



CORBA Tutorial C++

Version 6.3, December 2005

Making Software Work Together™

IONA, IONA Technologies, the IONA logo, Orbix, Orbix/E, Orbacus, Artix, Orchestrator, Mobile Orchestrator, Enterprise Integrator, Adaptive Runtime Technology, Transparent Enterprise Deployment, and Total Business Integration are trademarks or registered trademarks of IONA Technologies PLC and/or its subsidiaries.

Java and J2EE are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

CORBA is a trademark or registered trademark of the Object Management Group, Inc. in the United States and other countries. All other trademarks that appear herein are the property of their respective owners.

While the information in this publication is believed to be accurate, IONA Technologies PLC makes no warranty of any kind to this material including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. IONA Technologies PLC shall not be liable for errors contained herein, or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

#### **COPYRIGHT NOTICE**

No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, photocopying, recording or otherwise, without prior written consent of IONA Technologies PLC. No third party intellectual property right liability is assumed with respect to the use of the information contained herein. IONA Technologies PLC assumes no responsibility for errors or omissions contained in this book. This publication and features described herein are subject to change without notice.

Copyright © 2001–2005 IONA Technologies PLC. All rights reserved.

All products or services mentioned in this manual are covered by the trademarks, service marks, or product names as designated by the companies who market those products.

Updated: 09-Dec-2005

# Contents

Chapter 1 Getting Started with Orbix	1
Creating a Configuration Domain	2
Setting the Orbix Environment	11
Hello World Example	12
Development Using the Client/Server Wizard	14
Development from the Command Line	25
Index	31

CONTENTS

## CHAPTER 1

# Getting Started with Orbix

You can use the CORBA Code Generation Toolkit to develop an Orbix application quickly.

Given a user-defined IDL interface, the toolkit generates the bulk of the client and server application code, including makefiles. You then complete the distributed application by filling in the missing business logic.

This chapter contains the following sections:

Creating a Configuration Domain	page 2
Setting the Orbix Environment	page 11
Hello World Example	page 12
Development Using the Client/Server Wizard	page 14
Development from the Command Line	page 25

In this chapter

# **Creating a Configuration Domain**

Overview	This section describes how to create a simple configuration domain, simple, which is required for running basic demonstrations. This domain deploys a minimal set of Orbix services.
Prerequisites	Before creating a configuration domain, the following prerequisites must be satisfied:
	• Orbix is installed.
	• Some basic system variables are set up (in particular, the
	IT_PRODUCT_DIR, IT_LICENSE_FILE, and PATH variables).
	Fore more details, please consult the Installation Guide.
Licensing	The location of the license file, licenses.txt, is specified by the IT_LICENSE_FILE system variable. If this system variable is not already set in your environment, you can set it now.
Steps	To create a configuration domain, simple, perform the following steps:
	1. Run itconfigure.
	2. Choose the domain type.
	3. Specify service startup options.
	4. Specify security settings.
	5. Specify fault tolerance settings.
	6. Select services.
	7. Confirm choices.
	8. Finish configuration.

#### Run itconfigure

To begin creating a new configuration domain, enter *itconfigure* at a command prompt. An **Orbix Configuration Welcome** dialog box appears, as shown in Figure 1.

Select Create a new domain and click OK.

😽 Orbix Configuration Welcome 🔀
Welcome to the Orbix Configuration tool. Please select an option:
Create a <u>n</u> ew domain
○ <u>O</u> pen an existing domain
○ Go straight into itconfigure
OK Cancel

Figure 1: The Orbix Configuration Welcome Dialog Box

#### Choose the domain type

A Domain Type window appears, as shown in Figure 2. In the Configuration Domain Name text field, type simple. Under Configuration Domain Type, click the Select Services radiobutton. Click Next> to continue.

😽 Create a Configuration Dom	aain - Standard Mode	
Steps	Domain Type	
1. Domain Type	Configuration Identification	
2. Service Startup	You can create many different configuration domains and	
3. Security	access them by their unique name.	
4. Fault Tolerance	What name do you wish to give this configuration domain?	
5. Select Services	Configuration Do <u>m</u> ain Name: simple	
6. Confirm Choices	Configuration Domain Type	
7. Deploying 8. Summary	The configuration tool can create configuration domains with different combinations of Orbix services. Which Orbix services do you want to include in this domain?	
	O All Licensed Services	
	Select Services	
	Storage Location	
	Configuration Directory: c:\Orbix_62\etc	
	Data Directory: c:\Orbix_62\war	
	< <u>Back</u> <u>Next&gt;</u> <u>Finish</u> Cancel	

Figure 2: The Domain Type Window

#### Specify service startup options

A **Domain Type** window appears, as shown in Figure 3. You can leave the settings in this Window at their defaults. Click **Next>** to continue.

×	Create a Configuration Domain - Standard Mode	🖕 Create a Configura
	Steps Service Startup	Steps
	1. Domain Type Startup	1. Domain Type
	2. Service Startup       The services you are configuring can be programmed to run         3. Security       when your computer starts up or manually. All, except for a         4. Fault Tolerance       ininimal set, can start on demand. Do you want         5. Select Services       Image: A minimal set of services launched by a script I can run.         6. Confirm Choices       All selected services launched on machine startup (as system services).         7. Deploying       All selected services launched by a script I can run.         8. Summary       Port         The services need ports to listen for connections. The easiest way to set these port values is to set a base value.         Base Port:       3075	<ol> <li>Service Startup</li> <li>Security</li> <li>Fault Tolerance</li> <li>Select Services</li> <li>Confirm Choices</li> <li>Deploying</li> <li>Summary</li> </ol>
	< <u>Back</u>	

Figure 3: The Service Startup Window

#### Specify security settings

A **Security** window appears, as shown in Figure 4. You can leave the settings in this Window at their defaults (no security). Click **Next**> to continue.

😽 Create a Configuration Dom	ain - Standard Mode
Steps	Security
<ol> <li>Domain Type</li> <li>Service Startup</li> <li>Security</li> <li>Fault Tolerance</li> <li>Select Services</li> <li>Confirm Choices</li> <li>Deploying</li> <li>Summary</li> </ol>	Transports         What communication protocols do you want enabled in the domain?         Insecure Communication (IIOP/HTTP)         Secure and Insecure Communication         Secure Communication (TLS/HTTPS)         Security Features         What security features do you want enabled in the domain?         Expose Services through Firewall         Details         IONA Security Service         Enable Access Control for Core Services
	< <u>Back</u> <u>Pinish</u> Cancel

Figure 4: The Security Window

#### Specify fault tolerance settings

A **Fault Tolerance** window appears, as shown in Figure 5. You can leave the settings in this Window at their defaults. Click **Next>** to continue.

🖶 Create a Configuration Do	main - Standard Mode
Steps	Fault Tolerance
<ol> <li>Domain Type</li> <li>Service Startup</li> <li>Security</li> <li>Fault Tolerance</li> <li>Select Services</li> <li>Confirm Choices</li> <li>Deploying</li> <li>Summary</li> </ol>	Replication         You can run multiple replicas of the core Orbix services to make your system fault tolerant. The service instances on the replica hosts act as backups.         Replication Hosts:         Host       Add         Remove       Edit
	< <u>Back</u> <u>N</u> ext> <u>F</u> inish Cancel

Figure 5: The Fault Tolerance Window

#### Select services

A Select Services window appears, as shown in Figure 6.

In the Select Services window, select the following services and components for inclustion in the configuration domain: Location, Node daemon, Management, CORBA Interface Repository, CORBA Naming, and demos. Click Next> to continue.

😽 Create a Configuration Dom	ain - Standard Mode	×
Steps	Select Services	
<ol> <li>Domain Type</li> <li>Service Startup</li> <li>Security</li> <li>Fault Tolerance</li> <li>Select Services</li> <li>Confirm Choices</li> <li>Deploying</li> <li>Summary</li> </ol>	Infrastructure	Messaging CORBA Notification CORBA Events JMS (Java Messaging) JMS/Notification Bridge Security Firewall Proxy IONA Security Components Demos
	< <u>B</u> ack	Next> Finish Cancel

Figure 6: The Select Services Window

#### **Confirm choices**

You now have the opportunity to review the configuration settings in the **Confirm Choices** window, Figure 7. If necessary, you can use the **<Back** button to make corrections.

Click  $\ensuremath{\textit{Next}}\xspace >$  to create the configuration domain and progress to the next window.

Steps       Confirmation         1. Domain Type       This is your chance to review the choices you have made.         2. Service Startup       To deploy the services on the local host, press Next. To modify any of your choices, press Back. If you don't want to deploy now but wish to save your choices for future use, press Save to store them in a deployment descriptor, then press Cancel.         4. Fault Tolerance       Elocation Service         5. Select Services       Manual Activation         6. Confirm Choices       Manual Activation         7. Deploying       IIOP Port = 3075         8. Summary       Node Daemon Service         Manual Activation       Management Enabled         IIOP Port = 53079       CORBA Interface Repository Service         CORBA Interface Repository Service       Automatic Activation         Management Enabled       IIOP Port = Enabled         IIOP Port = Enabled       IIOP Port = Enabled         IIOP Port = Enabled       IIOP Port = Enabled	😽 Create a Configuration Dom	ain - Standard Mode	×
1. Domain Type       This is your chance to review the choices you have made.         2. Service Startup       To deploy the services on the local host, press Next. To modify any of your choices, press Back. If you don't want to deploy now but wish to save your choices for future use, press Save to store them in a deployment descriptor, then press Cancel.         4. Fault Tolerance       Exercises         5. Select Services       Manual Activation         6. Confirm Choices       Manual Activation         7. Deploying       Bioper Source         8. Summary       Manual Activation         Management Enabled       IIOP Port = 3075         Node Daemon Service       Manual Activation         Management Enabled       IIOP Port = 53079         CORBA Interface Repository Service       Automatic Activation         Management Enabled       IIOP Port = Enabled         IIOP Port = Enabled       IIOP Port = Enabled         IIOP Port = Enabled       IIOP Port = Enabled         IIOP Port = Enabled       IIOP Port = Enabled	Steps	Confirmation	
	Steps 1. Domain Type 2. Service Startup 3. Security 4. Fault Tolerance 5. Select Services 6. Confirm Choices 7. Deploying 8. Summary	Confirmation This is your chance to review the choices you have made. To deploy the services on the local host, press Next. To modify any of your choices, press Bac If you don't want to deploy now but wish to save your choices for future use, press Save to store them in a deployment descriptor, then press Cancel. Location Service Manual Activation Management Enabled IIOP Port = 3075 Node Daemon Service Manual Activation Management Enabled IIOP Port = 53079 CORBA Interface Repository Service Automatic Activation Management Enabled IIOP Port = Enabled	ж.
<pre></pre>		< <u>B</u> ack <u>N</u> ext> <u>F</u> inish Cancel	

Figure 7: The Confirm Choices Window

#### **Finish configuration**

The itconfigure utility now creates and deploys the simple configuration domain, writing files into the *OrbixInstallDir*/etc/bin, *OrbixInstallDir*/etc/domain, *OrbixInstallDir*/etc/log, and *OrbixInstallDir*/var directories.

If the configuration domain is created successfully, you should see a **Summary** window with a message similar to that shown in Figure 8.

Click **Finish** to quit the itconfigure utility.

🐈 Create a Configuration Doma	ain - Standard Mode	×
Steps	Summary	
<ol> <li>Domain Type</li> <li>Service Startup</li> <li>Security</li> <li>Fault Tolerance</li> <li>Select Services</li> <li>Confirm Choices</li> <li>Deploying</li> <li>Summary</li> </ol>	Configuration is now complete, see details below. Configuration completed successfully. You can view the log in 'c:\Orbix_62\var\simple\logs\simple_2004_Nov_23_1_59_6.log'. To set your environment for this configuration domain run: c:\Orbix_62\etc\bin\simple_env.bat To start the services in this configuration domain run: c:\Orbix_62\etc\bin\start_simple_services.bat To stop the services in this configuration domain run: c:\Orbix_62\etc\bin\stop_simple_services.bat	
	< <u>B</u> ack Next> <b>Finish</b> Cance	

Figure 8: Configuration Summary

# **Setting the Orbix Environment**

Prerequisites	Before proceeding with the demonstration in this chapter you need to ensure:
	• The CORBA developer's kit is installed on your host.
	• Orbix is configured to run on your host platform.
	The <i>Administrator's Guide</i> contains more information on Orbix configuration, and details of Orbix command line utilities.
	<b>Note:</b> OS/390, both native and UNIX system services, do not support the code generation toolkit and distributed genies. For information about building applications in a native OS/390 environment, see the readme files and JCL that are supplied in the DEMO data sets of your iPortal OS/390 Server product installation.
Setting the Domain	The scripts that set the Orbix environment are associated with a particular <i>domain</i> , which is the basic unit of Orbix configuration. Consult the <i>Installation Guide</i> , and the <i>Administrator's Guide</i> for further details on configuring your environment.
	To set the Orbix environment associated with the <i>domain-name</i> domain, enter:
	Windows
	> config-dir\etc\bin\domain-name_env.bat
	UNIX
	% . config-dir/etc/bin/domain-name_env
	<i>config-dir</i> is the root directory where the Appliation Server Platform stores its configuration information. You specify this directory while configuring your domain. <i>domain-name</i> is the name of a configuration domain.

## **Hello World Example**

This chapter shows how to create, build, and run a complete client/server demonstration with the help of the CORBA code generation toolkit. The architecture of this example system is shown in Figure 9.



Figure 9: Client makes a single operation call on a server

The client and server applications communicate with each other using the Internet Inter-ORB Protocol (IIOP), which sits on top of TCP/IP. When a client invokes a remote operation, a request message is sent from the client to the server. When the operation returns, a reply message containing its return values is sent back to the client. This completes a single remote CORBA invocation.

All interaction between the client and server is mediated via a set of IDL declarations. The IDL for the Hello World! application is:

```
//IDL
interface Hello {
    string getGreeting();
};
```

The IDL declares a single Hello interface, which exposes a single operation getGreeting(). This declaration provides a language neutral interface to CORBA objects of type Hello.

The concrete implementation of the Hello CORBA object is written in C++ and is provided by the server application. The server could create multiple instances of Hello objects if required. However, the generated code generates only one Hello object.

The client application has to locate the Hello object—it does this by reading a stringified object reference from the file Hello.ref. There is one operation getGreeting() defined on the Hello interface. The client invokes this operation and exits.

# **Development Using the Client/Server Wizard**

Overview	On the Windows NT platform, Orbix provides a wizard add-on to the Microsoft Visual Studio integrated development environment (IDE) that enables you to generate starting point code for CORBA applications. If you are not working on a Windows platform or if you prefer to use a command line approach to development, see "Development from the Command Line" on page 25.
Installing the client/server wizard	You can install the Orbix v6.2 Client/Server wizard into the Microsoft Visual $C++$ 6.0 development environment either automatically or manually.
	Automatic Install
	To install the client/server wizard automatically, use Windows Explorer to navigate to the following directory:
	OrbixInstallDir\asp\6.2\etc\wizard
	Double-click setup.exe in this directory to install the wizard files.
	Manual Install
	To install the client/server wizard manually, copy the following files from the <i>OrbixInstallDir</i> \asp\6.2\etc\wizard directory:
	it_artwiz5_vc60.awx it_artwiz5_vc60.hlp
	Paste these files into the <i>VisualStudioInstallDir</i> \Common\MSDev98\Template directory. The value of <i>VisualStudioInstallDir</i> is usually C:\Program Files\Microsoft Visual Studio.

Prerequisites	You must ensure that the Orbix include and library directories are added to the Microsoft Visual Studio configuration. Start up the Microsoft Visual C++ 6.0 IDE, select <b>Tools   Options</b> from the menu bar, and click on the <b>Directories</b> tab. Use this dialog box to add the following Orbix directories to the Visual Studio configuration: <b>Orbix Include Directory</b>		
	OrbixInstall\asp\6.2\include		
	Orbix Library Directory		
	OrbixInstall\asp\6.2\lib		
Steps to implement Hello World	You implement the Hello World! application with the following steps:		
	1. Define the IDL interface, Hello.		
	2. Generate the server.		
	3. Complete the server program by implementing the single IDL		
	getGreeting() operation.		
	4. Build the server program.		
	5. Generate the client.		
	6. Complete the client program by inserting a line of code to invoke the		
	getGreeting() operation.		
	7. Build the client program.		
	8. Run the demonstration.		
Define the IDL interface	Create the IDL file for the Hello World! application. First of all, make a directory to hold the example code:		
	> mkdir C:\OCGT\HelloExample		
	Create an IDL file C:\OCGT\HelloExample\hello.idl using a text editor.		
	Enter the following text into the hello.idl file:		
	<pre>//IDL interface Hello {     string getGreeting(); };</pre>		

This interface mediates the interaction between the client and the server halves of the distributed application.

#### Generate the server

Generate files for the server application using the CORBA Code Generation Toolkit.

To create a server project using the IONA Orbix client/server wizard:

- 1. Open the Microsoft Visual C++ 6.0 integrated development environment (IDE).
- 2. From the Visual C++ menus, select File | New
- 3. In the New dialog, click on the Projects tab.
- 4. In the Projects tab, perform these actions:
  - Select IONA Orbix v6.2 Client/Server Wizard
  - In the Project name text box, enter server
  - Under the **Location** text box, enter

C:\OCGT\HelloExample\server

New				<u>?</u> ×
Files	Projects	Workspaces	Other Documents	
Clus Clus Clus Cus Dat. Dev Exte	COM AppW ster Resource tom AppWize abase Project Studio Add-i ended Storect A Orbix v6.2	/izard e Type Wizard ard xt in Wizard d Proc Wizard Client/Server Wiz	♥ Win32 Dynamic-Link L ♥ Win32 Static Library	Project game: server Logation: C:\OCGT\HelloExample\server
Mak MFC MFC MFC MFC MFC	PI Extension kefile C ActiveX Co C AppWizard C AppWizard v Database V	Wizard IntrolWizard I (dll) I (exe) Wizard		Create new workspace     Add to current workspace     Dependency of:
¶¶ Utili ∎ Win ■ Win	ty Project 32 Applicatio 32 Console /	on Application		<u>P</u> latforms: ₩Win32
				OK Cancel

5. Click OK.

The client/server wizard dialog displays.

- 6. Answer two questions as follows:
  - What CORBA IDL file would you like to use for this project? Enter the location of hello.idl.
  - Would you like to generate a working client or server?
     Select Server

IONA Orbix v6.2 Client/Ser	ver Wizard - Step 1 of 2	×
IONA	What CORBA IDL file would you like to use for this project?         C:\OCGT\HelloExample\hello.idl         Image: state	
Orbix	Would you like to generate a working client or server? C <u>Client</u> C <u>Server</u>	
	What object reference distribution method would you like to use? Stringified Object References Naming Service	
		_
< <u>B</u> ack <u>N</u> ex	t> <u>F</u> inish Cancel <u>H</u> elp	

7. Advance to the next screen by clicking **Next**.

IONA Orbix v6.2 Client/Serv	ver Wizard - Step 2 of 2
	POA Server Generation Options     Servant Inheritance matches IDL inheritance     Create a <u>multi-threaded server</u> Servant Implementation Approach     (• Inheritance from POA base classes     (• <u>I</u> IE (Delegation)     Servant Management Strategy     (• Create and activate <u>servants in mainline</u> (• ServantActivator creates servants on <u>d</u> emand     (• ServantLocator creates servants per-invocation
	C Use a single default servant for many objects
< <u>B</u> ack <u>N</u> ext	> <u>F</u> inish Cancel <u>H</u> elp

8. The server wizard displays the following dialog:

- 9. Accept the default settings and click **Finish** to generate the server.
- 10. The New Project Information scrollbox tells you about the generated files. Browse the information and select **OK**.
- 11. The server workspace is generated with the following source files:



12. Read the text file ReadmeOrbixServer.txt.

#### Complete the server program

Complete the implementation class, HelloImpl by providing the definition of getGreeting(). This method implements the IDL operation Hello::getGreeting().

Delete the generated boilerplate code that occupies the body of HelloImpl::getGreeting() and replace it with the highlighted line of code:

```
//C++
...
char*
HelloImpl::getGreeting()
{
    char* _result;
    _result = CORBA::string_dup("Hello World!");
    return _result;
}
...
```

The function CORBA::string\_dup() allocates a copy of the string on the free store. This is needed to be consistent with the style of memory management used in CORBA programming.

Build the server program	From within the Visual C++ IDE select <b>Build</b>   <b>Build server.exe</b> to compile and link the server.	
	By default, the project builds with debug settings and the server executable is stored in C:\OCGT\HelloExample\server\Debug\server.exe.	
	Close the server workspace by selecting File   Close Workspace	
Generate the client	Generate files for the client application using the Orbix code generation toolkit.	
	To create a client project using the IONA Orbix client/server wizard:	
	1. Open the Microsoft Visual C++ 6.0 IDE.	
	2. From the Visual C++ menus, select <b>File</b>  New	
	3. In the New dialog, click on the Projects tab.	

- 4. In the Projects tab, perform the following actions:
  - Select IONA Orbix v6.2 Client/Server Wizard
  - In the **Project name** text box, enter client
  - Under the **Location** text box, enter

C:\OCGT\HelloExample\client

New	<u>? ×</u>
Files Projects Workspaces Other Documents	
ATL COM AppWizard Cluster Resource Type Wizard Custom AppWizard Database Project DevStudio Add-in Wizard Extended Stored Proc Wizard IDNA Orbix v6.2 Client/Server Wizard SAPI Extension Wizard SAPI Extension Wizard MFC AppWizard (dll) MFC AppWizard (dll) MFC AppWizard (exe) New Database Wizard Utility Project Win32 Console Application Win32 Console Application	Project name: client Logation: C.VDCGTVHelloExample\client C freate new workspace C freate new workspace Dependency of: Platforms: Wwin32
	OK Cancel

- 5. Click OK.
- 6. The client/server wizard displays.

- 7. Answer two questions as follows:
  - What CORBA IDL file would you like to use for this project?
     Enter the location of hello.idl
  - Would you like to generate a working client or server? Select Client

IONA Orbix v6.2 Client/Serv	ver Wizard - Step 1 of 1	×
	What CORBA IDL file would you like to use for this project? C:\DCGT\HelloExample\hello.id  Browse View Remove Would you like to generate a working client or server? C Lient Server What object reference distribution method would you like to use? Stringified Object References Naming Service	
< <u>B</u> ack <u>N</u> ex	Einish Cancel <u>H</u> elp	-

- 8. To generate the client project, click **Finish**
- 9. The New Project Information scrollbox tells you about the generated files. Browse the information and select **OK**
- 10. The client workspace is generated with the following source files:



11. Read the text file ReadmeOrbixClient.txt

#### Complete the client program

Complete the implementation of the client main() function in the client.cxx file. You must add a couple of lines of code to make a remote invocation of the operation getGreeting() on the Hello object.

Search for the line where the call\_Hello\_getGreeting() function is called. Delete this line and replace it with the two lines of code highlighted in bold font below:

The object reference Hello1 refers to an instance of a Hello object in the server application. It is already initialized for you.

A remote invocation is made by invoking getGreeting() on the Hellol object reference. The ORB automatically establishes a network connection and sends packets across the network to invoke the HelloImpl::getGreeting() function in the server application.

The returned string is put into a C++ object, strv, of the type CORBA::string\_var. The destructor of this object will delete the returned string so that there is no memory leak in the above code.

```
      Build the client program
      From within the Visual C++ IDE select Build |Build client.exe to compile and link the client.

      By default, the project will build with debug settings and the client executable will be stored in
      C:\OCGT\HelloExample\client\Debug\client.exe.

      Close the client workspace by selecting File|Close Workspace.
```

#### Run the demonstration

Run the application as follows:

1. Run the Orbix services (if required).

If you have configured Orbix to use file-based configuration, no services need to run for this demonstration. Proceed to step **2**.

If you have configured Orbix to use configuration repository based configuration, start up the basic Orbix services.

> start\_domain-name\_services.bat

Where *domain-name* is the name of your configuration domain.

2. Set the Application Server Platform's environment.

> domain-name\_env.bat

3. Run the server program.

```
> cd C:\OCGT\HelloExample\server\Debug
> start server.exe
```

A new window opens and the server outputs the following lines:

```
Initializing the ORB
Writing stringified object reference to Hello.ref
Waiting for requests...
```

The server performs the following steps when it is launched:

- It instantiates and activates a single Hello CORBA object.
- The stringified object reference for the Hello object is written to the file C:\temp\Hello.ref.
- The server opens an IP port and begins listening on the port for connection attempts by CORBA clients.
- 4. Run the client program.
  - > cd C:\OCGT\HelloExample\client\Debug
  - > client.exe

The client outputs the following lines to the screen:

```
Client using random seed 0
Reading stringified object reference from Hello.ref
Greeting is: Hello World!
```

The client performs the following steps when it is run:

- It reads the stringified object reference for the Hello object from the C:\temp\Hello.ref file.
- It converts the stringified object reference into an object reference.
- It calls the remote Hello::getGreeting() operation by invoking on the object reference. This causes a connection to be established with the server and the remote invocation to be performed.
- 5. When you are finished, terminate all processes.
  - The server can be shut down by typing ctrl-c in the window where it is running.
- Stop the Orbix services (if they are running).
   From a DOS prompt enter:
  - > stop\_domain-name\_services

# **Development from the Command Line**

Starting point code for CORBA client and server applications can also be generated using the *idlgen* command line utility, which offers equivalent functionality to the client/server wizard presented in the previous section.

The idlgen utility can be used on Windows and UNIX platforms.

You implement the Hello World! application with the following steps:

- 1. Define the IDL interface, Hello.
- 2. Generate starting point code.
- Complete the server program by implementing the single IDL getGreeting() operation.
- Complete the client program by inserting a line of code to invoke the getGreeting() operation.
- 5. Build the demonstration.
- 6. Run the demonstration.

Define the IDL interface

Create the IDL file for the Hello World! application. First of all, make a directory to hold the example code:

#### Windows

> mkdir C:\OCGT\HelloExample

#### UNIX

% mkdir -p OCGT/HelloExample

Create an IDL file C:\OCGT\HelloExample\hello.idl (Windows) or OCGT/HelloExample/hello.idl (UNIX) using a text editor.

Enter the following text into the file hello.idl:

```
//IDL
interface Hello {
    string getGreeting();
};
```

This interface mediates the interaction between the client and the server halves of the distributed application.

```
Generate starting point code
```

Generate files for the server and client application using the CORBA Code Generation Toolkit.

In the directory C:\OCGT\HelloExample (Windows) or OCGT/HelloExample (UNIX) enter the following command:

```
idlgen cpp_poa_genie.tcl -all hello.idl
```

This command logs the following output to the screen while it is generating the files:

```
hello.idl:
cpp_poa_genie.tcl: creating it_servant_base_overrides.h
cpp_poa_genie.tcl: creating it_servant_base_overrides.cxx
cpp_poa_genie.tcl: creating HelloImpl.h
cpp_poa_genie.tcl: creating HelloImpl.cxx
cpp_poa_genie.tcl: creating server.cxx
cpp_poa_genie.tcl: creating client.cxx
cpp_poa_genie.tcl: creating call_funcs.h
cpp_poa_genie.tcl: creating call_funcs.cxx
cpp_poa_genie.tcl: creating it_print_funcs.h
cpp_poa_genie.tcl: creating it_print_funcs.cxx
cpp_poa_genie.tcl: creating it_print_funcs.cxx
cpp_poa_genie.tcl: creating it_random_funcs.h
cpp_poa_genie.tcl: creating it_random_funcs.cxx
cpp_poa_genie.tcl: creating it_random_funcs.cxx
```

You can edit the following files to customize client and server applications:

#### Client:

client.cxx

#### Server:

server.cxx HelloImpl.h HelloImpl.cxx

Complete the server program

Complete the implementation class, HelloImpl, by providing the definition of the HelloImpl::getGreeting() function. This C++ function provides the concrete realization of the Hello::getGreeting() IDL operation.

Edit the HelloImpl.cxx file, and delete most of the generated boilerplate code occupying the body of the HelloImpl::getGreeting() function. Replace it with the line of code highlighted in bold font below:

```
//C++
//File 'HelloImpl.cxx'
...
char *
HelloImpl::getGreeting() throw(
    CORBA::SystemException
)
{
    char * __result;
    _result = CORBA::string_dup("Hello World!");
    return _result;
}
....
```

The function  $CORBA::string_dup()$  allocates a copy of the "Hello World!" string on the free store. It would be an error to return a string literal directly from the CORBA operation because the ORB automatically deletes the return value after the function has completed. It would also be an error to create a copy of the string using the C++ new operator.

**Complete the client program**Complete the implementation of the client main() function in the client.cxx file. You must add a couple of lines of code to make a remote invocation of the getGreeting() operation on the Hello object.

Edit the client.cxx file and search for the line where the call\_Hello\_getGreeting() function is called. Delete this line and replace it with the two lines of code highlighted in bold font below:

The object reference Hellol refers to an instance of a Hello object in the server application. It is already initialized for you.

A remote invocation is made by invoking getGreeting() on the Hellol object reference. The ORB automatically establishes a network connection and sends packets across the network to invoke the

HelloImpl::getGreeting() function in the server application.

The returned string is put into a C++ object, strV, of the type CORBA::string\_var. The destructor of this object will delete the returned string so that there is no memory leak in the above code.

The Makefile generated by the code generation toolkit has a complete set of rules for building both the client and server applications.

To build the client and server complete the following steps:

- 1. Open a command line window.
- 2. Go to the .../OCGT/HelloExample directory.
- 3. Enter:

#### Windows

> nmake

Build the demonstration

#### UNIX

	% make -e
Run the demonstration	Run the application as follows:
	1. Run the Orbix services (if required).
	If you have configured Orbix to use file-based configuration, no services need to run for this demonstration. Proceed to step 2.
	If you have configured Orbix to use configuration repository based configuration, start up the basic Orbix services.
	Open a DOS prompt in Windows, or xterm in UNIX. Enter:
	start_domain-name_services
	Where <i>domain-name</i> is the name of the configuration domain.
	2. Set the Application Server Platform's environment.
	> domain-name_env
	3. Run the server program.
	Open a DOS prompt, or $xterm$ window (UNIX). From the
	$\texttt{C:} \verb  \texttt{OCGT} \verb  \texttt{HelloExample} \text{ directory enter the name of the executable} \\$
	file—server.exe (Windows) or server (UNIX).The server outputs the following lines to the screen:
	Initializing the ORB Writing stringified object reference to Hello.ref Waiting for requests
	The server performs the following steps when it is launched:
	• It instantiates and activates a single Hello CORBA object.
	• The stringified object reference for the Hello object is written to the local Hello.ref file.
	• The server opens an IP port and begins listening on the port for connection attempts by CORBA clients.

4. Run the client program.

Open a new DOS prompt, or xterm window (UNIX). From the C:\OCGT\HelloExample directory enter the name of the executable file—client.exe (Windows) or client (UNIX).

The client outputs the following lines to the screen:

```
Client using random seed 0
Reading stringified object reference from Hello.ref
Greeting is: Hello World!
```

The client performs the following steps when it is run:

- It reads the stringified object reference for the Hello object from the Hello.ref file.
- It converts the stringified object reference into an object reference.
- It calls the remote Hello::getGreeting() operation by invoking on the object reference. This causes a connection to be established with the server and the remote invocation to be performed.
- 5. When you are finished, terminate all processes.

Shut down the server by typing **ctrl-c** in the window where it is running.

6. Stop the Orbix services (if they are running).

From a DOS prompt in Windows, or xterm in UNIX, enter:

#### stop\_domain-name\_services

The passing of the object reference from the server to the client in this way is suitable only for simple demonstrations. Realistic server applications use the CORBA naming service to export their object references instead (see Chapter 18).

# Index

## Α

Application running 23, 28

## С

Client building 22 generating 19, 26 implementing 22, 27 Code generation toolkit idlgen utility 26 wizard 14 cpp\_poa\_genie.tcl 26

## Η

Hello World! example 12

#### Μ

Memory management string type 27

## 0

Object reference passing as a string 13

## S

Server building 19 generating 16, 26 implementing 19, 26 Services 23, 24, 29, 30 string\_dup() 27 String\_var 28

### W

Wizard for code generation 14

INDEX