMO.IDL(SIMPLE), and subsequently generates:

- The relevant PL/I include files for the CICS server, which are stored in the *orbixhlq*.DEMO.CICS.PLI.PLINCL PDS.
- The source code for the CICS server mainline program, which is stored in *orbixhlq*.DEMO.CICS.PLI.SRC(SIMPLEV).
- The CICS adapter mapping file, which is stored in the *orbixhlq*.DEMO.CICS.MFAMAP PDS.

2. Build the server executable by submitting

   ```
   orbixhlq.DEMO.CICS.PLI.BLD.JCLLIB(SIMPLESB)
   ```

   This creates the CICS server load module, which is stored in the *orbixhlq*.DEMO.CICS.PLI.LOADLIB PDS

   If you use the binder to build your CICS programs, build the server executable by submitting:

   ```
   orbixhlq.DEMO.CICS.PLI.BLD.JCLLIB(SIMPBDSB)
   ```

   This creates the CICS server load module, which is stored in the *orbixhlq*.DEMO.CICS.PLI.BD.LOADLIB PDSE.

3. Define a transaction definition for the server, to allow it to run in CICS. See *orbixhlq*.JCLLIB(ORBIXCSD) for an example of the transaction definition for the supplied demonstration.

4. Provide the server load module to the CICS region that is to run the transaction, by adding *orbixhlq*.DEMO.CICS.PLI.LOADLIB and *orbixhlq*.MFA.LOADLIB to the DFHRPL for that CICS region.

   If you use the binder to build your CICS programs, add *orbixhlq*.DEMO.CICS.PLI.BD.LOADLIB and *orbixhlq*.MFA.BD.LOADLIB to the DFHRPL for that CICS region.

5. Build the client executable by submitting:

   - *orbixhlq*.DEMO.PLI.BLD.JCLLIB(SIMPLIDL) to create the include files needed by the client program, from the IDL.
   - *orbixhlq*.DEMO.PLI.BLD.JCLLIB(SIMPLECB) to create the client load module, which is then stored in the *orbixhlq*.DEMO.PLI.LOADLIB PDS.

6. Ensure that the full path to the mapping file that contains the relevant mapping entries is specified in the `plugins:cicsa:mapping_file` configuration item. The sample mapping entries are in *orbixhlq*.DEMO.CICS.MFAMAP(SIMPLEA).

7. Start the CICS server adapter. See the CICS Adapters Administrator's Guide for details of how to do this, or ask your systems administrator to do it for you.

   Note: CICS must be running, with the server load module and the server transaction definitions available at this stage.

8. Retrieve the CICS server adapter's IOR by submitting

   ```
   orbixhlq.DEMO.CICS.PLI.BLD.JCLLIB(SIMPLIOR)
   ```

   This retrieves the IOR for the `simple` interface and places it in *orbixhlq*.DEMO.IORS(SIMPLE).

9. Run the client by submitting

   ```
   orbixhlq.DEMO.PLI.RUN.JCLLIB(SIMPLECL)
   ```

   The client contacts the CICS server adapter, to get it to run the transaction in CICS. The client subsequently displays that it has completed after it receives a response back from the adapter.

   The client output should appear as follows:

   ```
   simple persistent demo
   ======================
   Calling operation call_me…
   Operation call_me completed (no results to display)

   End of the simple_persistent demo
   ```

Note: To test a PL/I installation with the IMS server adapter, see "Testing a COBOL installation with the IMS server adapter" on page 84 for guidelines, and simply substitute `CBL` with `PLI`, and substitute `COPYLIB` with `PLINCL`, in the dataset names. Generated source member names and client output are, however, the same as when testing a PL/I installation with the CICS server adapter.

Testing a PL/I installation with the client adapter

To ensure that the client adapter component of your Orbix Mainframe installation is fully operational, run the CICS simple PL/I client demonstration as follows against the simple batch server:

Note: The batch server implementation code is already supplied in *orbixhlq*.DEMO.PLI.SRC(SIMPLEI), so the option to generate it is disabled in the SIMPLIDL JCL, to avoid overwriting the shipped code.

1. Run the Orbix IDL compiler by submitting

   ```
   orbixhlq.DEMO.PLI.BLD.JCLLIB(SIMPLIDL)
   ```

   This takes as input the sample IDL in *orbixhlq*.DEMO.IDL(SIMPLE), and subsequently generates:

   - The relevant PL/I include members for the batch server, which are stored in the *orbixhlq*.DEMO.PLI.PLINCL PDS.
   - The source code for the batch server mainline program, which is stored in *orbixhlq*.DEMO.PLI.SRC(SIMPLEV).

2. Build the server executable by submitting

   ```
   orbixhlq.DEMO.PLI.BLD.JCLLIB(SIMPLESB)
   ```

   This creates the batch server load module, which is stored in the *orbixhlq*.DEMO.PLI.LOADLIB PDS.

3. Run the Orbix IDL compiler again by submitting

   ```
   orbixhlq.DEMO.CICS.PLI.BLD.JCLLIB(SIMPLIDL)
   ```

   First you must edit the JCL to change the IDLPARM to be as follows, to ensure that the line IDLPARM='-pli:-V' is not commented out with an asterisk:

   ```
   //*    IDLPARM='-pli:-TCICS -mfa:-tSIMPLESV'
   //*    IDLPARM='-pli:-TCICS -mfa:-tSMSV'
   //     IDLPARM='-pli:-V'
   ```

This JCL takes as input the sample IDL in *orbixhlq*.DEMO.IDL(SIMPLE), and subsequently generates the relevant PL/I include members for the CICS client, which are stored in the *orbixhlq*.DEMO.CICS.PLI.PLINCL PDS.

4. Build the client executable by submitting:

```
orbixhlq.DEMO.CICS.PLI.BLD.JCLLIB(SIMPLECB)
```

This creates the CICS client load module, which is stored in *orbixhlq*.DEMO.CICS.PLI.LOADLIB(SIMPLECL).

If you use the binder to build your CICS programs, build the client executable by submitting:

```
orbixhlq.DEMO.CICS.PLI.BLD.JCLLIB(SIMPBDCB)
```

This creates the CICS client load module, which is stored in the *orbixhlq*.DEMO.CICS.PLI.BD.LOADLIB PDSE.

5. Define a transaction definition for the client, to allow it to run in CICS. See *orbixhlq*.JCLLIB(ORBIXCSD) for an example of the transaction definition for the supplied demonstration.
6. Provide the client load module to the CICS region that is to run the transaction, by adding *orbixhlq*.DEMO.CICS.PLI.LOADLIB to the DFHRPL for that CICS region.

   If you use the binder to build your CICS programs, add *orbixhlq*.DEMO.CICS.PLI.BD.LOADLIB to the DFHRPL for that CICS region.
7. Start the locator and node daemon on the batch server host, by submitting the following:

```
orbixhlq.JCLLIB(LOCATOR)
orbixhlq.JCLLIB(NODEDAEM)
```

8. Start the batch server by submitting

```
orbixhlq.DEMO.PLI.RUN.JCLLIB(SIMPLESV)
```

This places the IOR for the batch server in *orbixhlq*.DEMO.IORS(SIMPLE).

9. Enable the CICS client to obtain the batch server's IOR by submitting

```
orbixhlq.DEMO.CICS.PLI.BLD.JCLLIB(UPDTCONF)
```

This writes a configuration entry to the configuration member to enable the CICS client to contact the batch server.

10. Configure the client adapter. See the CICS Adapters Administrator's Guide for more details.
11. Run the client adapter by submitting

```
orbixhlq.JCLLIB(CICSCA)
```

12. Run the CICS client by entering the transaction name, `SMCL`, in the relevant CICS region.

Note: To test an IMS PL/I installation with the client adapter, see "Testing a COBOL installation with the client adapter" on page 87 for guidelines, and simply substitute `CBL` with `PLI`, and substitute `COPYLIB` with `PLINCL`, in the dataset names. Generated source member names and client output are, however, the same as when testing a PL/I installation with the CICS server adapter.

Testing a PL/I installation for two-phase commit

To ensure that two-phase commit is operational for your Orbix Mainframe installation, run the IMS PL/I two-phase commit client demonstration as follows:

Note: Two-phase commit client support is available for C++ batch clients, and for COBOL and PL/I clients that are running in CICS or IMS. Two-phase commit client support is not currently available for COBOL and PL/I batch clients.

1. Build the server executable by submitting

```
orbixhlq.DEMO.CPP.BLD.JCLLIB(DATASV)
```

This:

- Runs the Orbix C++ IDL compiler on the IDL in *orbixhlq*.DEMO.IDL(DATA).
- Compiles the generated stub code and C++ server code.

- Links the C++ server code to generate the server executable in *orbixhlq*.DEMO.CPP.LOADLIB(DATASV).

2. Run the Orbix IDL compiler by submitting

```
orbixhlq.DEMO.IMS.PLI.BLD.JCLLIB(DATAIDL)
```

This takes as input the sample IDL in *orbixhlq*.DEMO.IDL(DATA), and subsequently generates:

- The relevant PL/I include members for the IMS client, which are stored in the *orbixhlq*.DEMO.IMS.PLI.COPYLIB PDS.
- Typeinfo data which is stored in the *orbixhlq*.DEMO.TYPEINFO(DATAB) PDS.

3. Build the client executable by submitting

```
orbixhlq.DEMO.IMS.PLI.BLD.JCLLIB(DATACB)
```

This creates the IMS client load module, which is stored in *orbixhlq*.DEMO.IMS.PLI.LOADLIB(DATACL).

4. Define a transaction definition for the client, to allow it to run in IMS. For example, the following transaction is already defined for the supplied demonstration:

```
APPLCTN GPSB=DATACL,                          x
      PGMTYPE=(TP,,2),                        x
      SCHDTYP=PARALLEL                        x
      LANG=PLI
TRANSACT CODE=DATACL,                         x
      EDIT=(ULC)
```

5. Provide the client load module to the IMS region that is to run the transaction, by adding *orbixhlq*.DEMO.IMS.PLI.LOADLIB to the STEPLIB for that IMS region.
6. Start the locator, node daemon, and RRS OTSTM service on the batch server host, by submitting the following:

```
orbixhlq.JCLLIB(LOCATOR)
orbixhlq.JCLLIB(NODEDAEM)
orbixhlq.JCLLIB(OTSTM)
```

7. Start the two batch servers by submitting the following:

```
orbixhlq.DEMO.CPP.RUN.JCLLIB(DATAA)
orbixhlq.DEMO.CPP.RUN.JCLLIB(DATAB)
```

   This places the IOR for each batch server in *orbixhlq*.DEMO.IORS(DATAA) and *orbixhlq*.DEMO.IORS(DATAB) respectively.

8. Enable the IMS client to obtain the batch servers' IORs by submitting

```
orbixhlq.DEMO.IMS.PLI.BLD.JCLLIB(DATAIORS)
```

   This writes configuration entries to the configuration member to enable the IMS client to contact each batch server.

9. Configure the client adapter. See the IMS Adapters Administrator's Guide for more details.

   In particular, for this demonstration, ensure that you define the following in the iona_services.ims_client configuration scope:

```
plugins:amtp_appc:maximum_sync_level  = "2";
initial_references:TransactionFactory:reference =
       "%{LOCAL_OTSTM_REFERENCE}";
```

10. Run the client adapter by submitting *orbixhlq*.JCLLIB(IMSCA).
11. Run the IMS client by entering the transaction name, DATACL, in the relevant IMS region.

Note: To test a CICS installation for two-phase commit with the client adapter, see "Testing a COBOL installation for two-phase commit" on page 90 for guidelines, and simply substitute CBL with PLI, and substitute COPYLIB with PLINCL, in the dataset names. Generated source member names and client output are, however, the same as when testing an IMS PL/I two-phase commit client.

CHAPTER 5

Uninstalling

This chapter describes how to uninstall Orbix Mainframe. It also provides a section on where to find more information about Orbix Mainframe.

In this chapter

This chapter contains the following sections:

Uninstalling Orbix Mainframe

Overview

This section describes how to uninstall Orbix Mainframe, both in a native z/OS and the optional z/OS UNIX System Services environment.

Native z/OS environment

To uninstall Orbix Mainframe in a native z/OS environment, stop all Orbix Mainframe services and delete all files under the high-level-qualifier that you used for this installation.

z/OS UNIX System Services environment

To uninstall Orbix Mainframe in an z/OS UNIX System Services environment, remove all installed files manually.

Finally, remove any references to the *OrbixInstallDir*`/etc/bin/default-domain_env.sh` shell script that you might have in startup scripts, such as `/etc/profile`, or in individual user profiles.

See also the CORBA Administrator's Guide for a full list of environment variables.

For More Information

Release notes	For release-specific information about Orbix Mainframe, see the Mainframe Release Notes.
Knowledge base	Review Knowledge Base articles for Orbix Mainframe at: http://www.iona.com/support/kb/
Technical support	Contact technical support with questions and suggestions at: http://www.progress.com/support