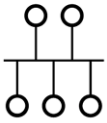

Orbix®

COMet Migration Guide

Version 6.1, December 2003

Making Software Work Together™

Updated: 19-Dec-2003

M 3 1 5 0

Contents

Preface v

Chapter 1 Why Migrate? 1
- **Advantages of Migrating** 2
- **Implications of Migrating** 3

Chapter 2 Usage Models and Interoperability Issues 5
- **Overview of Supported Usage Models** 6
- **COM/Automation Clients of CORBA Servers** 8
  - Orbix 6.x-based COMet Clients of Orbix 6.x Servers 9
  - OrbixCOMet 3 Clients of Orbix 6.x Servers 10
- **CORBA Clients of COM/Automation Servers** 12

Chapter 3 Code Changes and Feature Changes 13
- **Code Changes** 14
- **Feature Changes** 18

Preface

This guide is provided as an addendum to the CORBA Migration and Interoperability Guide. It is aimed at customers who have been using IONA's OrbixCOMet generation 3 product to develop and deploy distributed applications that combine COM and CORBA objects. This guide provides detailed technical guidelines specifically relating to the migration of COMet applications from an Orbix generation 3-based solution to an Orbix 6.*x*-based solution.

Orbix 6.*x* complies with the following specifications:

- CORBA 2.3.
- GIOP 1.2 (default), 1.1, and 1.0.

If you need help with this or any other IONA products, contact IONA at `support@iona.com`. Comments on IONA documentation can be sent to `docs-support@iona.com`.

Audience

This guide is aimed at existing users of the OrbixCOMet generation 3 product who want to migrate their applications to an Orbix 6.*x*-based solution.

Organization of this guide

This guide is divided as follows:

Chapter 1, "Why Migrate?"

This chapter outlines the reasons for and advantages of migrating your COMet applications from an Orbix generation 3-based solution to an Orbix 6.*x*-based solution.

Chapter 2, "Usage Models and Interoperability Issues"

This chapter outlines the various usage models that are supported by Orbix 6.*x*-based COMet. It also describes any interoperability issues associated with each usage model.

Chapter 3, "Code Changes and Feature Changes"

This chapter provides details of the code changes required to COMet clients for successful interoperation with Orbix 6.*x* servers. It also describes OrbixCOMet 3 features that have been deprecated in Orbix 6.*x*-based COMet, because they are either not compatible with or not required in an Orbix 6.*x*-based solution.

Related documentation

The related documentation provided includes:

- CORBA Migration and Interoperability Guide
- COMet Programmer's Guide and Reference

The latest updates to Orbix 6.*x* documentation can be found at http://www.iona.com/support/docs/.

Additional resources

The IONA knowledge base contains helpful articles, written by IONA experts, about the Orbix and other products. You can access the knowledge base at the following location:

http://www.iona.com/support/knowledge_base/

The IONA update center contains the latest releases and patches for IONA products:

http://www.iona.com/support/updates/

Typographical conventions

This guide uses the following typographical conventions:

`Constant width`

Constant width (courier font) in normal text represents portions of code and literal names of items such as classes, functions, variables, and data structures. For example, text might refer to the `CORBA::Object` class.

Constant width paragraphs represent code examples or information a system displays on the screen. For example:

```
#include <stdio.h>
```

Italic	Italic words in normal text represent *emphasis* and *new terms*. Italic words or characters in code and commands represent variable values you must supply, such as arguments to commands or path names for your particular system. For example: `% cd /users/your_name` **Note:** Some command examples may use angle brackets to represent variable values you must supply. This is an older convention that is replaced with *italic* words or characters.

Keying conventions

This guide may use the following keying conventions:

No prompt	When a command's format is the same for multiple platforms, a prompt is not used.
%	A percent sign represents the UNIX command shell prompt for a command that does not require root privileges.
#	A number sign represents the UNIX command shell prompt for a command that requires root privileges.
>	The notation > represents the DOS, Windows NT, Windows 95, or Windows 98 command prompt.
. . . . . .	Horizontal or vertical ellipses in format and syntax descriptions indicate that material has been eliminated to simplify a discussion.
[ ]	Brackets enclose optional items in format and syntax descriptions.
{ }	Braces enclose a list from which you must choose an item in format and syntax descriptions.
|	A vertical bar separates items in a list of choices enclosed in { } (braces) in format and syntax descriptions.

CHAPTER 1

Why Migrate?

This chapter outlines the reasons for and advantages of migrating your COMet applications from an Orbix generation 3-based solution to an Orbix 6.x-based solution.

In this chapter

This chapter discusses the following topics:

Advantages of Migrating	page 2
Implications of Migrating	page 3

Advantages of Migrating

Advantages

Migrating your COM or Automation clients to Orbix 6.*x*-based COMet brings the following advantages:

- Orbix 6.*x*-based COMet results in a much smaller and simpler client deployment than OrbixCOMet 3.
- Orbix 6.*x*-based COMet can use the Orbix 6.*x* Configuration Repository, which means that all clients can access a central configuration repository instead of the need for separate client configurations.
- Orbix 6.*x*-based COMet is more tightly integrated with Orbix 6.*x*.
- Orbix 6.*x*-based COMet is easier to deploy and configure in large-scale projects.

Implications of Migrating

Legacy Support

Although it is possible to continue running your COM or Automation clients on OrbixCOMet 3 instead of migrating them, there are no real advantages in doing this. After upgrading to Orbix 6.*x* on your server side, the task of upgrading your COM or Automation clients to Orbix 6.*x*-based COMet is relatively straightforward and simple.

Note: If you have COMet applications that feature CORBA clients communicating with COM or Automation servers, you can continue to use OrbixCOMet 3 on the client or server side.

Required Code Changes

Certain code changes are required to your client applications to allow them to interoperate successfully with Orbix 6.*x* servers. Such changes are required regardless of whether you choose to migrate your clients to Orbix 6.*x*-based COMet or leave them running on OrbixCOMet 3. In fact, more configuration changes are required if you decide to leave your clients on OrbixCOMet 3. Refer to "OrbixCOMet 3 Clients of Orbix 6.x Servers" on page 10 for details of those further changes.

CHAPTER 2

Usage Models and Interoperability Issues

This chapter outlines the various usage models that are supported by Orbix 6.x-based COMet. It also describes any interoperability issues associated with each usage model.

In this chapter

This chapter discusses the following topics:

Overview of Supported Usage Models	page 6
COM/Automation Clients of CORBA Servers	page 8
CORBA Clients of COM/Automation Servers	page 12

Overview of Supported Usage Models

Supported Usage Models

The following table summarizes and compares the usage models that are supported by OrbixCOMet generation 3 and Orbix 6.*x*-based COMet:

Table 1: *Summary of Supported and Not Supported Usage Models*

Usage Models	OrbixCOMet 3	Orbix 6.*x*-based COMet
Automation client to CORBA server `IDispatch` for late binding Bridge loaded in-process	Supported	Supported
Automation client to CORBA server `IDispatch` for late binding Bridge loaded out-of-process	Supported but not recommended	Supported but not recommended
Automation client to CORBA server Dual interfaces for early binding Bridge loaded in-process	Supported	Supported
Automation client to CORBA server Dual interfaces for early binding Bridge loaded out-of-process	Supported	Not Supported
COM client to CORBA serverBridge loaded in-process	Supported	Supported
COM client to CORBA serverBridge loaded out-of-process	Supported	Not Supported
CORBA client to COM server	Not Supported	Not Supported
CORBA client to Automation server	Supported	Not Supported

Note: Launching the bridge out-of-process dramatically reduces the number of clients that can be handled on certain versions of Windows NT.

The rest of this chapter describes in more detail any issues relating to these usage models. Going from having a bridge launched out-of-process to having a bridge launched in-process requires little or no change in your application code.

COM/Automation Clients of CORBA Servers

Overview

Certain code changes are required to your client applications, regardless of whether you migrate them to Orbix 6.*x*-based COMet or leave them running on OrbixCOMet 3. Refer to "Code Changes and Feature Changes" on page 13 for details of those code changes.

In This Section

The following topics are discussed in this section:

Orbix 6.x-based COMet Clients of Orbix 6.x Servers	page 9
OrbixCOMet 3 Clients of Orbix 6.x Servers	page 10

Orbix 6.x-based COMet Clients of Orbix 6.x Servers

Client Code Changes

No changes are required to your migrated clients, apart from the code changes described in "Code Changes and Feature Changes" on page 13.

Clients and Server States

- Loading the bridge in-process to each client is recommended for Automation clients and required for COM clients.
- Loading the bridge out-of-process is supported for Automation clients that are using `IDispatch` interfaces. However, it is not recommended, because it dramatically reduces the scale in terms of the number of clients that can be handled on certain versions of Windows NT.
- Loading the bridge out-of-process is not supported for Automation clients that are using dual interfaces, or for any COM clients.

OrbixCOMet 3 Clients of Orbix 6.x Servers

Overview

After upgrading to Orbix 6.*x* on your server side, the task of upgrading your COM or Automation clients to Orbix 6.*x*-based COMet is relatively straightforward and simple.

Migration Impact

Although it is possible to continue running your COM or Automation clients on OrbixCOMet 3 instead of migrating them, there are no real advantages in doing this.

If you choose to leave your clients running on OrbixCOMet 3, you must make further configuration changes to your client side on top of the code changes described in "Code Changes" on page 14. Specifically, these further changes relate to the following:

- Interface Repository.
- Configuration Repository.
- Naming Service.

Interface Repository

The OrbixCOMet 3 type store must use the Orbix generation 3 Interface Repository rather than the Orbix 6.*x* Interface Repository. There are three possible solutions to this:

1. Install an Orbix generation 3 Interface Repository on each client machine.
2. Install an Orbix generation 3 Interface Repository on a single intermediary machine for use by all clients.
3. Build the type store cache files and distribute them as part of your OrbixCOMet 3 deployment on each client machine.

Configuration Repository

OrbixCOMet 3 cannot use the Orbix 6.*x* Configuration Repository. The solution to this is to distribute the relevant configuration files as part of your OrbixCOMet 3 deployment on each client machine.

Naming Service

OrbixCOMet 3 clients can use either the Orbix 3 Naming Service or the Orbix 6.*x* Interoperable Naming Service (INS). If you want your clients to use the Orbix 6.*x* INS, you must set the `Common.Services.INS` configuration variable

in the `common.cfg` configuration file so that its value is equal to the IOR string for the root naming context of the Orbix 6.*x* INS. OrbixCOMet 3 then uses this value to locate the INS in any client calls to `GetObject()` that have a parameter in the format *`interface`*`:NAME_SERVICE:`*`naming service compound name`*.

CORBA Clients of COM/Automation Servers

Migration Impact

If you want to have a deployment scenario that features CORBA clients communicating with COM or Automation servers, you can continue to use OrbixCOMet 3. It is relatively straightforward to integrate such a deployment scenario into an Orbix 6.*x*-based system, because OrbixCOMet 3 is fully interoperable with Orbix 6.*x*.

OMG IDL Changes

Some modifications are required to the OMG IDL that is generated in OrbixCOMet 3 by the `ts2idl` utility, to allow the Orbix 6.*x* IDL compiler to use that generated OMG IDL. An IONA Knowledge Base article, 3659.191, is available which describes this in more detail. See the IONA Knowledge Base at `www.iona.com/support/kb/` for more details of this.

CHAPTER 3

Code Changes and Feature Changes

This chapter provides details of the code changes required to COMet clients for successful interoperation with Orbix 6.x servers. It also describes OrbixCOMet 3 features that have been deprecated in Orbix 6.x COMet, because they are either not compatible with or not required in an Orbix 6.x-based solution.

In this chapter

This chapter discusses the following topics:

Code Changes	page 14
Feature Changes	page 18

Code Changes

Overview

Code changes are required in the following areas:

- "Object Location and Binding" on page 14
- "Binding" on page 15
- "Resizing the Object Table" on page 15
- "Client Callbacks" on page 15
- "Timeouts for Remote Calls" on page 15
- "Connection Management" on page 15
- "Handler DLLs" on page 16
- "Configuration Value Handlers" on page 16
- "Output Handlers" on page 16
- "Automation Collections Mapping for CORBA Sequences" on page 16
- "Base Interfaces" on page 16
- "Collocation" on page 17
- "Setting Configuration Values" on page 17
- "Using Transient Port Numbers in Exported IORs" on page 17

Object Location and Binding

Orbix 6.*x*-based COMet clients use the `(D)ICORBAFactory ::GetObject()` method to bind to CORBA objects. However, Orbix 6.*x*-based COMet does not support the following format for the parameter to `GetObject()`:

```
interface:marker:server:host
```

This means that if your client applications bind to CORBA objects by specifying the object marker, server, and host, you must modify those clients so that the parameter to every `GetObject()` call takes the following format instead:

```
interface:TAG:Tag data
```

In the preceding example, *`TAG`* can be either of the following:

- `IOR`

  In this case, *`Tag data`* is the hexadecimal string for the stringified IOR. For example:

  ```
  fact.GetObject("employee:IOR:123456789…")
  ```

- `NAME_SERVICE`

  In this case, *`Tag data`* is the Naming Service compound name separated by ".". For example:

  ```
  fact.GetObject("employee:NAME_SERVICE:IONA.employees.PD.Tom")
  ```

Binding

Orbix 6.*x*-based COMet does not support the `_bind()` function for establishing connections between clients and serves. To this extent, the `(D)IOrbixORBObject::PingDuringBind()` method has been deprecated in Orbix 6.*x*-based COMet. You must update your client applications to remove any calls to `_bind()` and `PingDuringBind()`.

Resizing the Object Table

The `(D)IOrbixORBObject::ReSizeObjectTable()` method has been deprecated in Orbix 6.*x*-based COMet. You must update your client applications to remove any calls to resize the object table.

Client Callbacks

Client callbacks are supported in Orbix 6.*x*-based COMet. However, the `(D)IOrbixORBObject::ReclaimCallbackStore()` method that was available with OrbixCOMet 3 has been deprecated in Orbix 6.*x*-based COMet. You must update your client applications, so that they do not make calls to this method.

Timeouts for Remote Calls

The `(D)IOrbixORBObject::DefaultTxTimeout()` method has been deprecated in Orbix 6.*x*-based COMet. You must update your client applications to remove any calls to this method.

Connection Management

The following API methods in the `(D)IOrbixORBObject` interface have been deprecated in Orbix 6.*x*-based COMet:

- `ConnectionTimeout()`
- `MaxConnectRetries()`
- `NoReconnectOnFailure()`
- `AbortSlowConnects()`
- `CloseChannel()`
- `EagerListeners()`

You must remove any calls to these methods in your client applications.

Handler DLLs

Orbix 6.*x*-based COMet does not support the use of handler DLLs, because they are used to provide a transport for features such as smart proxies and filters, which are not supported in Orbix 6.*x*. To this extent the `(D)IOrbixORBObject::LoadHandler()` method has been deprecated in Orbix 6.*x*-based COMet. You must update your client applications, so that they do not make calls to this method to try to load handler DLLs at application runtime.

Configuration Value Handlers

Configuration value handlers are not supported in Orbix 6.*x*-based solutions. To this extent the following API methods in the `(D)IOrbixORBObject` interface have been deprecated in Orbix 6.*x*-based COMet:

- `ActivateCVHandler()`
- `DeactivateCVHandler()`
- `ReinitialiseConfig()`

Output Handlers

Output handlers are not supported in Orbix 6.*x*-based solutions. To this extent the following API methods in the (D)IOrbixORBObject interface have been deprecated in Orbix 6.*x*-based COMet:

- `ActivateOutputHandler()`
- `DeactivateOutputHandler()`

You must update your client applications to remove any calls to these API methods.

Automation Collections Mapping for CORBA Sequences

In adherence with the COM/CORBA Interworking specification, Orbix 6.*x*-based COMet does not support the mapping of Automation Collections to CORBA sequences. You must update your Automation client applications, so that they use SafeArrays instead of Collections to map to CORBA sequences.

Base Interfaces

The following API methods in the (D)IOrbixORBObject interface have been deprecated in Orbix 6.*x*-based COMet:

- `BaseInterfacesOf()`
- `IsBaseInterfaceOf()`

You must update your client applications to remove any calls to these API methods.

Collocation	The `(D)IOrbixORBObject::Collocated` method has been deprecated in Orbix 6.*x*-based COMet. You must update your client applications to remove any calls to this method.
Setting Configuration Values	The `(D)IOrbixORBObject::SetConfigValue()` method has been deprecated in Orbix 6.*x*-based COMet. You must update your client applications to remove any calls to this method.
Using Transient Port Numbers in Exported IORs	The `(D)IOrbixORBObject::UseTransientPort()` method has been deprecated in Orbix 6.*x*-based COMet. You must update your client applications to remove any calls to this method.

Feature Changes

Overview

This section describes the feature changes between OrbixCOMet 3 and Orbix 6.*x*-based COMet. Some feature changes result from the removal of features made obsolete by Orbix 6.*x*.

Development of CORBA clients and CORBA Servers

Orbix 6.*x*-based COMet does not support the development of either CORBA clients or CORBA servers. The facility to generate Visual Basic or PowerBuilder server stub code has been removed from the Orbix 6.*x*_based COMet set of development support tools.

Surrogate Executable

Because Orbix 6.*x*-based COMet does not support a scenario that features CORBA clients communicating with COM or Automation servers, you cannot use the surrogate executable, `custsur.exe`, in Orbix 6.*x*-based COMet to mimic a CORBA server.

However, if you choose to continue using OrbixCOMet 3, to allow Orbix 6.*x* clients to communicate with COM or Automation servers, you must either:

- Use the `-g` argument with `custsur` to generate IORs for objects belonging to the server that is being implemented by `custsur`.
- Place a reference to `custsur` in the Naming Service.

OMG IDL Types

Orbix 6.*x* supports a larger set of OMG IDL types than Orbix 6.*x*-based COMet. For the purposes of your Orbix 6.*x*-based COMet applications, you must restrict the IDL for your CORBA objects to the set of types supported by Orbix 6.*x*-based COMet. Refer to the *COMet Programmer's Guide and Reference* for details of those IDL types and how they map to both COM and Automation.

New SetOrbName method

In Orbix 6.*x*, there is a COMet method called `SetOrbName()` in the `(D)IOrbixORBObject` interface. This method lets you programmatically specify, in the form load at the start of your applications, the ORB name that you want your COMet applications to use. This means that you can

specify at runtime what configuration information is to be used by your COMet applications. See the *COMet Programmer's Guide and Reference* for more details of this method.

Deprecated APIs

The following APIs are deprecated in Orbix 6.*x*-based COMet:

- `(D)IOrbixServerAPI`
- `DCollection`
- `(D)IOrbixObject`
- `(D)IOrbixSSL`

Deprecated (D)IOrbixORBObject Methods

The following `(D)IOrbixORBObject` methods are deprecated in Orbix 6.*x*-based COMet:

- `AbortSlowConnects()`
- `ActivateCVHandler()`
- `ActivateOutputHandler()`
- `BaseInterfacesOf()`
- `CloseChannel()`
- `Collocated()`
- `ConnectionTimeout()`
- `DeactivateCVHandler()`
- `DeactivateOutputHandler()`
- `DefaultTxTimeout()`
- `EagerListeners()`
- `GetOrbixSSL()`
- `IsBaseInterfaceOf()`
- `LoadHandler()`
- `MaxConnectRetries()`
- `NoReconnectOnFailure()`
- `PingDuringBind()`
- `PlaceCVHandlerAfter()`
- `PlaceCVHandlerBefore()`
- `ReinitialiseConfig()`
- `ReSizeObjectTable()`
- `SetConfigValue()`

- `UseTransientPort()`