



CORBA Tutorial Java

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 2

Contents

Chapter 1 Getting Started with Orbix	1
Creating a Configuration Domain	2
Setting the Orbix Environment	9
Setting ORB Properties for the Orbix ORB	10
Hello World Example	12
Development from the Command Line	14
Index	1

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 build files. 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
Setting ORB Properties for the Orbix ORB	page 10
Hello World Example	page 12
Development from the Command Line	page 14

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. 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.

😼 Orbix Configuratio	n	×
Steps	Introduction	
 Introduction 	Orbix Configuration Welcome to the Orbi	« configuration tool.
	♀ <u>C</u> reate	Create a configuration domain based on the defaults.
	Connect	Connect to a configuration domain on another host.
	eploy	Deploy a configuration domain's services/replicas on this host.
	License	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**.

	x
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 to use a license file from a different location without installing it, set the IT_LICENSE_FILE environment variable, and run this tool again.	
Install a new license file from	٦
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				×
Steps	Services Settings			
1. Introduction	Infrastructure		Web Services	
2. Domain Settings	Location	Edit	Web Services Container	Edit
3. Services Settings	🗹 Node daemon	Edit	Messaging	
4 Cummon	🗹 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	
Steps 1. Introduction 2. Domain Settings 3. Services Settings 4. Summary 5. Complete	Summary The following configuration settings will be deployed. Domain Name: simple File Based Domain Configuration Files Directory = c:\Orbix_61_GA\etc Configuration included for demos Location Service Manual Activation Host = fboltan IIOP Port = 3075 Node daemon Service Manual Activation Host = fboltan IIOP Port = 53079 CORBA Interface Repository Service Automatic Activation Host = fboltan IIOP Port = called CORBA Naming Service Automatic Activation Host = fboltan IIOP Port Enabled CORBA Naming Service Automatic Activation Host = fboltan IIOP Port Enabled Management Service Automatic Activation Host = fboltan IIOP Port Enabled Management Service Measurel Betweet	
	<back next=""> Finish Cano</back>	el

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	<u>×</u>
Steps	Complete
 Introduction Domain Settings Services Settings Summary Complete 	Configuration is now complete, see details below. Configuration completed successfully. You can view the log in 'c:NOrbix_61_GAletclog'. To set your environment for this configuration domain run: c:NOrbix_61_GAletclbin\simple_env.bat To start the services in this configuration domain run: c:NOrbix_61_GAletclbin\start_simple_services.bat To stop the services in this configuration domain run: c:NOrbix_61_GAletclbin\stop_simple_services.bat
	<back next=""> Finish Cancel</back>

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. Your Java development kit (JDK) is configured to use the Orbix ORB runtime (see "Setting ORB Properties for the Orbix ORB" on page 10). The Administrator's Guide contains more information on Orbix configuration, and details of Orbix command line utilities.
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
	<pre>> set JAVA_HOME=YourJdkDir > config-dir\etc\bin\domain-name_env.bat</pre>
	UNIX
	<pre>% JAVA_HOME=YourJdkDir ; export JAVA_HOME % . config-dir/etc/bin/domain-name_env</pre>
	<i>YourJdkDir</i> is the root directory of the Java development kit that you want to use with Orbix. See the <i>Installation Guide</i> for details of supported Java platforms.

config-dir is the root directory where the Appliation Server Platform stores its configuration information. You specify this directory while configuring your domain. *domain-name* is the name of a configuration domain.

Setting ORB Properties for the Orbix ORB

SUN's Java development kit (JDK) comes with a built-in ORB runtime that is used by default. However, you cannot use SUN's ORB runtime with Orbix applications. You must configure the JDK to use the Orbix ORB runtime instead by setting system properties org.omg.CORBA.ORBClass and org.omg.CORBA.ORBSingletonClass to the appropriate values. You can set the ORB properties in one of the following ways:

- Using the iona.properties file
- Using Java interpreter arguments

Using the iona.properties file

Setting system properties org.omg.CORBA.ORBClass and org.omg.CORBA.ORBSingletonClass in the iona.properties file is the preferred way to configure your JDK to use the Orbix ORB runtime.

Location of the iona.properties file

The iona.properties file is located in the *JDKHome*/jre/lib directory, where *JDKHome* is the JDK root directory.

Contents of the iona.properties file

The iona.properties file should contain the following two lines of text:

```
org.omg.CORBA.ORBClass=com.iona.corba.art.artimpl.ORBImpl
org.omg.CORBA.ORBSingletonClass=
    com.iona.corba.art.artimpl.ORBSingleton
```

The first line sets org.omg.CORBA.ORBClass to the name of a class that implements org.omg.CORBA.ORB.

The second line sets ${\tt org.omg.CORBA.ORBSingletonClass}$ to the name of a
class that implements the static ORB instance returned from
org.omg.CORBA.ORB.init() (taking no arguments).

WARNING: By setting system properties org.omg.CORBA.ORBClass and org.omg.CORBA.ORBSingletonClass in the iona.properties file, as detailed above, you effectively specify the Orbix ORB classes as the ORB runtime for the JDK. This might affect other applications that use the same JDK but want to use different ORB classes—if this is the case, you should consider using one of the alternative mechanisms for setting ORB properties, given in the following sub-sections.

Using Java interpreter arguments	You can use the <i>-Dproperty_name=</i> property_value option on the Java Interpreter to specify the org.omg.CORBA.ORBClass and org.omg.CORBA.ORBSingletonClass properties. For example, to set the ORB properties for an orbix_app Orbix application:
	<pre>java -Dorg.omg.CORBA.ORB=com.iona.corba.art.artimpl.ORBImpl\ -Dorg.omg.CORBA.ORBSingletonClass=\ com.iona.corba.art.artimpl.ORBSingleton orbix_app</pre>

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 Java 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 from the Command Line

Starting point code for CORBA client and server applications can be generated using the idlgen command line utility.

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 java_poa_genie.tcl -all -jP HelloExample hello.idl
```

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

```
hello.idl:
```

```
java_poa_genie.tcl: creating idlgen/RandomFuncs.java
java_poa_genie.tcl: creating
    idlgen/HelloExample/RandomHello.java
java_poa_genie.tcl: creating idlgen/RandomHelloExample.java
java_poa_genie.tcl: creating HelloExample/HelloCaller.java
java_poa_genie.tcl: creating HelloExample/client.java
java_poa_genie.tcl: creating HelloExample/HelloImpl.java
java_poa_genie.tcl: creating HelloExample/server.java
java_poa_genie.tcl: creating HelloExample/server.java
```

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

Client:

HelloExample/client.java

Server:

HelloExample/server.java HelloExample/HelloImpl.java

Complete the server program

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

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

```
//Java
//File 'HelloImpl.java'
...
public java.lang.String getGreeting()
throws org.omg.CORBA.SystemException
{
    java.lang.String __result;
    __result = "Hello World!";
    return _result;
}
...
```

Complete the client program

Complete the implementation of the client main() function in the client.java 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.java file and search for the line where the HelloExample.HelloCaller.getGreeting() method is called. Delete this line and replace it with the line of code highlighted in bold font below:

```
//Java
//File: 'client.java'
...
    try
    {
        ...
        // Exercise interface HelloExample.Hello.
        //
        tmp_ref = read_reference("Hello.ref");
        HelloExample.Hello Hello1 =
            HelloExample.Hello Hello1 =
            HelloExample.HelloHelper.narrow(tmp_ref);
        System.out.println("Greeting is: " +
        Hellol.getGreeting());
     }
     catch(Exception ex)
     {
        System.out.println("Unexpected COREA exception: " + ex);
     }
...
```

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() method in the server application.

Build the demonstration

The itant utility—a Java-based build tool—is used to build the generated Java code. For more details about itant, see http://jakarta.apache.org/ant. The itant utility is bundled with Orbix.

The generated file build.xml is used to build this demonstration. This file contains the rules for building the Hello World! application in an XML format that is understood by the itant utility.

To build the client and server complete the following steps:

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

3	5. Enter:
	> itant
-	
Run the demonstration Run 1.	Run the application as follows:
	. 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:
	<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). Enter the following command:
	itant runserver
	The server outputs the following lines to the screen:
	Buildfile: build.xml
	<pre>runserver: [java] Initializing the ORB [java] Writing stringified object reference to Hello.ref [java] Waiting for requests</pre>
	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). Enter the following command:

itant runclient

The client outputs the following lines to the screen:

Buildfile: build.xml runclient: [java] Reading stringified object reference from Hello.ref Greeting is: Hello World!

```
Total time: 3 seconds
```

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).

CHAPTER 1 | Getting Started with Orbix

Index

Α

Application running 17

В

build.xml 17 building applications 17

С

Client generating 15 implementing 16 Code generation toolkit idlgen utility 15

G

Genie-generated application package name 15

Η

Hello World! example 12

J

java_poa_genie.tcl 15

0

Object reference passing as a string 13 ORBClass 10 org.omg.CORBA.ORBClass 10 org.omg.CORBA.ORBSingletonClass 11

Ρ

Package name 15

S

Server generating 15 implementing 15 Services 18, 19 INDEX