



CORBA Tutorial C++

Version 6.1, December 2003

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 © 2001–2003 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: 26-Apr-2004

M 3 1 4 1

## Contents

| Chapter 1 Getting Started with C           | Drbix | 1  |
|--|-------|----|
| Creating a Configuration Domain            |       | 2  |
| Setting the Orbix Environment              |       | 9  |
| Hello World Example                        |       | 10 |
| Development Using the Client/Server Wizard |       | 12 |
| Development from the Command I             | Line  | 23 |
| Index                                      |       | 29 |

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 9  |
| Hello World Example                        | page 10 |
| Development Using the Client/Server Wizard | page 12 |
| Development from the Command Line          | page 23 |

In this chapter

### **Creating a Configuration Domain**

|       | 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.  |  |
|-------|---|--|
| -     | <ul> <li>Before creating a configuration domain, the following prerequisites must be satisfied:</li> <li>Orbix is installed.</li> <li>Some basic system variables are set up (in particular, the IT_PRODUCT_DIR, IT_LICENSE_FILE, and PATH variables).</li> <li>Fore more details, please consult the <i>Installation Guide</i>.</li> </ul> |  |
|       |   |  |
|       | 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. Specify the license location.  |  |
|       | 3. Choose expert mode and specify domain settings.  |  |
|       | 4. Specify services settings.   |  |
|       | 5. Review the summary window.   |  |
|       | 6. Finish configuration.  |  |

### Run itconfigure

To begin creating a new configuration domain, enter *itconfigure* at a command prompt. An **Introduction** window appears, as shown in Figure 1.

| Ӌ Orbix Configuratio                       | n  | ×   |
|--|--|---|
| Steps                                      | Introduction                               |   |
| <ol> <li>Introduction</li> <li></li> </ol> | Orbix Configuration<br>Welcome to the Orbi | x configuration tool.   |
|  | ☐ <u>C</u> reate                           | Create a configuration domain based on the defaults.            |
|  | Co <u>n</u> nect                           | Connect to a configuration domain on another host.              |
|  | eploy                                      | Deploy a configuration domain's services/replicas on this host. |
|  | 🔒 L <u>i</u> cense                         | Install and use a new license file.                             |
|  | ✓ Expert                                   | Use the advanced configuration wizard.                          |
|  |  |   |
|  |  | <back next=""> Finish Cancel</back>                             |

Figure 1: The itconfigure Introduction Window

### Specify the license location

If you have not already specified the license location by setting the IT\_LICENSE\_FILE environment variable (see "Licensing" on page 2), specify the location now by clicking the **License** button on the **Introduction** window (Figure 1 on page 3).

A License dialog box appears, as shown in Figure 2. Enter the license file location in the License File text field or use the **Browse** button to select the license file, then click **OK**.

|  | ×  |
|--|----|
| You are currently configured to use the license file located at  |    |
| 'c:\Orbix_61_GA\licenses.bt'   |    |
| You can update this license file by specifying a new location below. If you want<br>to use a license file from a different location without installing it, set the<br>IT_LICENSE_FILE environment variable, and run this tool again. |    |
| Install a new license file from  | ור |
| License File: Browse   |    |
| OK Cancel  |    |

Figure 2: The License Dialog Box

### Choose expert mode and specify domain settings

From the **Introduction** window (Figure 1 on page 3), click **Expert** to begin creating a configuration domain in expert mode. A **Domain Settings** window appears, as shown in Figure 3.

In the Domain Name text field, type simple. Select the File Based Domain option.

Make sure that the **Allow Insecure Communication** option is selected and the **Allow Secure Communication** option is unselected.

Click **Next>** to continue.

| 💙 Orbix Configuration | ×   |  |  |
|-----------------------|---|--|--|
| Steps                 | Domain Settings   |  |  |
| 1. Introduction       | Domain Details  |  |  |
| 2. Domain Settings    | Domain Name: simple   |  |  |
| 3. Services Settings  | Location Domain: simple.location  |  |  |
| 4. Summary            | ● File Based Domain ✓ Allow Insecure Communication                      |  |  |
| 5. Complete           | ○ Configuration Repository Domain □ Allow Secure Communication          |  |  |
|                       | Launch domain on machine startup (as system services).                  |  |  |
|                       | Generate EMS Configuration Files  |  |  |
|                       | Address Mode Policy for Object References: Short (unqualified) hostname |  |  |
|                       | Storage Location  |  |  |
|                       | Configuration Files Directory: c:\Orbix_61_GA\etc Browse                |  |  |
|                       | Configuration Database Directory: c:\Orbix_61_GA\war Browse             |  |  |
|                       | Defaults Load Localize  |  |  |
|                       | <back next=""> Finish Cancel</back>                                     |  |  |

Figure 3: The itconfigure Domain Settings Window

### Specify services settings

A Services Settings window appears, as shown in Figure 4.

In the Services Settings window, select the following services and components for inclusion in the configuration domain: Location, Node daemon, Management, Distributed Transaction, CORBA Interface Repository, CORBA Naming, and Demos.

Click **Next>** to continue.

| 💝 Orbix Configuration |                              |      |                                | X      |
|-----------------------|------------------------------|------|--------------------------------|--------|
| Steps                 | Services Settings            |      |                                |        |
| 1. Introduction       | Infrastructure               |      | Web Services                   |        |
| 2. Domain Settings    | Location                     | Edit | Web Services Container         | Edit   |
| 3. Services Settings  | 🗹 Node daemon                | Edit | Messaging                      |        |
|                       | 🗹 Management                 | Edit | CORBA Notification             | Edit   |
| 4. Summary            | Distributed Transaction      | Edit | CORBA Events                   | Edit   |
| 5. Complete           | Configuration                | Edit | 🔲 JMS (Java Messaging Service) | Edit   |
|                       | Directory                    |      | JMS/Notification Bridge        |        |
|                       | 🗹 CORBA Interface Repository | Edit | Security                       |        |
|                       | 🗹 CORBA Naming               | Edit | Firewall Proxy                 | Edit   |
|                       | CORBA Trader                 | Edit | 🗆 IONA Security                | Edit   |
|                       | CORBA Telco Logging          |      | Components                     |        |
|                       | Basic Logging                | Edit | 🗹 Demos                        |        |
|                       | Event Logging                | Edit |                                |        |
|                       | Notify Logging               | Edit |                                |        |
|                       | Clear All Check All          |      | -                              |        |
|                       |                              |      | <back next=""> Finish</back>   | Cancel |

Figure 4: The itconfigure Services Settings Window

### Review the summary window

You now have the opportunity to review the configuration settings in the **Summary** window, Figure 5. 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.

| 😼 Orbix Configuration | ×   |
|-----------------------|---|
| Steps                 | Summary   |
| 1. Introduction       | The following configuration settings will be deployed. Domain Name: simple                                  |
| 2. Domain Settings    | File Based Domain   |
| 3. Services Settings  | Configuration Files Directory = c:\Orbix_61_GA\etc<br>Configuration Database Directory = c:\Orbix_61_GA\var |
| 4. Summary            | Configuration included for demos Location Service   |
| 5. Complete           | Manual Activation<br>Host = fboltan<br>IIOP Port = 3075   |
|                       | Node daemon Service<br>Manual Activation  |
|                       | Host = fboltan<br>IIOP Port = 53079   |
|                       | CORBA Interface Repository Service Automatic Activation   |
|                       | Host = fboltan  |
|                       | IIOP Port Enabled<br>CORBA Naming Service   |
|                       | Automatic Activation  |
|                       | Host = fboltan<br>IIOP Port Enabled   |
|                       | Management Service  |
|                       | Save  |
|                       | <back next=""> Finish Cancel</back>   |

Figure 5: The itconfigure Summary 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 **Complete** window with a message similar to that shown in Figure 6.

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

| 😼 Orbix Configuration | x   |
|-----------------------|---|
| Steps                 | Complete  |
|                       | Configuration is now complete, see details below.<br>Configuration completed successfully.                  |
| 2. Domain Settings    | You can view the log in 'c:\Orbix_61_GA\etc\log'.   |
|                       | To set your environment for this configuration domain run:<br>c:\Orbix_61_GA\etc\bin\simple_env.bat         |
|                       |   |
| 5. Complete           | To start the services in this configuration domain run:<br>c:\Orbix_61_GA\etc\bin\start_simple_services.bat |
|                       | To stop the services in this configuration domain run:<br>c:\Orbix_61_GA\etc\bin\stop_simple_services.bat   |
|                       |   |
|                       |   |
|                       |   |
|                       |   |
|                       |   |
|                       | <pre><back next=""> Finish Cancel</back></pre>  |

Figure 6: Finishing Configuration

### **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 Administrator's Guide 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 7.

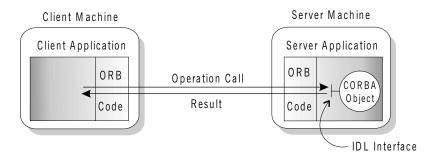


Figure 7: Client makes a single operation call on a server

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<br>Microsoft Visual Studio integrated development environment (IDE) that<br>enables you to generate starting point code for CORBA applications.<br>If you are not working on a Windows platform or if you prefer to use a<br>command line approach to development, see "Development from the<br>Command Line" on page 23. |  |
|-------------------------------------|---|--|
| Installing the client/server wizard | Ordinarily, the client/server wizard is installed at the same time as Orbix. If the wizard is not on your system, however, consult the <i>Installation Guide</i> for instructions on how to install it.   |  |
| 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:                                  |  |
|                                     | Orbix Include Directory   |  |
|                                     | OrbixInstall/asp/6.1/include  |  |
|                                     | Orbix Library Directory   |  |
|                                     | Orbix Library Directory<br>OrbixInstall/asp\6.1\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: //IDL interface Hello { string getGreeting(); }; 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 v6.1 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 | L  |
|   | Dynamic-L<br>Static Librz<br>Server<br>Logation:<br>C-\DCGT\HelloExample\server<br>C (geate new workspace<br>Add to current workspace<br>Dependency of:<br>client<br>Platforms:<br>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

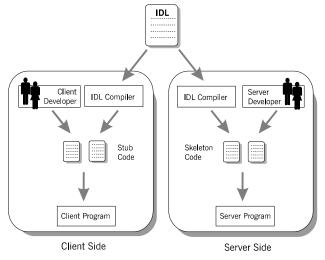
| IONA Orbix E2A v6.1 Client/ | Server Wizard - Step 1 of 2   | × |
|-----------------------------|---|---|
|                             | What CORBA IDL file would you like to use for this project?  C:\OCGT\HelloExample\hello.idl  Browse Uiew Bernove  Would you like to generate a working client or server?  C Client C Server  What object reference distribution method would you like to use?  Stringified Object References Naming Service |   |
| < <u>B</u> ack <u>N</u> ext | > <u>Finish</u> Cancel <u>H</u> elp   |   |

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

| IONA Orbix E2A v6.1 Client/ | 'Server Wizard - Step 2 of 2   |
|-----------------------------|--|
| IONA                        | PDA Server Generation Options           Image: Servert inheritance matches IDL inheritance         Image: Create a multi-threaded server           Servant Implementation Approach         Servant Implementation Approach |
| oo<br>Toto<br>Vibix         | Inheritance from POA base classes     IIE (Delegation)     Servant Management Strategy   |
|                             | Create and activate <u>s</u> ervants in mainline   |
|                             | C ServantActivator creates servants on demand<br>C ServantLocator creates servants per-invocation<br>C Use a single default servant for many objects   |
| < <u>B</u> ack <u>N</u> ex  | Einish 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>   <b>New</b>   |  |
|                          | 3. In the New dialog, click on the Projects tab.  |  |

- 4. In the Projects tab, perform the following actions:
  - Select IONA Orbix v6.1 Client/Server Wizard
  - In the **Project name** text box, enter client
  - Under the **Location** text box, enter

C:\OCGT\HelloExample\client

| New  |  | ? >   |
|--|--|---|
| Files Projects Workspaces Other  | Documents                              |   |
| ATL COM AppWizard<br>Cluster Resource Type Wizard<br>Cluster Resource Type Wizard<br>Custom AppWizard<br>DevStudio Add-in Wizard<br>Extended Stored Proc Wizard<br>IDNA Orbix E2A v6.1 Client/Server Wizard<br>IDAN Orbix E2A v6.1 Client/Server Wizard<br>MFC ActiveX ControlWizard<br>MFC ActiveX ControlWizard<br>MFC AppWizard (dll)<br>MFC AppWizard (dll)<br>MFC AppWizard (exe)<br>New Database Wizard<br>Utility Project | (Win32 Dynamic-L<br>Win32 Static Libra | Project <u>name:</u> client Logation: C:\OCGT\HelloExample\client C Cteate new workspace C Add to current workspace Dependency of: server |
| Win32 Application     Win32 Console Application  | F                                      | Platforms:<br>Win32   |
|  | <u>Ľ</u>                               | 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 E2A v6.1 Client/ | Server Wizard - Step 1 of 1  | × |
|-----------------------------|--|---|
|                             | What CORBA IDL file would you like to use for this project? C:\DCGT\HelloExample\hello.idl Browse View Bernove 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 Help   | - |

- 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 Appliation Server Platform's environment.

> domain-name\_env.bat

3. Run the server program.

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

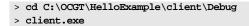
The server outputs the following lines to the screen:

```
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 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.

Open a new MS-DOS prompt.



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 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.

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 <b>2</b> . |
|                       | 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:   |
|                       | <pre>start_domain-name_services</pre>  |
|                       | Where <i>domain-name</i> is the name of the configuration domain.  |
|                       | 2. Set the Appliation Server Platform's environment.   |
|                       | > domain-name_env  |
|                       | 3. Run the server program.   |
|                       | Open a DOS prompt, or $xterm$ window (UNIX). From the  |
|                       | C:\OCGT\HelloExample 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<br>Writing stringified object reference to Hello.ref<br>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   |

4. Run the client program.

connection attempts by CORBA clients.

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 17).

# Index

### Α

Application running 21, 26

### С

Client building 20 generating 17, 24 implementing 20, 25 Code generation toolkit idlgen utility 24 wizard 12 cpp\_poa\_genie.tcl 24

### Η

Hello World! example 10

### Μ

Memory management string type 25

### 0

Object reference passing as a string 11

### S

Server building 17 generating 13, 24 implementing 17, 24 Services 21, 22, 27, 28 string\_dup() 25 String\_var 26

### W

Wizard for code generation 12

INDEX