MICRO FOCUS

Orbix 6.3.12

Migrating from ASP 5.1 to Orbix 6.3

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK
http://www.microfocus.com

2020-11-27

Contents

Upgrading from ASP 5.1 to Orbix 6.3

This chapter provides an overview of upgrading from ASP 5.1 to Orbix 6.3, briefly highlighting the main migration issues and providing a summary of the new features in Orbix 6.3.

Main Migration Issues

Because upgrading from ASP 5.1 to Orbix 6.3 is a progression between major releases there are some issues that affect migration.

In this section

This section summarizes the main migration issues. These are:

- Binary incompatibility
- Third-party libraries for HP-UX platforms
- Reorganization of JAR files
- New node daemon

For detailed migration guidance, please consult the other chapters in this document.

Binary incompatibility

Orbix 6.3 is binary incompatible with pre-6.0 releases of the ASP product. Consequently, it is necessary to recompile applications that are upgraded to run in an Orbix 6.3 environment.

To prevent old applications from accidentally loading binary incompatible libraries, the library version number has been incremented in Orbix 6.3. For example, on the Windows platform the old `it_art4_vc60.dll` library in ASP 5.1 is replaced by the `it_art5_vc60.dll` library in Orbix 6.3.

Third-party libraries for HP-UX platforms

In contrast to the initial release of ASP 6.0, which supported only the standard I/O streams on the HP-UX platform (selected by the `-AA` compiler switch), Orbix 6.3 supports both the classic I/O streams library and the standard I/O streams library. Hence, with Orbix 6.3 you should have no difficulty linking with third-party libraries that use one or other of the I/O streams libraries.

For more details, see "I/O Streams on the HP Platform" on page 16.

Reorganization of JAR files

There has been a major reorganization of JAR files in Orbix 6.3, which is due to the adoption of a sophisticated new internal packaging tool, *Xsume*. Consequently, Java developers must edit their build systems to adapt to the new JAR directory structure.

The advantage of the *Xsume* packaging system is that JAR files are organized in a modular hierarchy, facilitating much better maintenance and patching of Orbix Java applications.

New node daemon

The node daemon has been refactored in Orbix 6.3. In some cases, it is now necessary to deploy a node daemon to hosts where, previously, none was required. The advantage of the new node daemon is that it provides more reliable monitoring of the status of server processes.

New Features in Orbix 6.3

This section provides a summary of the new features provided in Orbix 6.3, relative to ASP 5.1.

This section describes the following features:

- New Configuration Tool
- The Orbix Security Framework
- Management
- High Availability through Server Clustering
- Internationalization
- Firewall Proxy Service
- Java New I/O

New Configuration Tool

Orbix 6.3 includes a new configuration tool that simplifies the process of creating a new configuration domain. The new tool automatically imposes constraints to ensure that the selected configuration options are consistent with each other. In addition, the new tool supports extensible configuration. You do not need to configure everything up front. Domain functionality can be extended at a later stage by deploying, for example, a naming service, or by adding or deleting service replicas.

This subsection provides an overview of the tasks that you can perform with the Orbix configuration tool. The following topics are covered:

- Configuration setup tasks
- Runtime management tasks
- More information

Configuration setup tasks

You can use the Orbix configuration tool to perform basic setup tasks such as the following:

- Install or update your license.
- Create a configuration domain.
- Deploy services into a configuration domain.
- Link to existing configuration domains.

- Create server replicas for clustering.
- Add services or replicas to existing configuration domains.

Runtime management tasks

In addition, when you have set up your environment, you can use this tool to perform runtime tasks such as the following:

- Start and stop your Orbix services.
- Open a command prompt configured for your domain.
- Launch the Administrator Web Console.
- Launch other configuration tools (for example, Orbix Configuration Explorer).
- Open other GUI tools for specific Orbix services (for example, Orbix Notification Service Console).

More information

For more information, see the *Orbix Deployment Guide*.

The Orbix Security Framework

The Orbix security framework provides the common underlying security framework for all types of application in Orbix. This subsection provides a high-level overview of the Orbix security framework and its features. It includes the following topics:

Architecture

Figure 1 shows how the Orbix security framework fits into a typical Orbix distributed application, providing a common security infrastructure for every part of the system.

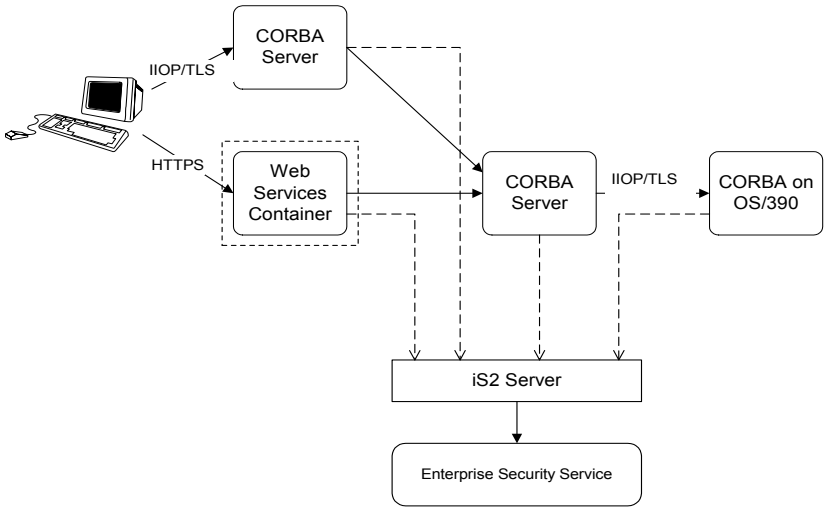

Figure 1 Orbix Security Framework Architecture

Orbix security service

The Orbix security service is the central component of the Orbix security framework. It acts as a repository for security data and supports the following types of service:

- Authentication—either by username and password, by X.509 certificate, smart card or smart token.
- Role-based access control—the Orbix security service can provide a list of realms and roles associated with each user.
- Single sign-on (SSO)—the Orbix security service can be configured to cache security data in a user session the first time a user logs on to the security service.

CORBA security

In addition to the SSL/TLS security, which was available in previous releases, CORBA applications now support the following security features:

- Integration with third-party enterprise security systems via pluggable enterprise security adapters.
- Role-based access control.
- Logging.
- Username/password or token-based authentication/authorization/SSO.
- Identity propagation.
- X.509 certificate-based authentication.

Orbix security service adapters

An *Orbix security service adapter* is a replaceable component of the Orbix security service that integrates the service with a third-party enterprise security service. Micro Focus provides several ready-made adapters that are implemented with the Orbix security service adapter API, including:

- LDAP adapter.
- File adapter.

Orbix security service custom adapters

The Orbix security framework allows you to implement custom adapters that integrate with the enterprise security system of your choice. The *Adapter Software Development Kit* provides the programming interfaces you need to implement your own customized security framework adapter.

Clustered Orbix security service

New in Orbix 6.3, multiple security servers can be deployed to remove any single points of failure through automatic failover to backup servers. Orbix security supports load balancing across security server instances in a security service cluster.

In addition, security servers can be federated so that you only need to sign on once to have access to multiple security domains.

More information

For more information, see the *Orbix Security Guide*.

Management

Orbix 6.3 provides easy to use management and administration tools. System management capabilities in Orbix support test and debug, fine-tuning of applications, and operational control and support. Orbix management facilities provide mechanisms to set thresholds on critical system attributes in order to alert devices, system operators, or other software components of problems without requiring human intervention.

The Orbix 6.3 management framework also provides for integration with the major third-party Enterprise Management Systems (EMS), such as IBM Tivoli, HP OpenView and BMC Patrol. This enables system administrators and production operators to monitor enterprise-critical applications from a single management console.

In this subsection

This subsection provides a high-level overview of the Orbix management and administration tools. It includes the following topics:

- Administrator Web Console
- Administrator Management Service
- Configuration Explorer
- Orbix Configuration Authority

- Performance logging
- EMS integration
- More information

Administrator Web Console

The Administrator Web Console provides a standard web browser interface to explore and manage distributed applications. The Administrator Web Console uses HTML and JavaScript to create a standard explorer view to represent the data.

Administrator Management Service

The Administrator management service is the central point of contact for accessing management information in a domain. The management service acts as a buffer between managed applications and management tools.

Key features provided by the management service are:

- Centralized repository for all management information.
- Centralized collection of event logging information.
- Persistent storage of event log and agent information.
- Load management gateway plug-ins (for example, an SNMP plug-in).
- Capability to terminate server processes.

Configuration Explorer

The Configuration Explorer is an intuitive Java GUI that enables you to view, modify, and search for configuration settings.

Orbix Configuration Authority

The Orbix Configuration Authority displays text descriptions of all Orbix configuration settings. Its web browser interface enables you to navigate to and search for configuration information.

Performance logging

Orbix 6.3 introduces client-side and server-side performance logging. This gives metrics on server availability and response time. It does not require any changes to code. A simple configuration setting is all that is required to set this in action.

Orbix 6.3 also includes a plug-in that monitors managed entities. It gathers statistics on whatever has been instrumented and stores them in the log file. For example, the Orbix work queue has been instrumented and its length can be monitored. In addition, any application-level managed entities can be monitored.

EMS integration

Orbix can be integrated with several Enterprise Management Systems (EMSs). These include BMC Patrol, HP OpenView, and IBM Tivoli. An integrated management system means that fault reports can be organized and correlated so that operators can find the cause of a problem, rather than being swamped by the symptoms.

Having a single management console reduces the learning curve for the operations staff. In addition, an EMS helps by providing the automatic triggering of recovery actions when problems occur. And, an integrated EMS enables service-level agreement compliance to be monitored and the business impact of system problems to be analyzed.

More information

For more information, see the *Orbix Management User's Guide* and the *Orbix Management Programmer's Guide*.

High Availability through Server Clustering

Orbix supports server clustering. It is possible to group together multiple physical servers—each of which can be running on a different machine—into a single logical server. To clients using the server, it appears as a single server process. Orbix, however, distributes invocations across the set of server processes in the cluster.

In this subsection

This subsection gives a high-level overview of the new high availability features in Orbix 6.3. The following topics are covered:

- Berkeley DB replication
- Slave automatically promoted to master
- More information

Berkeley DB replication

In Orbix 6.3, changes have been made at the Berkeley DB level. Berkeley DB has the ability to propagate replication data between different instances of the database. Orbix inherits this ability to replicate and propagates the data across the network through the Persistent State Service layer. For the user this provides a dramatic performance improvement when slaves are being promoted to master. The database does not need to be opened, closed and recovered with each replication update on a slave replica.

Slave automatically promoted to master

If the master fails, a slave is automatically promoted without the need to restart any services or make any changes to configuration. Berkeley DB has an election protocol that guarantees that the most appropriate slave is promoted when the master fails. The most up-to-date slave is always elected first. If all slaves are at the same level, then they are promoted according to a priority setting. If no priorities are assigned, slaves are promoted randomly.

More information

For more information, see the *Orbix Administrator's Guide* and the *Orbix CORBA Programmer's Guides*.

Internationalization

Orbix 6.3 introduces major improvements in its support for internationalization. The following aspects of the ASP product are affected:

- Operating system support.
- CORBA internationalization.

Operating system support

Orbix is now comprehensively tested on the following internationalized operating systems:

- Japanese Windows.
- Solaris 8 in a Japanese locale (`ja` locale).

CORBA internationalization

Orbix features greatly enhanced support for internationalization and codeset negotiation in CORBA applications, including the following new features:

- CORBA code set negotiation has been greatly extended. More than 100 code sets are now supported, including the most popular code sets used in European, Chinese, Japanese and Korean locales.
- Preferred code sets (that is, the native code set and the communication code set) can now be specified for a CORBA application through the ART plug-in, `codeset`.

Firewall Proxy Service

Note: The Firewall Proxy Service is deprecated as of Orbix 6.3.12.

Orbix's firewall proxy service (FPS) addresses a problem that often arises in large organizations, where CORBA applications are required to communicate across internal firewalls within an intranet. Unfortunately, most TCP/IP firewalls do not support IIOP traffic at the protocol proxy level.

The FPS is a firewall proxy that listens on a specified, limited range of IP ports and is capable of routing IIOP messages to CORBA servers behind the firewall. Hence, the FPS can be deployed on a bastion host. Using the FPS eliminates the need to open up a wide range of ports thus avoiding a major security weakness.

This subsection gives a high-level overview of the FPS. The following topics are covered:

- Firewall architecture
- Description
- Configuring CORBA servers to use the FPS
- Incompatibility with SSL/TLS
- More information

Firewall architecture

Figure 2 gives an overview of the FPS architecture.

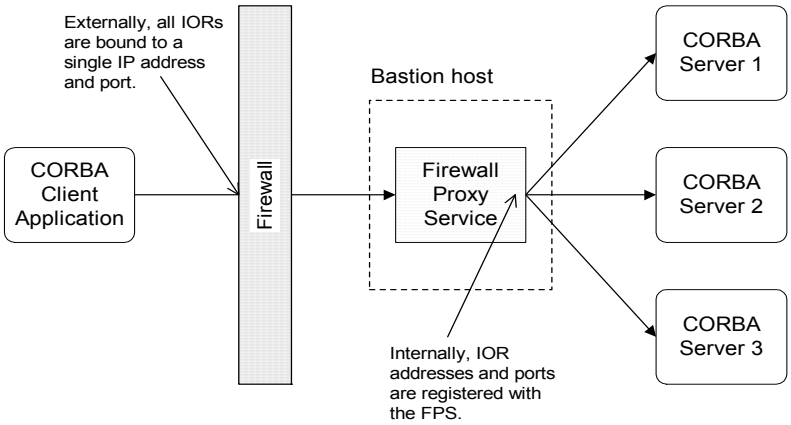

Figure 2 Architecture of the Firewall Proxy Service

Description

The FPS maps interoperable object references (IORs) exposed to the external clients to those of the real CORBA servers. Only Portable Object Adapter (POA) based servers can be accessed through the FPS.

A CORBA server that uses the FPS exchanges IOR template information with the FPS during a registration process that is initiated when a POA is created. Once a server has registered with the FPS, it generates IORs that point clients to proxies managed by FPS. FPS maintains a persistent store of registration information. When the Firewall Proxy Service initializes, it recreates the bindings for any server that registered with the service during a previous execution. This assures that server registration is persistent across many executions of FPS.

Configuring CORBA servers to use the FPS

A CORBA server application can be configured to use the FPS just by adding the `fps` plug-in to its ART plug-ins list. No coding or recompilation of the application is required.

By default, all of the server's incoming requests are then routed through the FPS. If a finer granularity of control is required, however, the firewall routing can be enabled or disabled at the level of individual POA instances by programming an *interdiction policy*.

Incompatibility with SSL/TLS

The FPS supports IIOP traffic only. It is not compatible with IIOP over SSL/TLS.

More information

For more information, see the *Orbix Administrator's Guide*.

Java New I/O

Orbix 6.3 offers support for the Java new I/O API (NIO) through a new implementation of the ATLI2 plug-in (ATLI2 is the ART transport layer plug-in). The existing Java ATLI2 plugin (based on Java classic I/O, or CIO) is still available and remains the default, because Java NIO does not yet support multicast.

This subsection includes the following topics:

NIO features

According to the Java 2 SDK documentation, Java NIO offers the following features:

- Buffers for data of primitive types.
- Character-set encoders and decoders.
- A pattern-matching facility based on Perl-style regular expressions.
- Channels, a new primitive I/O abstraction.
- A file interface that supports locks and memory mapping.
- A multiplexed, non-blocking I/O facility for writing scalable servers.

Prerequisites

The following prerequisites must be satisfied to use NIO in your CORBA Java applications:

- J2SE 1.4.x or higher is required.
- Orbix must be configured to use NIO in the transport layer.

Restrictions

Applications that use EGMIOP must continue to use classic I/O, as multicast sockets are not supported by Java NIO.

Enabling NIO in Orbix 6.3

To enable Java NIO for a CORBA Java application, modify the `plugins:atli2_ip:ClassName` setting as follows:

```
# Orbix configuration file
plugins:atli2_ip:ClassName =
   "com.iona.corba.atli2.ip.nio.ORBPlugInImpl";
```

The `plugins:atli2_ip:ClassName` configuration variable can have either of the values shown in Table 3.

Figure 3 Java I/O API Selection

Configuration Value	Selected I/O API
`com.iona.corba.atli2.ip.nio.ORBPlugInImpl`	New I/O.
`com.iona.corba.atli2.ip.cio.ORBPlugInImpl`	Classic I/O.

More information

For more information about Java NIO, see the following:

- Java 2 SDK New I/O Documentation `http://docs.oracle.com/cd/E19683-01/806-7930/6jgp65ikc/index.html`.
- ***Orbix 6 Administrator's Guide***

Rebuilding CORBA C++ Applications

This chapter is aimed at C++ developers who want to take a CORBA C++ application developed in ASP 5.1 and migrate it to Orbix 6.3. The discussion focuses on necessary source code modifications and on changes to the application build environment.

Source Code Modifications

This section describes any changes in Orbix 6.3 that might require you to modify your C++ source code when migrating from ASP 5.1 to Orbix 6.3.

Incompatible C++ API Changes

The following area of the C++ API must be modified when migrating a CORBA C++ application from ASP 5.1 to Orbix 6.3:

- C++ Management API.
- Work queue policy ID.

C++ Management API

The C++ API for enabling management in CORBA applications has changed very significantly in Orbix 6.3.

Please consult the *Management Programmer's Guide* for a detailed explanation of how to program the new C++ management API.

Work queue policy ID

The policy ID that identifies the manual work queue policy has changed in Orbix 6.3. That is, the `IT_WorkQueue::WORK_QUEUE_POLICY_ID` policy ID has changed to `IT_PortableServer::DISPATCH_WORKQUEUE_POLICY_ID`.

For example, the ASP 5.1 code for creating a manual work queue policy on the POA would include the following line:

```
// C++  - ASP 5.1
...
policies[0] = global_orb->create_policy(
                  IT_WorkQueue::WORK_QUEUE_POLICY_ID,
                  workQueuePolicy);
...
```

Whereas the Orbix 6.3 code for creating a manual work queue policy would include a line like the following:

```
// C++  - Orbix 6.3
...
policies[0] = global_orb->create_policy(

   IT_PortableServer::DISPATCH_WORKQUEUE_POLICY_ID,
                  workQueuePolicy);
...
```

Makefiles

This section discusses any changes that could have an impact on your makefiles when migrating CORBA C++ applications from ASP 5.1 to Orbix 6.3.

Directory Structure

Renamed directories

The directories needed for building CORBA C++ applications in Orbix 6.3 are arranged similarly to the directories in ASP 5.1. The difference in the directory names results just from the change in version number from 5.1 to 6.3. Table 1 shows how the relevant directories have been renamed (relative to the ASP installation directory, *ASPInstallDir*), going from ASP 5.1 to Orbix 6.3.

Table 1 Directories Needed for Building CORBA C++ applications

ASP 5.1 Directories	Orbix 6.3 Directories
ASPInstallDir/asp/5.1/bin	*OrbixInstallDir*/asp/6.3/bin
ASPInstallDir/asp/5.1/include	*OrbixInstallDir*/asp/6.3/include
ASPInstallDir/asp/5.1/lib	*OrbixInstallDir*/asp/6.3/lib
ASPInstallDir/asp/5.1/idl	*OrbixInstallDir*/asp/6.3/idl

Library Reorganization

This subsection explains how the libraries have been reorganized, going from ASP 5.1 to Orbix 6.3. The following topics are discussed:

- Binary incompatibility.
- Libraries removed in Orbix 6.3.
- Libraries replaced in Orbix 6.3.

Binary incompatibility

Because the Orbix 6.3 release is binary incompatible with the ASP 5.1 release, the version numbers of all the shared libraries (or DLLs in Windows) have been incremented by one.

For example, on the Windows platform the old `it_art4_vc60.dll` library in ASP 5.1 is replaced by the `it_art5_vc60.dll` library in Orbix 6.3.

Libraries removed in Orbix 6.3

Table 2 lists the libraries that have been removed in Orbix 6.3.

Table 2 Libraries Removed in Orbix 6.3.

Removed Library	Impact
`it_logging.lib` (Win), `libit_logging.*` (UNIX)	This library is replaced by the following three libraries: `it_notify_log.lib`, `it_event_log.lib`, `it_basic_log.lib`.
`it_admin.lib` (Win), `libit_admin.*` (UNIX)	*No user impact*.
`it_kdm_server.lib` (Win), `libit_kdm_server.*` (UNIX)	*No user impact*.
`it_location_psk.lib` (Win), `libit_location_psk.*` (UNIX)	*No user impact*.
`it_cxx_ibe.lib` (Win), `libit_cxx_ibe.*` (UNIX)	*No user impact*.
`it_ifr_ibe.lib` (Win), `libit_ifr_ibe.*` (UNIX)	*No user impact*.
`it_kdm_store_pss_r.lib` (Win), `libit_kdm_store_pss_r.*` (UNIX)	*No user impact*.
`it_locator_svr_store_pss_r.lib` (Win), `libit_locator_svr_store_pss_r.*` (UNIX)	*No user impact*.
`it_poa_cxx_ibe.lib` (Win), `libit_poa_cxx_ibe.*` (UNIX)	*No user impact*.
`it_pss_cxx_ibe.lib` (Win), `libit_pss_cxx_ibe.*` (UNIX)	*No user impact*.
`it_pss_r_cxx_ibe.lib` (Win), `libit_pss_r_cxx_ibe.*` (UNIX)	*No user impact*.

Libraries replaced in Orbix 6.3

Table 3 lists the libraries that have been replaced in Orbix 6.3. These libraries have been replaced because the Abstract Transport Layer Interface (ATLI) was refactored for Orbix 6.3.

Table 3 Libraries Replaced in Orbix 6.3.

Old ATLI Libraries from ASP 5.1	New ATLI2 Libraries in Orbix 6.3
`it_atli.lib` (Win), `libit_atli.*` (UNIX)	`it_atli2.lib` (Win), `libit_atli2.*` (UNIX)
`it_atli_iop.lib` (Win), `libit_atli_iop.*` (UNIX)	`it_atli2_iop.lib` (Win), `libit_atli2_iop.*` (UNIX)
`it_atli_tls.lib` (Win), `libit_atli_tls.*` (UNIX)	`it_atli2_tls.lib` (Win), `libit_atli2_tls.*` (UNIX)
`it_tls_atli.lib` (Win), `libit_tls_atli.*` (UNIX)	`it_tls_atli2.lib` (Win), `libit_tls_atli2.*` (UNIX)
`it_atli_tcp_ws.lib` (Win), `libit_atli_tcp_ws.*` (UNIX)	`it_atli2_ip.lib` (Win), `libit_atli2_ip.*` (UNIX)

I/O Streams on the HP Platform

Orbix 6.3 supports both standard I/O streams and classic I/O streams on the HP-UX platform. This contrasts with ASP 6.0, which supported only standard I/O streams.

History of compiler versions and I/O stream support

The history of I/O streams support since the release of ASP 5.1 is described as follows:

- ASP 5.1 compiler versions.
- ASP 6.0 compiler versions.
- Orbix 6.1 compiler versions.

ASP 5.1 compiler versions

ASP 5.1 on the HP platform supports the following C++ compilers:

Table 4 ASP 5.1 Supported Compilers on the HP Platform

Platform	Hardware	Compiler	I/O Streams
HP-PA/HP-UX 11.0	PA-RISC	aC++ A.03.25 (32 bits only)	Classic
		aC++ A.03.31 (-AA 32 and 64 bits)	Standard
HP-PA/HP-UX 11.i	PA-RISC	aC++ A.03.26 (32 bits only)	Classic
		aC++ A.03.31 (-AA 32 and 64 bits)	Standard

ASP 6.0 compiler versions

ASP 6.0, ASP 6.0.1 (service pack 1), and ASP 6.0.2 (service pack 2) on the HP platform support only the following C++ compiler:

Table 5 ASP 6.0 Supported Compilers on the HP Platform

Platform	Hardware	Compiler	I/O Streams
HP-PA/HP-UX 11.0	PA-RISC	aC++ A.03.31 (-AA 32 and 64 bits)	Standard
HP-PA/HP-UX 11.i	PA-RISC	aC++ A.03.31 (-AA 32 and 64 bits)	Standard

Orbix 6.1 compiler versions

Orbix 6.1 and ASP 6.0.3 (service pack 3) on the HP platform support the following C++ compilers:

Table 6 Orbix 6.3 Supported Compilers on the HP Platform

Operating System	Hardware	Compiler	I/O Streams
HP-UX 11.0	PA-RISC	aC++ A.03.31 (no -AA, 32 bits)	Classic
		aC++ A.03.31 (-AA, 32 and 64 bits)	Standard

Table 6 Orbix 6.3 Supported Compilers on the HP Platform

Operating System	Hardware	Compiler	I/O Streams
HP-UX 11.i	PA-RISC	aC++ A.03.31 (no -AA, 32 bits)	Classic
		aC++ A.03.31 (-AA, 32 and 64 bits)	Standard

Orbix 6.3 compiler versions

Orbix 6.3 on the HP platform supports the following C++ compilers:

Table 7 Orbix 6.3 Supported Compilers on the HP Platform

Operating System	Hardware	Compiler	I/O Streams
HP-UX 11.0	PA-RISC	aC++ A.03.55 (no -AA, 32 bits)	Classic
		(-AA, 32 and 64 bits)	Standard
HP-UX 11.i	PA-RISC	aC++ A.03.55. (no -AA, 32 bits)	Classic
		(-AA, 32 and 64 bits)	Standard
HP-UX 11	Itanium	aC++ A.05.5 (-AA)	Standard

-AA compiler switch

The `-AA` C++ compiler flag selects the standard C++ library, which includes the standard version of I/O streams. If you build an application using this flag, any other libraries that link with your application must also be built with this flag.

Using standard I/O streams

By default, Orbix 6.3 is set up to use standard I/O streams on the HP platform (that is, where applications are built using the `-AA` compiler flag).

For example, the `cxx_demo.mk` makefile, which is used by demonstrations in the *OrbixInstallDir*`/asp/6.3/demos` directory, is set up to use standard I/O streams. You can use this as a model for your own makefiles.

Using classic I/O streams

The classic I/O stream libraries and header files are included in a `cios` subdirectory of the Orbix `lib` and `include` directories. Hence, to use classic I/O streams in an Orbix application you should do the following:

1. Modify your source code to include Orbix header files from the *OrbixInstallDir*`/asp/6.3/include/cios` include directory.
2. Modify your makefiles to link with Orbix libraries from the *OrbixInstallDir*`/asp/6.3/lib/cios` library directory.
3. Omit the `-AA` flag from the list of C++ compiler flags.

For example, to compile the Orbix demonstrations with classic I/O streams, you would have to change the `cxx_demo.mk` file in the *OrbixInstallDir*`/asp/6.3/demos` directory to be a link to the `demo_acca0331cios_32.mk` file. This is the same mechanism used to pick up the 64-bit versions rather than the default 32-bit versions of libraries on Solaris and HP.

Rebuilding and Running CORBA Java Applications

This chapter is aimed at Java developers who want to take a CORBA Java application developed in ASP 5.1 and migrate it to Orbix 6.3. The discussion focuses on source code modifications and on changes to the build environment.

Source Code Modifications

This section describes the modifications that you might need to make to the source code of your CORBA Java applications when migrating from ASP 5.1 to Orbix 6.3.

Changes in the IDL-to-Java Mapping

The following changes have been made to the IDL-to-Java mapping in Orbix 6.3, resulting in changes to the stub code that affect CORBA Java applications:

- Wide char/string holder types.

Wide char/string holder types

The IDL-to-Java mapping defines `Holder` types to simulate pass-by-reference semantics for operation parameters. The `Holder` types for wide characters and wide strings have been changed in Orbix 6.3 to be consistent with the OMG IDL-to-Java Language Mapping document (for example, the 01-06-06 Java mapping document).

To migrate Java applications to Orbix 6.3:

1. Replace any instances of `org.omg.CORBA.WcharHolder` by `org.omg.CORBA.CharHolder`.
2. Replace any instances of `org.omg.CORBA.WstringHolder` by `org.omg.CORBA.StringHolder`.

Table 8 shows the IDL-to-Java mapping of all IDL character and string types, comparing ASP 5.1 with Orbix 6.3.

Table 8 Mapping of Character and String Types to Java Holder Types

IDL Data Types	ASP 5.1 Holder Types	Orbix 6.3 Holder Types
`char`	`org.omg.CORBA.CharHolder`	`org.omg.CORBA.CharHolder`
`string`	`org.omg.CORBA.StringHolder`	`org.omg.CORBA.StringHolder`
`wchar`	`org.omg.CORBA.WcharHolder`	`org.omg.CORBA.CharHolder`
`wstring`	`org.omg.CORBA.WstringHolder`	`org.omg.CORBA.StringHolder`

The `CharHolder` type is now used both for ordinary characters and for wide characters. Likewise, the `StringHolder` type is now used both for ordinary strings and wide strings.

Incompatible Java API Changes

The following areas of the Java API must be modified when migrating a CORBA Java application from ASP 5.1 to Orbix 6.3:

- Java management beans.
- Work queue policy ID.

Java management beans

The Java management API, which is used for instrumenting CORBA applications, has changed in Orbix 6.3. To migrate old ASP 5.1 applications, make the following changes:

Step	Action
1	Remove all calls to the `addtoRootMBean()` method and replace them with the `createParentChildRelation()` method (from the `com.iona.management.jmx_iiop.IT_IIOPAdaptorServer` Java interface) instead. The `createParentChildRelation()` method takes the parent and child `MBean`s as parameters. It creates the hierarchical relationships between `MBean`s that are displayed in the navigation tree of the Administrator Web Console.
2	The concept of the Root `MBean` is no longer used. There is a new Process `MBean` instead, which has the same role as the starting point for browsing a server process in the console.
3	Remove all calls to the `removeFromRootMBean()` method. This method is deprecated and no longer needed. When you unregister the `MBean` the parent-child relationships are automatically removed.

For complete details of these changes, see the *Orbix Management Programmer's Guide*.

Work queue policy ID

The policy ID that identifies the manual work queue policy has changed in Orbix 6.3. That is, the `IT_WorkQueue::WORK_QUEUE_POLICY_ID` policy ID has changed to `IT_PortableServer::DISPATCH_WORKQUEUE_POLICY_ID`.

For example, the ASP 5.1 code for creating a manual work queue policy on the POA would include the following line:

```
// Java  - ASP 5.1
import com.iona.corba.IT_WorkQueue.*;
...
policies[0] = orb.create_policy(
                  WORK_QUEUE_POLICY_ID.value,
                  workQueuePolicy);
...
```

Whereas the Orbix 6.3 code for creating a manual work queue policy would include a line such as the following:

```
// Java  - Orbix 6.3
import com.iona.corba.IT_WorkQueue.*;
import com.iona.corba.IT_PortableServer.*;
...
policies[0] = orb.create_policy(
                  DISPATCH_WORKQUEUE_POLICY_ID.value,
                  workQueuePolicy);
...
```

Build-Time Classpaths and JAR Files

This section describes any changes that might affect the build environment for your CORBA Java applications when migrating from ASP 5.1 to Orbix 6.3. In particular, the most important changes are related to the reorganization of JAR files in Orbix 6.3 and the effect this has on the build CLASSPATH.

Organization of JAR Files

New packaging system

Orbix 6.3 uses a new system for combining the components that make up the Orbix product. The new system, which has the internal code name *Xsume*, provides a flexible and scalable system for packaging Orbix.

For the most part, the adoption of the *Xsume* system has little user-visible impact. One area in which changes are visible, however, is the organization of JAR files within Orbix 6.3.

New organization of JAR files

Xsume provides a highly modular approach to packaging and this modularity is reflected in a reorganization of the JAR files in Orbix 6.3. The structure of the two main library directories that contain JAR files can be described as follows:

- *OrbixInstallDir*`/asp/6.3/lib`—holds the entry-point Orbix 6.3 JAR files. These are facade JAR files that can be included on your runtime CLASSPATH.
- *OrbixInstallDir*`/lib`—a directory containing subdirectories, each of which represents a particular module. The JAR files at the bottom of the directory structure are referenced, either directly or indirectly, by the entry-point Orbix 6.3 JAR files.

Facade JAR files

A JAR file that contains no classes of its own and consists of nothing but references to other JAR files is known as a *facade JAR*.

The standard JAR file format defines the mechanism for referencing other JAR files as follows. To reference another JAR file, add the JAR file's pathname to the `Class-Path:` entry in the entry-point JAR's manifest file (using a space character as a delimiter). The referenced JAR file is then implicitly included in the CLASSPATH at runtime.

The JAR files should be referenced using *relative* pathnames only. For more details see:

http://docs.oracle.com/javase/6/docs/technotes/guides/extensions/spec.html#bundled

Example of a facade JAR

As an example of a facade JAR, consider the `asp-corba.jar` file, which is the entry-point JAR file required for running CORBA applications. The `asp-corba.jar` file contains only a manifest file, as follows:
`META-INF/MANIFEST.MF`

The manifest file has the following contents:

```
Manifest-Version: 1.0
Class-Path: ../../../lib/art/art/5.1/art-rt.jar
   ../../../lib/art/omg/5/omg-rt.jar
   ../../../lib/common/classloading/1.1/classloading-rt.jar
   ../../../lib/common/concurrency/1.1/concurrency-rt.jar
   ../../../lib/common/ifc/1.1/ifc-rt.jar
   ../../../lib/common/management/1.1/management-rt.jar
   and so on ... (rest of the file not shown) ...
```

Note: Facade JARs can be nested to arbitrary levels of recursion before reaching the JAR files that actually contain Java classes.

Building a CORBA Java Application

ANT build file

A demonstration `ant` build file is provided in the following location:
OrbixInstallDir`/asp/6.3/demos/corba/demo.xml`

This file defines a set of CLASSPATH IDs, which you can use to construct CLASSPATHs for your own `ant` build systems.

For example, the `basic.classpath` ID lists the basic JAR files needed for compiling a CORBA Java application. The `basic.classpath` ID includes the following JAR files:
OrbixInstallDir`/lib/art/omg/1.3/omg.jar`
OrbixInstallDir`/lib/art/art/1.3/art.jar`

JAR file descriptions

Descriptions of the JAR files that you need to build CORBA Java applications are provided in the following README file:
OrbixInstallDir`/asp/6.3/demos/corba/README_JAVA.txt`

Runtime Classpaths and JAR Files

This section describes any changes that affect the runtime environment for your CORBA Java applications when migrating from ASP 5.1 to Orbix 6.3.

Entry-Point JAR Files

At runtime, you can add entry-point JAR files to your CLASSPATH to get access the classes that your application needs. These entry-point JAR files are facade JAR files, which reference the actual JARs to be loaded.

Entry-point Orbix 6.3 JAR files

Table 9 provides descriptions of the entry-point Orbix 6.3 JAR files, which are located in the *OrbixInstallDir*`/asp/6.3/lib` directory.

Table 9 Descriptions of Entry-Point Orbix 6.3 JAR Files

Entry-Point Orbix 6.3 JAR File	Description
`asp-corba.jar`	Runtime facade JAR file for CORBA Java applications.

Running applications with facade JAR files

To run an application with a facade JAR, simply add the JAR to your CLASSPATH before running the application with the Java interpreter.

For example, if you want to use the classes referenced by the `asp-corba.jar` facade JAR, you would modify your CLASSPATH as follows:

Windows

```
set CLASSPATH=OrbixInstallDir\asp\6.3\lib\asp-corba.jar;%CLASSPATH%
```

UNIX (Bourne shell)

```
CLASSPATH=OrbixInstallDir/asp/6.3/lib/asp-corba.jar:$CLASSPATH
export CLASSPATH
```

Java Endorsed Standards Override Mechanism

The J2SE (formerly JDK) 1.4 runtime provides a new mechanism, the *endorsed standards override mechanism*, for overriding standard interfaces and APIs not under Sun's control.

Overriding standard OMG interfaces and classes

The `java` interpreter uses the endorsed standards override mechanism to specify the standard OMG interfaces and classes that constitute the core CORBA API.

How to use the endorsed standards override mechanism

When running applications using the J2SE 1.4 `java` command, it is recommended that you set the `java.endorsed.dirs` property as follows:

```
java.endorsed.dirs=OrbixInstallDir/lib/art/omg/5
```

The Java runtime environment will use the classes in the endorsed JAR files to override the corresponding classes provided in the Java 2 Platform shipped by Sun.

Setting java.endorsed.dirs on the command line

You can set the `java.endorsed.dirs` property on the command line when running the `java` interpreter. For example:

Windows

```
java -Djava.endorsed.dirs="OrbixInstallDir\\lib\\art\\omg\\1.3" ...
```

UNIX

```
java -Djava.endorsed.dirs=OrbixInstallDir/lib/art/omg/1.3 ...
```

Reference

For more information about the Java endorsed standards override mechanism, see the following URL:

http://docs.oracle.com/javase/8/docs/technotes/guides/standards/index.html

Configuring and Redeploying

This chapter is aimed at system administrators. The differences between ASP 5.1 and Orbix 6.3 that affect application configuration and deployment are highlighted and discussed.

Configuration Domain Deployment

The procedure for deploying an Orbix configuration domain to multiple hosts has changed in Orbix 6.3, as a result of internal reorganization and refactoring of the configuration tools.

Configuration tools

Both `itconfigure` (GUI-based Orbix configuration tool) and `itdeployer` (tool for script-based deployment of configuration domains) have changed considerably for this release of Orbix. These tools can now be used to manipulate a new kind of file, the *configuration deployment descriptor*, that defines the main properties of a configuration domain.

Configuration deployment descriptors

A configuration deployment descriptor, *DomainName*`_dd.xml`, is an XML file generated by the `itconfigure` tool which captures the configuration options selected by the user while running the tool.

Reference

For details about how to use the `itconfigure` and `itdeployer` tools, see the *Orbix Administrator's Guide*.

Advanced deployment requirements

If you have advanced deployment requirements that are beyond the capabilities of the `itconfigure` GUI tool (for example, deploying to a user base numbering in the thousands), we recommend that you contact Micro Focus's SupportLine for further assistance:

https://supportline.microfocus.com/

In particular, our consultants can provide you with migration assistance for advanced system deployment.

New Node Daemon

Orbix 6.3 features a new node daemon, which has been modified to provide more reliable monitoring of server processes. This gives rise to the following migration issues:

- Wider deployment of node daemons.
- Incompatibility with old server binaries.
- Incompatibility of node daemon database.

Wider deployment of node daemons

When upgrading your system to Orbix 6.3, it might be necessary to deploy a node daemon to some hosts where, previously, none was required.

Prior to ASP 6.0, a node daemon was required on a host only if you needed the capability to automatically start (or restart) a CORBA server in response to incoming invocations. Monitoring the state of a server process could be performed by a single central node daemon, which monitored the server through a remote connection.

With Orbix 6.3, a node daemon is required on every machine that hosts servers with persistent POAs (a *persistent POA* is a POA whose `PortableServer::LifespanPolicy` is set to `PERSISTENT`). Monitoring the state of a server process through a local node daemon is more reliable than monitoring by a remote node daemon.

Incompatibility with old server binaries

Because the internal service interfaces for the locator, node daemon, and POA have changed significantly, the new node daemon is incompatible with old (pre-ASP 6.0) server binaries. It is, therefore, necessary to rebuild old application binaries before deploying them to an Orbix 6.3 configuration domain.

Incompatibility of node daemon database

You cannot copy an old node daemon database (usually located in *ASPInstallDir*`/var/`*DomainName*`/dbs/node_daemon`) to a new Orbix 6.3 node daemon database, because the node daemon database schema has changed significantly in Orbix 6.3.