Orbix 6.3.7

Deployment Guide

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK

http://www.microfocus.com

2014-06-12

Contents

Preface

Orbix enables you to develop and deploy enterprise-level applications across different platform and programming language environments. This guide explains how to setup an Orbix environment, and examines the Orbix configuration and deployment process in detail.

Note: The scope of this guide is limited to the configuration and deployment features that are supported by Micro Focus. Unsupported configuration and deployment features are not documented. These are proprietary features and are subject to change without notice.

Audience

This guide is aimed at system administrators who are setting up Orbix environments.

It is also aimed at programmers who are developing and deploying Orbix applications. It contains advanced information about customizing Orbix configuration and deployment. This guide should be read in conjunction with the *Orbix Administrator's Guide.*

Related documentation

The document set for Orbix includes the following related documentation:

- *Orbix Administrator's Guide*
- *Orbix Configuration Reference*
- *Orbix Management User's Guide*

Typographical conventions

This guide uses the following typographical conventions:

`Constant width`	Constant width (courier font) in normal text represents portions of code and literal names of items such as classes, functions, variables, or data structures. For example, text might refer to the `CORBA::Object` class. Constant width paragraphs represent code examples or information a system displays on the screen. For example: `#include <stdio.h>`
Italic	Italic words in normal text represent *emphasis* and *new terms*. Italic words or characters in code and commands represent variable values you must supply, such as arguments to commands or path names for your particular system. For example: `% cd /users/`*your_name* **Note:** Some command examples may use angle brackets to represent variable values you must supply. This is an older convention that is replaced with *italic* words or characters.

Keying conventions

This guide may use the following keying conventions:

No prompt	When a command's format is the same for multiple platforms, a prompt is not used.
`%`	A percent sign represents the UNIX command shell prompt for a command that does not require root privileges.
`#`	A number sign represents the UNIX command shell prompt for a command that requires root privileges.
`>`	The notation > represents the DOS or Windows command prompt.

`...`
`.`
`.`
`.`
	Horizontal or vertical ellipses in format and syntax descriptions indicate that material has been eliminated to simplify a discussion.

[ ]	Brackets enclose optional items in format and syntax descriptions.
{ }	Braces enclose a list from which you must choose an item in format and syntax descriptions.

|	A vertical bar separates items in a list of choices enclosed in { } (braces) in format and syntax descriptions.

Contacting Micro Focus

Our Web site gives up-to-date details of contact numbers and addresses.

Further Information and Product Support

Additional technical information or advice is available from several sources.

The product support pages contain a considerable amount of additional information, such as:

- The WebSync service, where you can download fixes and documentation updates.
- The Knowledge Base, a large collection of product tips and workarounds.
- Examples and Utilities, including demos and additional product documentation.

To connect, enter http://www.microfocus.com in your browser to go to the Micro Focus home page.

Note:
Some information may be available only to customers who have maintenance agreements.

If you obtained this product directly from Micro Focus, contact us as described on the Micro Focus Web site, http://www.microfocus.com. If you obtained the product from another source, such as an authorized distributor, contact them for help first. If they are unable to help, contact us.

Information We Need

However you contact us, please try to include the information below, if you have it. The more information you can give, the better Micro Focus SupportLine can help you. But if you don't know all the answers, or you think some are irrelevant to your problem, please give whatever information you have.

- The name and version number of all products that you think might be causing a problem.
- Your computer make and model.
- Your operating system version number and details of any networking software you are using.
- The amount of memory in your computer.
- The relevant page reference or section in the documentation.
- Your serial number. To find out these numbers, look in the subject line and body of your Electronic Product Delivery Notice email that you received from Micro Focus.

Contact information

Our Web site gives up-to-date details of contact numbers and addresses.

Additional technical information or advice is available from several sources.

The product support pages contain considerable additional information, including the WebSync service, where you can download fixes and documentation updates. To connect, enter http://www.microfocus.com in your browser to go to the Micro Focus home page.

If you are a Micro Focus SupportLine customer, please see your SupportLine Handbook for contact information. You can download it from our Web site or order it in printed form from your sales representative. Support from Micro Focus may be available only to customers who have maintenance agreements.

You may want to check these URLs in particular:

- http://www.microfocus.com/products/corba/orbix/orbix-6.aspx (trial software download and Micro Focus Community files)
- https://supportline.microfocus.com/productdoc.aspx. (documentation updates and PDFs)

To subscribe to Micro Focus electronic newsletters, use the online form at:

http://www.microfocus.com/Resources/Newsletters/infocus/newsletter-subscription.asp

Getting Started

This chapter introduces Orbix configuration and deployment. It also explains how to run the Orbix configuration tool and how to license your Orbix installation.

Introduction

This section introduces Orbix configuration and deployment. It includes the following topics:

- "Configuration and deployment process"
- "Orbix Configuration tool (itconfigure)"
- "Orbix deployment descriptor"
- "Orbix deployer and component XML files"
- "Deployed configuration models"
- "Implementation Repository"

Configuration and deployment process

Figure 1 shows a general overview of the Orbix configuration and deployment process.

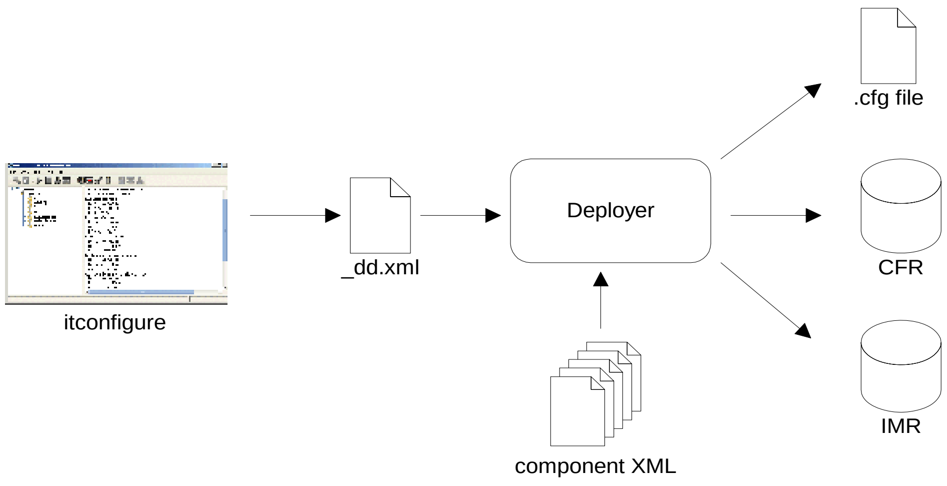


Figure 1: *Overview of Orbix Configuration and Deployment*

Figure 1 can be described as follows:

1. The Orbix configuration tool (`itconfigure` command) is used to generate the domain deployment descriptor (*`domain-name`*`_dd.xml`).
2. The deployer parses the deployment descriptor, taking input from XML templates for the various Orbix components and services.

3. The deployer deploys the configuration domain into a configuration domain file or the Configuration Repository (CFR), and also into the Implementation Repository (IMR).

The components in Figure 1 are described in more detail in the topics that follow.

Orbix Configuration tool (itconfigure)

The Orbix configuration tool (`itconfigure` command) guides you through configuring Orbix components in your environment. You can use it to perform tasks such as installing a license, creating a configuration domain, or linking to an existing configuration domain.

You can run the Orbix configuration tool in GUI and command-line modes. You should create a domain deployment descriptor by using this tool in GUI mode (shown in Figure 1).

GUI mode

The GUI creates a domain deployment descriptor file (*domain-name*`_dd.xml`). You can create the configuration domain specified by this deployment descriptor using the GUI. Alternatively, you can save the descriptor and create your domain later. GUI mode imposes constraints and performs validity checking (for example, on the combinations of Orbix services that are permitted).

Command-line mode

You can also create a configuration domain in command-line mode by passing a previously created deployment descriptor to the `itconfigure` command.

For detailed information on how to use the Orbix configuration tool, see "Configuring and Deploying a Domain".

Orbix deployment descriptor

The domain deployment descriptor (*domain-name*`_dd.xml`) describes the contents of a configuration domain. For example, for a domain named `sample-domain`, a deployment descriptor named *sample-domain*`_dd.xml` specifies the services, components, features and hosts that are included in that domain. By default, the deployment descriptor file is stored in your `etc\domains` directory, for example:

```
<install-dir>\etc\domains\sample-domain\sample-domain_dd.xml
```

The Orbix configuration GUI generates the deployment descriptor, which it then uses to automatically deploy the specified configuration into your environment (as shown in Figure 1).

Alternatively, you can also save the deployment descriptor before it is deployed by the GUI, and then perform a command-line deployment at a later stage. This is particularly useful if you want to customize your configuration by editing your deployment descriptor, or use multiple deployments with the same configuration.

For full details of how to perform a command-line deployment, see "Configuring a Machine with no GUI" on page 39. For details on the contents of the deployment descriptor file, see "Orbix Deployment Descriptors".

Orbix deployer and component XML files

The deployer parses the deployment descriptor produced by the Orbix configuration GUI. It also takes input from the XML templates for the various Orbix components and services (for example, `event_log.xml`). By default, these XML templates are stored in the following directory:

install-dir`\asp\`*version*`\etc\conf`

These template files all conform to a standard XML format as specified by the `ABDeploy.dtd` file. For details of this DTD file, see "Orbix Deployment DTD".

You can also specify custom XML files to the deployer. For details, see "Using Custom XML Files" on page 51.

Deployed configuration models

Depending on which option you chose in the configuration GUI, the deployer gathers your configuration information into either a configuration file or a Configuration Repository (CFR), and creates scripts to start and stop the domain services.

The Interoperable Object References (IORs) that the deployer obtains by preparing the domain services are an essential part of this configuration domain data. If these are stored in a file, and clients need access to these IORs, you need to make sure that this file can be accessed by all clients (using NFS or similar network services). If you are dealing with a larger number of clients, or expect to modify configuration data, using a Configuration Repository might be your preferred choice.

A Configuration Repository is a centralized database for all configuration information. This centralized configuration model is suitable for environments with a potentially large number of clients and servers, or when configuration is likely to change.

Implementation Repository

The deployer also stores server process information in the Implementation Repository (IMR). This specifies whether the process can be started up on demand by a node daemon, and includes details such as POA names, and ORB names.

For more details on Orbix configuration models and the IMR, see the *Orbix Administrator's Guide*.

Orbix Configuration Tool

The Orbix Configuration tool guides you through licensing and configuring the components in your Orbix environment. You can also use this tool to manage your environment at runtime. This section includes the following:

- "Configuration setup tasks"
- "Runtime management tasks"
- "Example screen"

Configuration setup tasks

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

- Install or update your license.
- Create a configuration domain.
- Deploy services into a configuration domain.
- Link to existing configuration domains.
- Create server replicas for clustering.
- Add services to existing configuration domains.

The Orbix configuration tool analyzes your installation and provides you with the options available for your system.

Runtime management tasks

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

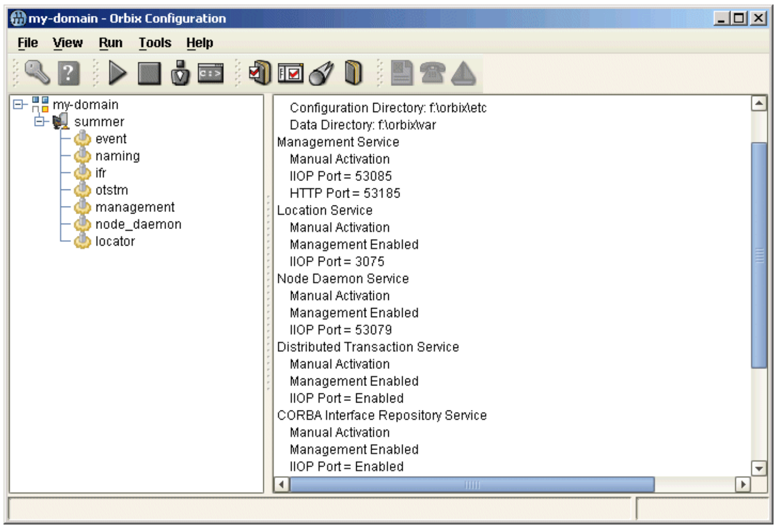
- Start and stop your Orbix services.
- Open a command prompt configured for your domain.
- Launch the Administrator Web Console.
- Launch other configuration tools (for example, Orbix Configuration Explorer).
- Open other GUI tools for specific Orbix services (for example, Orbix Notification Service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

For information on using runtime tools such as the Administrator Web Console and the Orbix Configuration Explorer, see the *Orbix Management User's Guide*.

Example screen

Figure 2 shows a newly created configuration domain named `my-domain`. The left pane displays details such as the domain and machine name, and all the services that have been configured. The right pane displays summary information about the domain.

While the toolbar across the top displays buttons for licensing and various runtime options, such as starting services, and launching other tools.



```
my-domain
  summer
    event
    naming
    ifr
    otstm
    management
    node_daemon
    locator
```

```
  Configuration Directory: f:\orbix\etc
  Data Directory: f:\orbix\var
Management Service
  Manual Activation
  IIOP Port = 53085
  HTTP Port = 53185
Location Service
  Manual Activation
  Management Enabled
  IIOP Port = 3075
Node Daemon Service
  Manual Activation
  Management Enabled
  IIOP Port = 53079
Distributed Transaction Service
  Manual Activation
  Management Enabled
  IIOP Port = Enabled
CORBA Interface Repository Service
  Manual Activation
  Management Enabled
  IIOP Port = Enabled
```

Figure 2: *Orbix Configuration GUI*

Running the Orbix Configuration Tool

This section explains the full syntax of the `itconfigure` command, which is used to run the Orbix Configuration tool. It includes the following:

- "Requirements"
- "Command syntax"
- "Orbix Configuration screen"

Requirements

Before you run the configuration tool, check the following system requirements:

- Set `JAVA_HOME` so it points to your current Java installation.
- Set UNIX access permissions to account for the following contingencies:
  - The configuration tool must have write access to directories `/var/opt/iona` and `/etc/opt/iona`. These directories are usually restricted to accounts with `superuser` privileges.
  - The configuration tool prompts you to designate a user to run domain services, and sets ownership of files and directories accordingly.
- Set the `IT_PRODUCT_DIR` environment variable to point to the latest Orbix installation on your system.

Command syntax

To run the configuration tool, use this command syntax:

```
itconfigure
              [-ORBproduct_dir install_dir]
              [-ORBlicense_file license_file]
              [-nogui]
              [-gui]
              [-load, -l domain_descriptor]
              [-deployed_descriptor, -d file]
              [-compatible]
              [-entities file]
              [-save, -s file]
              [-localize]
              [-from host]
              [-name domain_name]
              [-link, -i hostname]
              [-host, -h hostname]
              [-multihome hostname]
              [-etc config_dir]
              [-var var_dir]
              [-range, -r base_port]
              [-port iiop_port]
              [-tlsport tls_port]
              [-ndport iiop_port]
              [-ndtlsport tls_port]
              [-credentials credentials]
              [-hostnamePolicy policy]
              [-libs, -L path]
              [-substitutions file]
              [-listen_address_list list]
              [-remove replica]
              [-reprepare service]
              [-service service]
              [-instance service]
              [-add_master replica]
              [-jvm64]
              [-help, -?]
              [-demos]
```

The configuration tool options are described as follows:

`-ORBproduct_dir` *install_dir*	Specifies your installation directory when Orbix is installed in a non-default location and the `IT_PRODUCT_DIR` environment variable is not set.
`-ORBlicense_file` *license_file*	Specifies your license directory when the Orbix license file is not stored in the default location and the `IT_LICENSE_FILE` environment variable is not set. For more details, see "Licensing your Orbix Environment".
`-nogui`	Runs the configuration tool silently. This option can be used with `-load`, `-link`, and `-save`. For example, see "Replicating Services in a Domain".

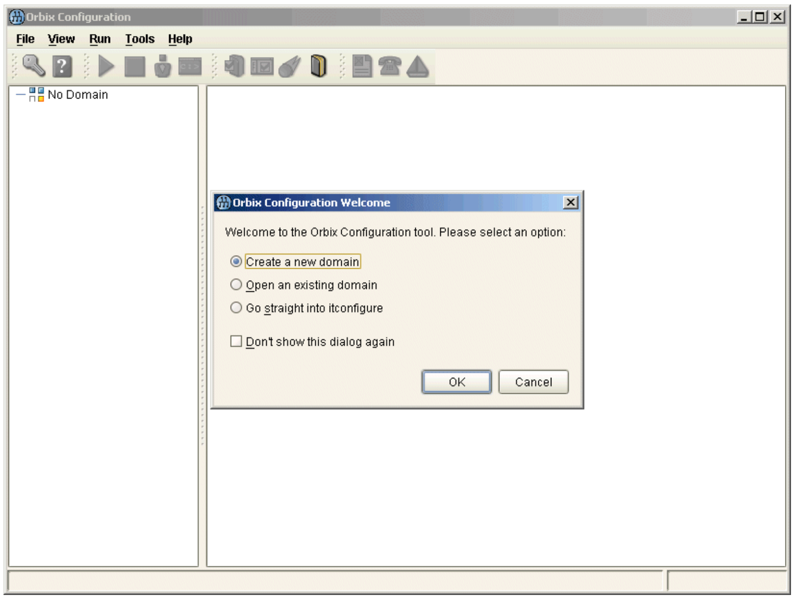
`-gui`	Runs the configuration tool GUI. This is the default.
`-load, -l` *domain_descriptor*	Loads a preconfigured domain descriptor file. When used in conjunction with `-nogui`, silently deploys the local parts of the configuration defined in the deployment descriptor. For more details, see "Deploying a Distributed Domain".
`-deployed_descriptor, -d` *file*	Specifies the deployment descriptor of an existing configuration domain. This enables you to add an additional service to an existing domain.
`-compatible`	For interoperability with previous versions, this loads the specified file with the `-load` option as an Orbix 5.x driver file. For more details, see "Migrating from Orbix 5.1 Driver Files".
`-entities` *filename*	Uses the specified entities file when loading the driver file specified with the `-load` option. For more details, see "Migrating from Orbix 5.1 Driver Files".
`-save, -s` *filename*	Saves a deployment descriptor in the specified file. When used with `-nongui`, this option will not deploy the saved configuration.
`-localize`	Replaces all deployment nodes in a descriptor with the local host. For more details, see "Localizing a Preconfigured Domain".
`-name` *domain_name*	Specifies the name of the domain. The specified name overrides the name in a loaded domain descriptor. For more details, see "Changing the domain name"
`-link` *cfr_host*	Specifies the machine which hosts the domain's configuration repository.
`-host, -h` *hostname*	Specifies the name of the domain's host machine. This setting overrides the setting in a loaded domain descriptor.
`-multihome` *hostname*	Denotes that the specified host is virtual on a multi-homed host. For more details, see "Deploying on Multi-homed Machines" on page 41.
`-etc` *etc_dir*	Specifies the directory where configuration information is stored.
`-var` *var_dir*	Specifies the directory where database files are stored.
`-range` *base_port*	Specifies the base port number from which to begin allocating port numbers. This option is only used in conjunction with `-nogui`.

`-port` *iiop_port*	Overrides the default CFR IIOP port when used with `-link`.
`-tlsport` *tls_port*	Overrides the default CFR TLS port when used with `-link`.
`-ndport` *iiop_port*	Overrides the default node daemon IIOP port when used with `-link`.
`-ndtlsport` *tls_port*	Overrides the default node daemon TLS port when used with `-link`.
`-libs, -L` *path*	Prefixes the library path to the built-in path used when preparing and running Orbix services. For more details, see "Specifying Custom Library Paths" on page 50.
`-substitutions` *file*	Specifies substitution file in properties file syntax containing values for substitution variables. For more details, see "Specifying Custom Certificates" on page 59.
`-listen_address_list` *list*	Listen on specified addresses
`-credentials`	Specifies credentials in the following format: `"username=<name>, \ password_file=<file>,domain=<domain>"`
`-hostnamePolicy` *policy*	Specifies the address mode policy for IORS. Value can be one of the following: • `ip` (IP addresses) • `localhost` ('localhost'), • `localhost_ip` ('`127.0.0.1`'), • `long` (fully qualified hostname), • `short` (unqualified hostname—the default). For more details, see "Specifying Address Mode Policies" on page 54.
`[-remove` *replica*`]`	Permanently removes a replica service from an existing domain (for example, a replica locator). For more details, see "Updating an Existing Domain" on page 26.
`[-reprepare` *service*`]`	Re-prepares an existing indirect persistent service after creating or removing a replica locator.
`[-service` *service*`]`	Adds a service to an existing domain.
`[-instance` *service*`]`	Adds a replica instance that was configured as the preferred master replica to an existing domain.
`[-add_master` *replica*`]`	Adds a new preferred master. This is used when you remove a replica service from an existing domain.
`[-jvm64]`	On Solaris, enables support for a 64 bit version of Java.

`-help, -?`	Displays an explanation of the command flags.
`-demos`	Specifies the configuration needed to run the Orbix demos in the domain.

Orbix Configuration screen

When the Orbix configuration tool first runs for the first time, it displays a screen similar to that shown in Figure 3:



Welcome to the Orbix Configuration tool. Please select an option:

[✓ Create a new domain]
Open an existing domain
Go straight into itconfigure

Don't show this dialog again

Figure 3: *Main Configuration Window*

Using the **Orbix Configuration Welcome** dialog, you can perform basic configuration setup tasks, such as create a new configuration domain, or update an existing one. For detailed information on how to perform all the main configuration tasks, see "Configuring and Deploying a Domain".

Licensing your Orbix Environment

The Orbix configuration tool enables you to specify the location of your Orbix license file.

Note: You must first specify your license details before you can perform actions such as creating an Orbix configuration domain.

Specifying a license file

To specify a license file:

1. From the Orbix configuration tool main menu, select **Tools**>**License**. Alternatively, click the **License** button in the toolbar, shown in Figure 4:

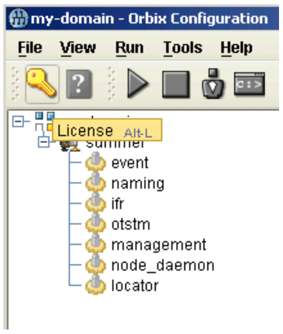
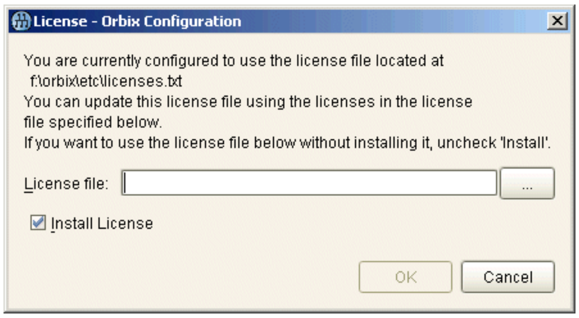


Figure 4: *Orbix License Button*

2. This displays a dialog similar to that shown in Figure 5:



License - Orbix Configuration

You are currently configured to use the license file located at
f:\orbix\etc\licenses.txt
You can update this license file using the licenses in the license
file specified below.
If you want to use the license file below without installing it, uncheck 'Install'.

License file:

[✓] Install License

Figure 5: *Entering the License File*

3. Enter the location of the license file that you wish to install in the **License File** text box. Alternatively, use the **Browse** to navigate to the file. You should have received this file from your Micro Focus representative and stored it in a secure location.

   The default locations are as follows:

Windows	`<`*`install-dir`*`>\etc\licenses.txt`
UNIX	`<`*`install-dir`*`>/etc/opt/iona/licenses.txt`

4. Click **OK** to return to the main screen.

The `licenses.txt` file is copied from your specified location. Any existing license files are overwritten. When you have specified a license file, you will not need to perform these steps again.

Configuring and Deploying a Domain

Orbix provides a GUI-based configuration tool to guide you through generating an Orbix environment.

Introduction

A configuration domain contains all the configuration information used by Orbix ORBs, services, and applications. The Orbix Configuration tool configures and deploys Orbix components into a configuration domain. It can also link a machine to an existing configuration domain, or make updates to an existing domain.

Centralized domain design

The Orbix Configuration tool provides a centralized mechanism for designing a distributed configuration domain. While designing your domain, you specify all of the machines that are to host services in your domain, which services are run on each machine, and which machines, if any, host replicas. You can also deploy location services onto machines that host custom servers.

When you have designed your configuration, you must then go to each machine in the domain and deploy the configuration. This populates each machine's configuration databases and correctly deploys the services on each machine.

Configuration setup options

The Orbix Configuration tool **File**>**New** menu provides the following setup options:

- **Standard**: This enables you to create a new configuration domain from scratch. It is used to determine the type of configuration being created, what ports the core services use, and what services will be deployed into the domain.
- **Expert**: This enables you to create a new configuration domain from scratch. It is similar to using **Standard**, but it provides access to advanced configuration options. This option is recommended if you are familiar with Orbix administration.
- **Deploy**: This enables you to load and deploy a previously created deployment descriptor, to deploy services on the host machines in a domain, and to deploy replicated services in a domain. For more information, see "Replicating Services in a Domain".
- **Open**: This enables you to open an existing configuration domain, and make incremental changes as necessary. For example, you may wish to add additional services to an existing domain, or create replica servers.

- **Connect**: This enables you to connect a client machine to an existing configuration domain. The new machine will link to the existing configuration repository to retrieve its configuration information.

  Note: This option fails to create a domain if the configuration repository is not running, or if the domain is file based.

For details of other tasks that you can perform using the Orbix Configuration tool, see "Runtime management tasks".

Creating a New Domain

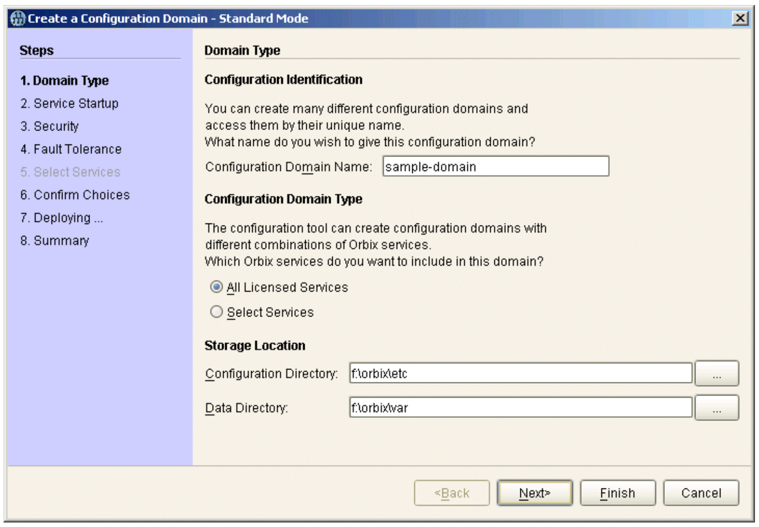
The Orbix Configuration tool's **File**>**New**>**Standard** menu option enables you to create a new configuration domain, or modify an existing one, by walking you through the procedure and providing basic configuration options.

For more advanced configuration options, use the **File**>**New**>**Expert** option (explained in the next section).

Procedure

To create a configuration domain, follow these steps:

1. Start the Orbix Configuration tool using the `itconfigure` command (see "Getting Started").
2. From the main menu, select **File**>**New>Standard**. This displays a screen similar to Figure 6.



Configuration Identification

You can create many different configuration domains and access them by their unique name.
What name do you wish to give this configuration domain?

Configuration Domain Name: sample-domain

Configuration Domain Type

The configuration tool can create configuration domains with different combinations of Orbix services.
Which Orbix services do you want to include in this domain?

[✓ All Licensed Services]
Select Services

Storage Location

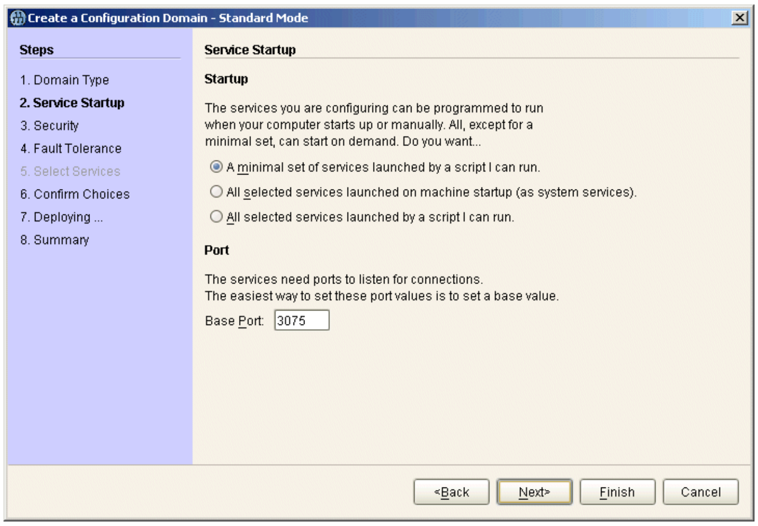
Configuration Directory: f:\orbix\etc
Data Directory: f:\orbix\var

Figure 6: *Domain Type Screen*

3. Specify the domain name. If you are creating a new domain, this name must be unique among any pre-deployed configuration domains. If it is not, the existing domain is overwritten.

Note: On AIX, if you wish Orbix services to be launched on start up, the domain name must be no longer than seven characters.

4. Set the level of services to deploy into the domain by selecting one of the following options:
   - **All Licensed Services** automatically deploys all services for which you have purchased licenses.
   - **Select Services** enables you to select which services you wish to deploy into the domain on the particular machine.
5. Specify the directories where you would like configuration data stored on this system. In most cases, the defaults are sufficient.
6. Click **Next** to select how your services start. This displays a screen similar to Figure 7.


Create a Configuration Domain - Standard Mode

Steps
1. Domain Type
2. Service Startup
3. Security
4. Fault Tolerance
5. Select Services
6. Confirm Choices
7. Deploying ...
8. Summary

Service Startup

Startup

The services you are configuring can be programmed to run when your computer starts up or manually. All, except for a minimal set, can start on demand. Do you want...

[✓ A minimal set of services launched by a script I can run.]
All selected services launched on machine startup (as system services).
All selected services launched by a script I can run.

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

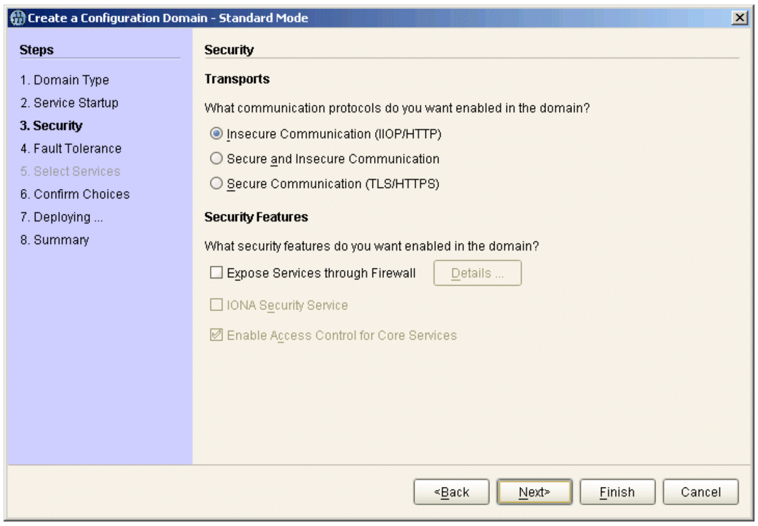
Figure 7: *Startup Mode and Base Port*

7. Choose one of the following options:
   - **A minimal set of services launched by a script I can run** generates a script that to start the location service and, if it selected, the configuration repository. All other deployed services will be started on demand.

Note: When the proceeding options are selected, the location service is deployed by default. You will not be able to unselect it.

- **A minimal set of services launched at machine startup** configures the location service and, if selected, the configuration repository to start up when the machine is booted. All other deployed services will be started on demand.
- **All services launched by a script I can run** generates a script that will start all deployed services.

8. Enter a number for the **Base Port**. This is the number from which Orbix begins sequentially assigning listener ports for its services. The default is `3075`.
9. Click **Next** to configure your domain security features. This displays a screen similar to Figure 8.


Create a Configuration Domain - Standard Mode

Steps
1. Domain Type
2. Service Startup
3. Security
4. Fault Tolerance
5. Select Services
6. Confirm Choices
7. Deploying ...
8. Summary

Security

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
☐ IONA Security Service
☑ Enable Access Control for Core Services

Figure 8: *Setting Security Features*

10. Select a protocol:
    - **Insecure communication (IIOP/HTTP)** configures your domain so that it does not use TLS or HTTPS protocols. It rejects any attempts to make a secure connection.

Note: This is the only mode in which the Firewall Proxy Service will run.

- **Secure communication (TLS/HTTPS)** configures your system so that all communication is done securely. Any attempts to make a connection using a protocol other than TLS or HTTPS are be rejected.
- **Secure and insecure communication** configures your system so that it can use IIOP, TLS, HTTP, and HTPS protocols.

Note: This option is automatically selected if you configure the IS2 Security Infastructure. You can only select secure communication.

11. Select the security features you wish to enable in the domain:

- **Expose services through Firewall** configures your domain to use the firewall proxy service.

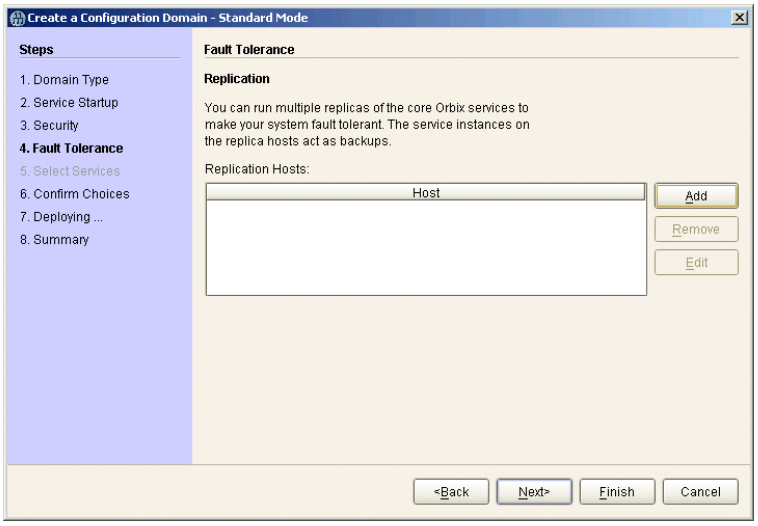
Note: This option is only available for insecure domains.

- **Security Service** configures your domain to take advantages of the ORBIX security platform. For more information read the *Security Guide*.

Note: This option forces you to use TLS and HTTPS. Therefore the firewall proxy service is unavailable.

- **Enable Access Control for Core Services** is only available for use when the IS2 security infrastructure is configured, For more information read the *Security Guide*.

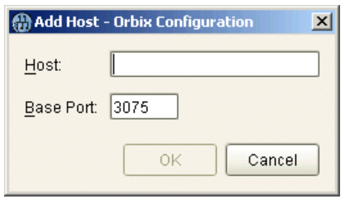
12. Click **Next** to configure any replicas you wish to include in your domain. This displays a screen similar to Figure 9.



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.

Figure 9: *Replica Configuration*

13. To add a replica to the domain, click **Add**, and enter the machine's host name and a listener port in the **Add Host** dialog, shown in Figure 10.



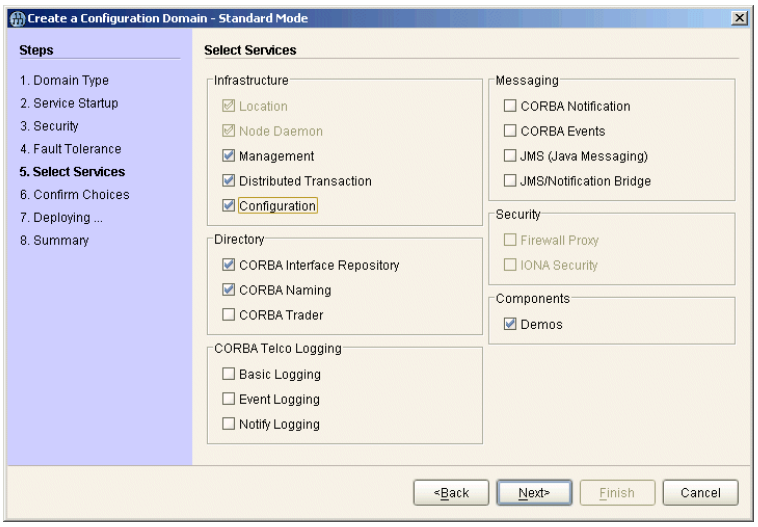
Host:

Base Port: 3075

Figure 10: *Add Host Dialog*

To remove a replica from the list, highlight its hostname and click **Remove**. When you have specified all of the replicas for your domain, click **Next**.

14. If you chose to deploy only selected services, you will see a screen similar to Figure 11.



Steps

1. Domain Type
2. Service Startup
3. Security
4. Fault Tolerance
5. **Select Services**
6. Confirm Choices
7. Deploying ...
8. Summary

Select Services

Infrastructure
- ✓ Location
- ✓ Node Daemon
- ✓ Management
- ✓ Distributed Transaction
- ✓ Configuration

Directory
- ✓ CORBA Interface Repository
- ✓ CORBA Naming
- ☐ CORBA Trader

CORBA Telco Logging
- ☐ Basic Logging
- ☐ Event Logging
- ☐ Notify Logging

Messaging
- ☐ CORBA Notification
- ☐ CORBA Events
- ☐ JMS (Java Messaging)
- ☐ JMS/Notification Bridge

Security
- ☐ Firewall Proxy
- ☐ IONA Security

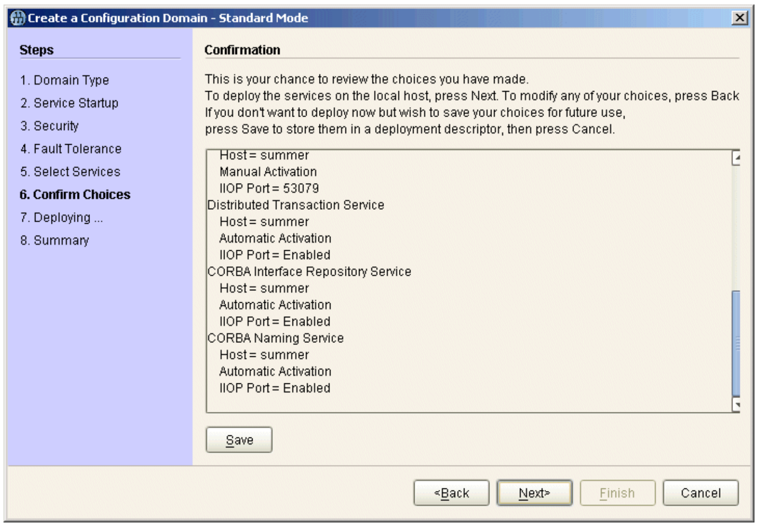
Components
- ✓ Demos

Figure 11: *Selecting Services to Deploy*

Note: You must check **Demos** if you wish to use the demo programs provided with Orbix.

If you chose to deploy all licensed services, go to step 16.

15. Select the services you wish deployed into your configuration. When you have selected the desired services, click **Next** to see a summary of the configuration options you have chosen. This displays a screen similar to Figure 12.



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 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.

```
   Host = summer
   Manual Activation
   IIOP Port = 53079
Distributed Transaction Service
   Host = summer
   Automatic Activation
   IIOP Port = Enabled
CORBA Interface Repository Service
   Host = summer
   Automatic Activation
   IIOP Port = Enabled
CORBA Naming Service
   Host = summer
   Automatic Activation
   IIOP Port = Enabled
```

Figure 12: *Confirmation Screen*

16. If you have configured replicas for this domain, or have configured services to run on a different host, you should save the domain descriptor. To save a domain descriptor for this domain, click **Save**.

Note: The name of the domain descriptor must end in "`_dd`".

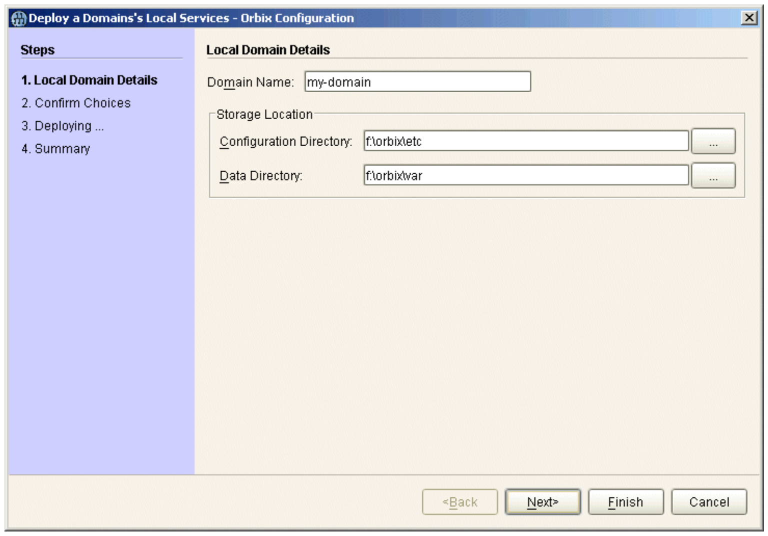
17. Click **Next** to create the domain and deploy the local services.
18. When the domain is successfully created, the **Finish** button becomes enabled. Click it to close the tool.

Deploying a Distributed Domain

When you have designed a distributed domain, you must deploy the domain on all of the hosts that make up the domain. To do this, you must take the deployment descriptor created when you designed the domain and migrate it to each host machine.

The Orbix Configuration tool provides the following options for deploying your domain on the remaining hosts:

- Use the **File**>**Deploy** option from the GUI main menu.
- Use the `-load` and `-nogui` command-line options.



Domain Name: my-domain

Storage Location

Configuration Directory: f:\orbix\etc

Data Directory: f:\orbix\var

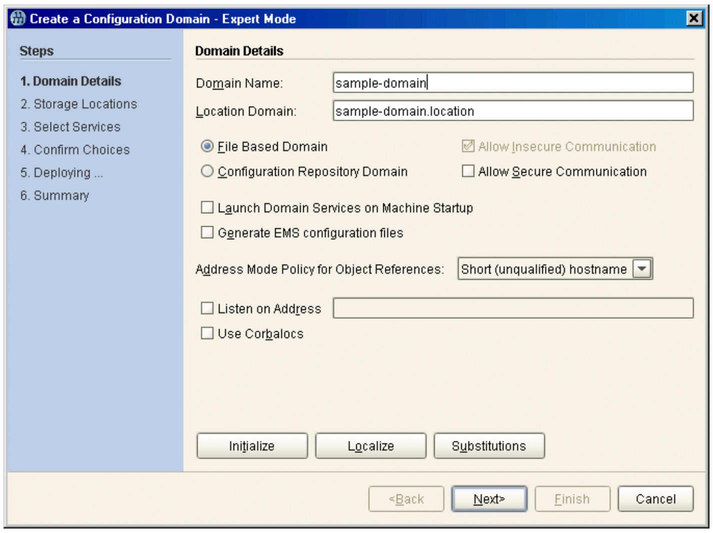
Figure 13: *Deploying a Domain*

Using the Deploy option

The simplest way to deploy the local part of a domain is to use the **Deploy** option. To use this option, perform the following steps:

1. Select **File**>**Deploy** from the main menu.
2. Select the deployment descriptor from the file selection dialog, and click **Open**.
3. A screen similar to Figure 13 on page 21 should appear. Enter the location for the configuration databases to be stored, verify the domain name, and click **Next**.

4. Verify that the configuration details displayed in the **Confirmation** screen are accurate. If so, click **Next** to deploy the local services.
5. When the domain has successfully deployed, click **Finish** to exit.


Create a Configuration Domain - Expert Mode

Steps
1. Domain Details
2. Storage Locations
3. Select Services
4. Confirm Choices
5. Deploying ...
6. Summary

Domain Details

Domain Name: sample-domain
Location Domain: sample-domain.location

● File Based Domain
○ Configuration Repository Domain

[✓] Allow Insecure Communication
[ ] Allow Secure Communication

[ ] Launch Domain Services on Machine Startup
[ ] Generate EMS configuration files

Address Mode Policy for Object References: Short (unqualified) hostname

[ ] Listen on Address
[ ] Use Corbalocs

Figure 14: *Initializing a Domain*

Using the command line

If you can not or do not want to run the Orbix Configuration GUI, you can deploy a your domain on the local host using the following command line syntax:

```
itconfigure -nogui -load deployment-descriptor
```

This command deploys the specified domain and the services for the local host.

Connecting a Client Machine to a Domain

You may often need to configure machines into a domain that only run client programs. These client programs do not need to run any CORBA services, however, they must access the domain's configuration. The Orbix Configuration tool enables you to connect a new machine to an existing configuration domain. The new machine retrieves and stores its configuration in the configuration repository on the existing host machine.

Note: The **Connect** wizard does not enable you to deploy additional services on a machine. It only generates scripts that enable the current machine to join an existing configuration.

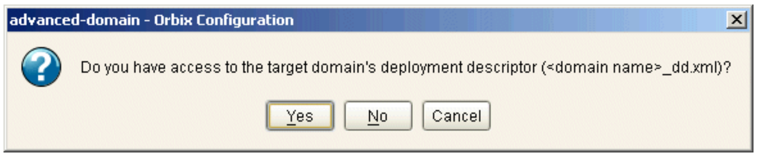
There are two approaches to connecting a client machine to an existing domain:

- "Connecting with a deployment descriptor".
- "Connecting without a deployment descriptor".

Connecting with a deployment descriptor

To connect a new machine to an existing domain using its deployment descriptor file, perform the following steps:

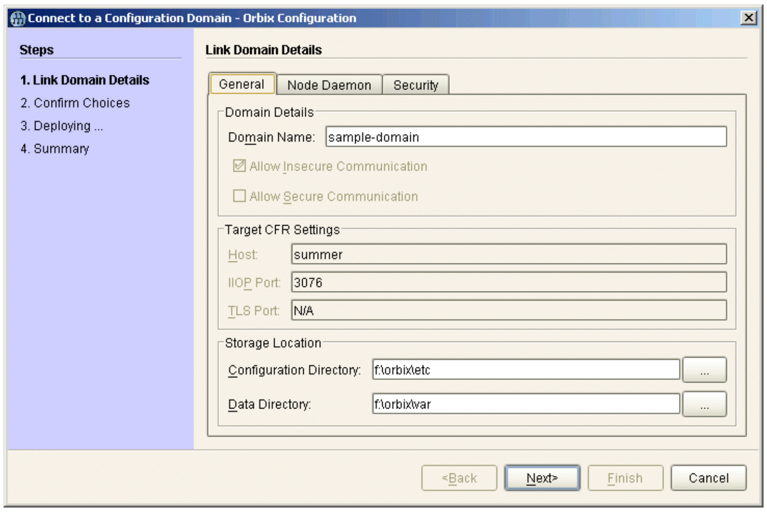
1. Select **File**>**Connect** from the GUI main menu.
2. If you have access to a deployment descriptor, select **Yes** in the dialog shown in Figure 15.


advanced-domain - Orbix Configuration

Do you have access to the target domain's deployment descriptor (<domain name>_dd.xml)?

Figure 15: *Target Domain Dialog.*

3. Select the deployment descriptor from the file selection dialog, and click **Open**.
4. In the **Connect to a Configuration Domain** wizard, enter the details for your link domain. For example, you can specify **General** details such as the location of your configuration files, and **Node Daemon** or **Security** details, if applicable. This wizard is shown in Figure 16.


Connect to a Configuration Domain - Orbix Configuration

Steps
1. Link Domain Details
2. Confirm Choices
3. Deploying ...
4. Summary

Link Domain Details

General | Node Daemon | Security

Domain Details
Domain Name: sample-domain
☑ Allow Insecure Communication
☐ Allow Secure Communication

Target CFR Settings
Host: summer
IIOP Port: 3076
TLS Port: N/A

Storage Location
Configuration Directory: f:\orbix\etc
Data Directory: f:\orbix\var

Figure 16: *Connecting to a Domain*

5. Click **Next** to confirm your input and view a summary of the configuration in the **Confirmation** screen.
6. Click **Next** to create the local files needed to connect the machine to the configuration domain and deploy the local services.

7. When the machine is successfully connected to the domain, in the **Summary** screen, select **Finish**.

Connecting without a deployment descriptor

Users without access to a a deployment descriptor can also connect a new machine using the **Connect to a Configuration Domain** wizard. To connect this way, perform the following steps:

1. Select **File**>**Connect** from the GUI main menu.
2. Because you do not have access to a deployment descriptor, select **No** in the dialog shown in Figure 15.
3. In the **Connect to a Configuration Domain** wizard, enter the hostname and port of the CFR to which you wish to connect the new machine. If applicable, enter a location for the configuration files, and **Node Daemon** or **Security** details. The wizard is shown in Figure 16.
4. Click **OK** to confirm your input and view a summary of the configuration.
5. Click **Next** to create the local files needed to connect the machine to the configuration domain and deploy the local services.
6. When the machine is successfully connected to the domain, in the **Summary** screen, select **Finish**.

Localizing a Preconfigured Domain

You may need to create a duplicate configuration, one that is identical, except for the hosts that it runs on. Some reasons for doing this are:

- Creating test and production configurations that are identical in everything but the host on which they run.
- Migrating a system from one machine to another.
- Packaging an Orbix installation with a software distribution. You can then ship a configuration template that can be run on each destination machine, with the services localized for that host, rather than the host on which the configuration was created.

If you wish to deploy a preconfigured domain, the Orbix Configuration tool provides two options:

- Use the GUI in Expert mode, and select **Localize**.
- Run the `itconfigure` command with the `-localize` and `-nogui` options.

Using the GUI

To use the Orbix Configuration GUI to deploy a localized domain, complete the following steps:

1. Select **File**>**New**>**Expert** from the main menu.
2. In the screen shown in Figure 14 on page 22, click **Initialize**, located at the bottom left of the screen.
3. Select the preconfigured domain descriptor from the file selection dialog.
4. Click **Localize**. This replaces the name of the host defined in the original configuration with the local host name.
5. Click **Next**.
6. Click **Finish**.

   Alternatively, you can make changes to suit the needs of your environment (see "Deploying a Distributed Domain").

Using the command line

If you can not or do not wish to run the GUI, you can deploy a localized domain from the command line by running:

```
itconfigure -nogui -localize [-from host] -load
   deployment-descriptor
```

Running this command first replaces a deploy node in the descriptor with the host specified in the `-from` *host* option or with the local host. It then deploys an exact replica of the specified domain on the new host.

The `-from` *host* option enables you to select the deploy node to replace when localizing a multi-profile descriptor.

You can specify other changes to the deployed domain by using other command-line options.

Replicating Services in a Domain

You can use the Orbix Configuration tool to configure a machine to use replicas of an existing CFR, locator, naming, and security service. A machine configured to host replicas can also host services as part of an existing configuration domain.

Note: To configure a machine to host replica services, you should have already specified that the domain include replicas when you created it (Figure 9).

Deploying a CFR-based replica

To deploy a CFR-based replica, perform the following steps:

1. Copy the generated deployment descriptor from the host machine to the replica machine that you wish to configure. The deployment descriptor name is *domain_name*`_dd.xml`. For example, the domain descriptor for a domain named `Apollo` will be `Apollo_dd.xml`.
2. Run the **Orbix Configuration** tool. From the main menu, select **File**>**Deploy**.
3. In the **Load Descriptor** dialog, select the domain descriptor that you wish to replicate. Click **Open**.
4. A screen similar to Figure 13 on page 21 should appear. Enter the location for the configuration databases to be stored, verify the domain name, and click **Next**.
5. Verify that the configuration details displayed in the **Confirmation** screen are accurate. If so, click **Next** to deploy the local services.
6. When the replica is successfully deployed, click **Finish**.

Updating an Existing Domain

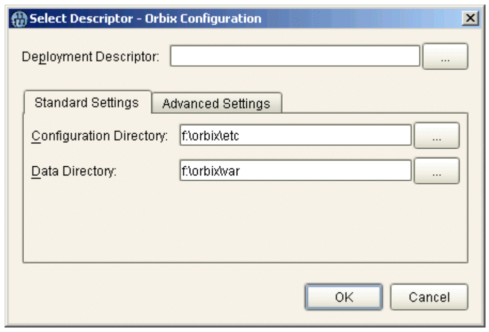
The Orbix Configuration tool enables you to perform dynamic updates on an existing configuration domain (for example, add services or replica services after the domain is created). You can use the **File**>**Open** option to specify an existing domain descriptor file. Alternatively, use the **File**>**Reopen** option to access the list of recently used domains. This section includes the following:

- “Opening a domain”
- “Reopening a domain”
- “Adding a service”
- “Adding a replica service”
- “Repreparing a service”
- “Removing a replica service”

Opening a domain

To open an existing domain, perform the following steps:

1. Select **File**>**Open** from the main menu.
2. Select the deployment descriptor using the **Select Descriptor** dialog, shown in Figure 17.

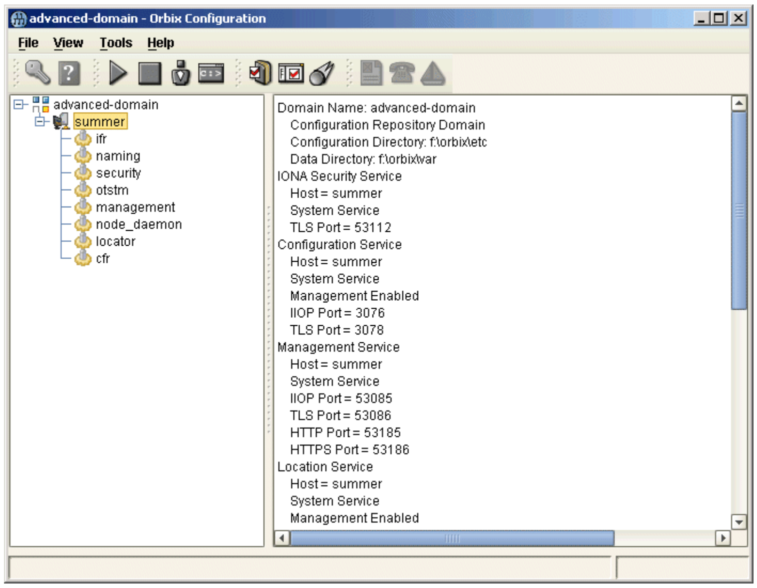


Configuration Directory: f:\orbix\etc
Data Directory: f:\orbix\var

Figure 17: *Select Descriptor Dialog*

This dialog also enables you to specify **Standard Settings** and **Advanced Settings** for domains with non-default locations.

3. Click **OK**. This loads up the domain and displays summary information in the GUI, shown in Figure 18.



```
Domain Name: advanced-domain
   Configuration Repository Domain
   Configuration Directory: f:\orbix\etc
   Data Directory: f:\orbix\var
IONA Security Service
   Host = summer
   System Service
   TLS Port = 53112
Configuration Service
   Host = summer
   System Service
   Management Enabled
   IIOP Port = 3076
   TLS Port = 3078
Management Service
   Host = summer
   System Service
   IIOP Port = 53085
   TLS Port = 53086
   HTTP Port = 53185
   HTTPS Port = 53186
Location Service
   Host = summer
   System Service
   Management Enabled
```

Figure 18: *Loaded Domain*

When the domain has been opened, you can then start or stop its services, perform dynamic updates (for example, add a service or replica service), or launch consoles for that domain.

Note: In CFR or link domains, the effective deployment descriptor for the domain is obtained from the CFR itself. Therefore, to open a CFR domain, at least one CFR replica must be running.

Reopening a domain

The **Reopen** option enables you to select a domain from a list of recently used domains, without having to specify a domain descriptor.

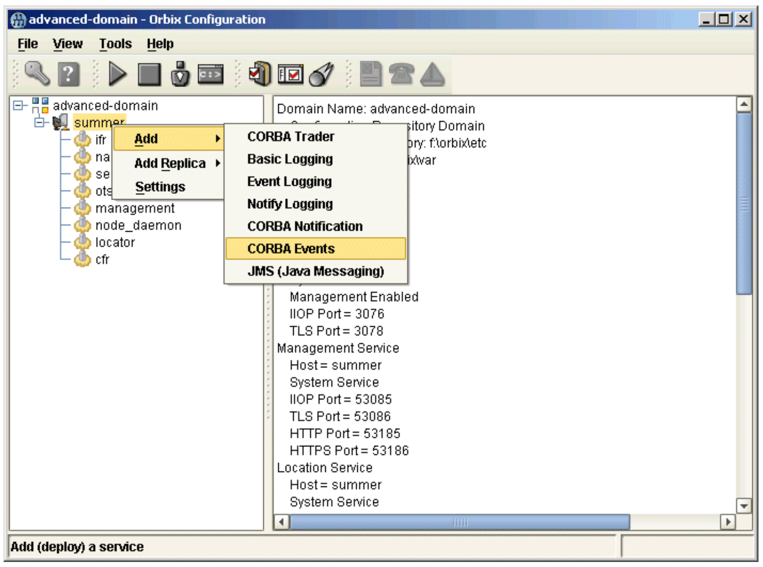
To use this option, select **File**>**Reopen** from the main menu, and select the domain that you wish to reopen.

Note: The **Reopen** option is only available if you are using JDK version 1.4.x or later. Otherwise, this option is not displayed.

Adding a service

To add a service to an existing domain, perform the following steps,

1. Select the machine node on which you want to add the service.
2. Right-click to select **Add**, and select the service that you require. Figure 18 shows an example of adding a CORBA Events service.



```
Domain Name: advanced-domain
                       sitory Domain
                     ory: f:\orbix\etc
                     ix\var

   Management Enabled
   IIOP Port = 3076
   TLS Port = 3078
Management Service
   Host = summer
   System Service
   IIOP Port = 53085
   TLS Port = 53086
   HTTP Port = 53185
   HTTPS Port = 53186
Location Service
   Host = summer
   System Service
```

Figure 19: *Adding a Service to a Domain*

The domain services must be running, and you can only select from services that are not already in the domain. You can only add services on the host that the **Orbix Configuration** tool is run. To extend a domain to include another host, create a link domain (see "Connecting a Client Machine to a Domain").

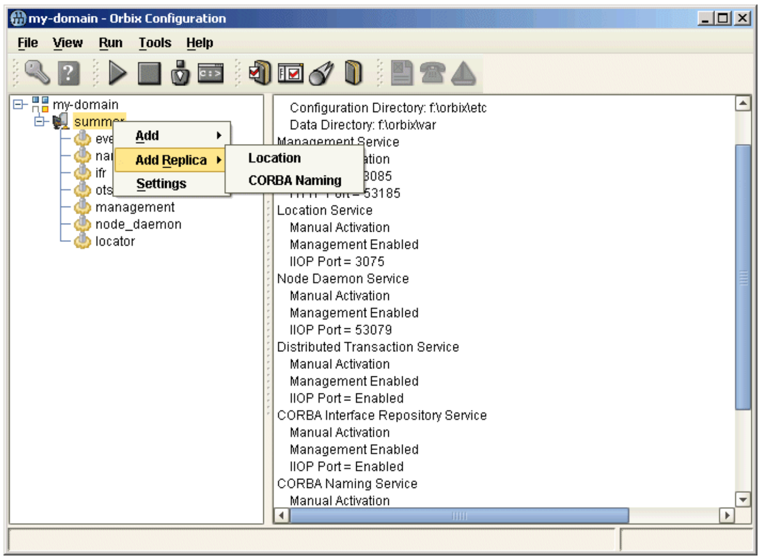
When you add a new service, the local domain scripts, configuration files, and log files are updated with details of the new service. The domain scripts include the following:

- *domain-name*_env
- start_*domain-name*_services
- stop_*domain-name*_services

Adding a replica service

To add a replica service to an existing domain, perform the following steps:

1. Select the machine node in the left pane, right click to select **Add**, and select the replica service that you require. Figure 20 shows an example of adding a replica locator.



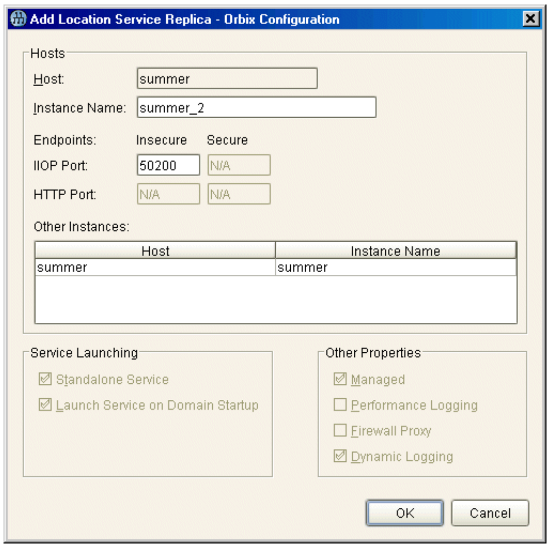
```
   Configuration Directory: f:\orbix\etc
   Data Directory: f:\orbix\var
Management Service
          ation
          3085
          53185
Location Service
   Manual Activation
   Management Enabled
   IIOP Port = 3075
Node Daemon Service
   Manual Activation
   Management Enabled
   IIOP Port = 53079
Distributed Transaction Service
   Manual Activation
   Management Enabled
   IIOP Port = Enabled
CORBA Interface Repository Service
   Manual Activation
   Management Enabled
   IIOP Port = Enabled
CORBA Naming Service
   Manual Activation
```

Figure 20: *Adding a Replica Service to a Domain*

The domain services must be running, and you can only select from services that are not already in the domain. You can only add services on the host that the **Orbix Configuration** tool is run. To extend a domain to include another host, create a link domain (see "Connecting a Client Machine to a Domain" on page 22

Note: You can replicate the CFR, locator, and naming services using the **Orbix Configuration** tool. Security replicas must be configured manually (see the *Security Guide* for details).

2. After selecting the replica service, you can specify details such as the **Instance Name** and ports for your replica service. Figure 21 shows an example of the **Add Location Service Replica** dialog.


Add Location Service Replica - Orbix Configuration

Hosts

Host:	summer	
Instance Name:	summer_2	
Endpoints:	Insecure	Secure
IIOP Port:	50200	N/A
HTTP Port:	N/A	N/A

Other Instances:

Host	Instance Name
summer	summer

Service Launching
- [✓] Standalone Service
- [✓] Launch Service on Domain Startup

Other Properties
- [✓] Managed
- [ ] Performance Logging
- [ ] Firewall Proxy
- [✓] Dynamic Logging

Figure 21: *Add Location Service Replica dialog*

3. After adding the replica, you may be prompted to perform additional steps. For example, after adding a replica locator, you must reprepare any indirect persistent services. This ensures that the object references for these services include the address of the locator replica, and enables clients to be correctly directed to these services.

Removing a replica service

To permanently remove a replica service from a domain, perform the following steps:

1. Select the service in the left pane, and right click to select **Remove**. When removing a replica, it must be running in the domain.

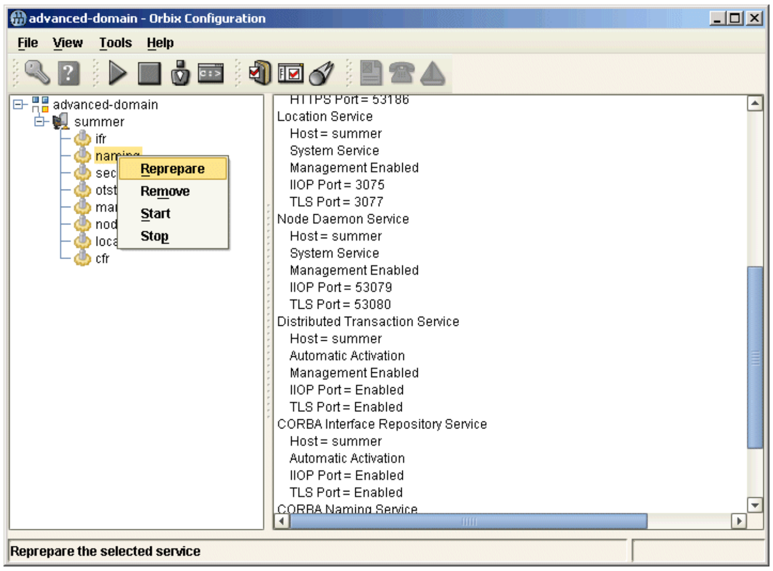
   This action removes details of the service from the domain scripts (for example, `start_my-domain_services`).
2. After removing a replica, you may be prompted to perform additional steps. For example, after removing a replica locator you must reprepare any indirect persistent services. This ensures that their object references do not include the address of the removed locator.

Repreparing a service

After creating or removing a replica locator, you must first reprepare any indirect persistent services. This ensures that the object references for these services include the correct locator address information.

For example, when creating a locator replica, an indirect persistent service must be reprepared to ensure that its object reference includes the address of the locator replica. When removing a locator replica, repreparing ensures that it no longer includes the address of the removed locator.

To reprepare a service, select the service in the left pane, and right click to select **Reprepare**. Figure 18 shows an example.



```
   HTTPS Port = 53186
Location Service
   Host = summer
   System Service
   Management Enabled
   IIOP Port = 3075
   TLS Port = 3077
Node Daemon Service
   Host = summer
   System Service
   Management Enabled
   IIOP Port = 53079
   TLS Port = 53080
Distributed Transaction Service
   Host = summer
   Automatic Activation
   Management Enabled
   IIOP Port = Enabled
   TLS Port = Enabled
CORBA Interface Repository Service
   Host = summer
   Automatic Activation
   IIOP Port = Enabled
   TLS Port = Enabled
CORBA Naming Service
```

Reprepare the selected service

Figure 22: *Repreparing a Service*

Starting and Stopping Orbix Services

The Orbix Configuration tool automatically generates start and stop scripts. These enable you to manually activate and deactivate all services deployed on the configured host. You can activate these scripts directly in the GUI or using the command line.

Starting Orbix domain services

To start all services for the current domain that are deployed on this machine, select **Run**>**Start** from the main menu, or click the **Start** button in the toolbar.

Alternatively, you can start all domain services using the following command:

```
config-dir/bin/start_domain-name_services
```

Stopping Orbix domain services

To start all services for the current domain that are deployed on this machine, select **Run**>**Stop** from the main menu, or click the **Stop** button in the toolbar.

Alternatively, you can stop all domain services using the following command:

```
config-dir/bin/stop_domain-name_services
```

Starting and stopping individual services

To start or stop an individual service, select the service in the left pane, right click, and select **Start** or **Stop**.

For details of starting and stopping individual services on the command line, see the *Orbix Administrator's Guide*.

Setting your environment in a command shell

To set your command shell to recognize a specific domain, select **Run**>**Command Shell**.

Alternatively, you can set your environment using the following command:

```
config-dir/bin/domain-name_env
```

Setting Java ORB Classes

To run Java applications, Orbix must use its own ORB classes instead of Sun ORB classes. You can ensure that Orbix finds the correct classes one of the following ways:

- Use an `iona.properties` file.
- Use Java system properties when invoking the Java interpreter.

Using an iona.properties file

You can create an `iona.properties` file in the *`JAVA_HOME`*`/jre/lib` directory. This file should contain the following settings:

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

Note: The **Orbix Configuration** tool automatically creates an `iona.properties` file in the *`install-dir`*`\etc\domains\`*`domain-name`* directory. This file is added to the `CLASSPATH` set by your domain's initialization script.

Using Java system properties

You can invoke the Java interpreter with the `-D` options as follows:

```
java
   -Dorg.omg.CORBA.ORBCLASS=com.iona.corba.art.artimpl.ORBImpl
-Dorg.omg.CORBA.ORBSingletonClass=com.iona.corba.art.artimpl.OR
   BSingleton app-name
```

Advanced Configuration and Deployment

This chapter explains advanced custom configuration and deployment features offered by Orbix.

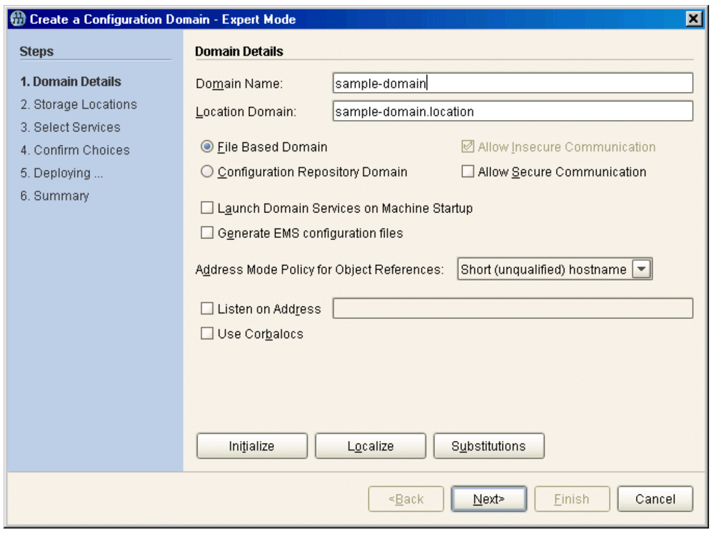
Creating a Domain in Expert Mode

Expert mode provides advanced users a more flexibility when creating and modifying configuration domains. It enables you to specify well-known addresses for Orbix services, and also to configure the services to run using direct or indirect persistence.

Procedure

To create a configuration domain using expert mode, complete the following steps:

1. From the main menu, select **File**>**New**>**Expert**.
2. In the **Domain Details** screen, enter a name for the domain and specify if the domain is to be file-based or CFR-based, shown in Figure 23.



Domain Details

Domain Name: sample-domain

Location Domain: sample-domain.location

[✓ File Based Domain] Configuration Repository Domain

[✓] Allow Insecure Communication

[ ] Allow Secure Communication

[ ] Launch Domain Services on Machine Startup

[ ] Generate EMS configuration files

Address Mode Policy for Object References: Short (unqualified) hostname

[ ] Listen on Address

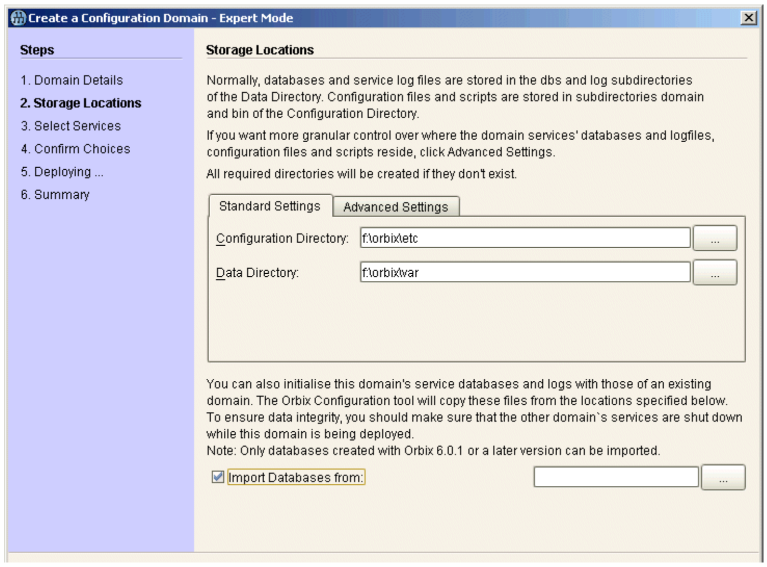
[ ] Use Corbalocs

Figure 23: *Domain Details Screen*

3. Select the level of security for your domain:
   - **Allow Insecure Communication** configures your domain to allow communication over insecure protocols such as HTTP.
   - **Allow Secure Communication** configures your system to allow secure communication using TLS or HTTPS.
4. To have the domain be started on system start-up place a check next to **Launch Domain Services on Machine Startup**.
5. If you wish to integrate your domain into an Enterprise Management System (for example, IBM Tivoli, BMC Patrol, or HP Openview), check **Generate EMS configuration files**.
6. Use the **Address Mode Policy for Object References** drop-down box to select how services publish hostnames in their object references. Select one of the following options:
   - `Short (unqualified) hostname`
   - `Fully qualified hostname`
   - `localhost (the default)`
   - `IP Addresss`
   - `127.0.0.1`
7. Click **Next** to specify the location of the files associated with your domain. The **Storage Locations** dialog is displayed, shown in Figure 24.
8. If you wish to store your configuration and logging information in non-default locations, you can specify these using the **Standard Settings** and **Advanced Settings**.

   Normally, databases and service log files are stored in the `db` and `log` directories of the **Data Directory**. Configuration files and scripts are stored in `domain` and `bin` directories of the **Configuration Directory**.

   If you want more fine-grained control over where the domain service database and log files, and configuration scripts and log files reside, specify the **Advanced Settings**. All required directories are created if they do not already exist.


Create a Configuration Domain - Expert Mode

Steps

1. Domain Details
2. Storage Locations
3. Select Services
4. Confirm Choices
5. Deploying ...
6. Summary

Storage Locations

Normally, databases and service log files are stored in the dbs and log subdirectories of the Data Directory. Configuration files and scripts are stored in subdirectories domain and bin of the Configuration Directory.

If you want more granular control over where the domain services' databases and logfiles, configuration files and scripts reside, click Advanced Settings.

All required directories will be created if they don't exist.

Standard Settings | Advanced Settings

Configuration Directory: f:\orbix\etc

Data Directory: f:\orbix\var

You can also initialise this domain's service databases and logs with those of an existing domain. The Orbix Configuration tool will copy these files from the locations specified below. To ensure data integrity, you should make sure that the other domain`s services are shut down while this domain is being deployed.
Note: Only databases created with Orbix 6.0.1 or a later version can be imported.

[✓] Import Databases from:

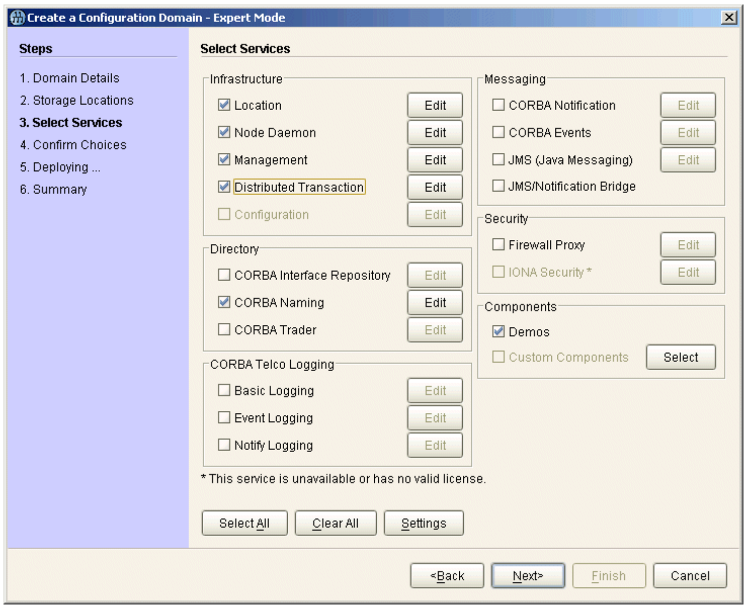
Figure 24: *Storage Locations Screen*

9. You can also initialize your domain's service database and log files with those of an existing domain. These domain files are copied from the locations specified by the **Import Databases from** field.

Note: To ensure data integrity, ensure that the other domain's services are shut down when this domain is being deployed. Only databases created with Orbix 6 SP 1, or later, can be imported.

10. Click **Next** to select the services to deploy into the domain. The **Select Services** screen is displayed, shown in Figure 25.

11. In the **Select Services** screen, check the services that you wish to deploy into the domain.


Create a Configuration Domain - Expert Mode

Steps
1. Domain Details
2. Storage Locations
3. **Select Services**
4. Confirm Choices
5. Deploying ...
6. Summary

Select Services

Infrastructure
- ☑ Location — Edit
- ☑ Node Daemon — Edit
- ☑ Management — Edit
- ☑ Distributed Transaction — Edit
- ☐ Configuration — Edit

Directory
- ☐ CORBA Interface Repository — Edit
- ☑ CORBA Naming — Edit
- ☐ CORBA Trader — Edit

CORBA Telco Logging
- ☐ Basic Logging — Edit
- ☐ Event Logging — Edit
- ☐ Notify Logging — Edit

Messaging
- ☐ CORBA Notification — Edit
- ☐ CORBA Events — Edit
- ☐ JMS (Java Messaging) — Edit
- ☐ JMS/Notification Bridge

Security
- ☐ Firewall Proxy — Edit
- ☐ IONA Security * — Edit

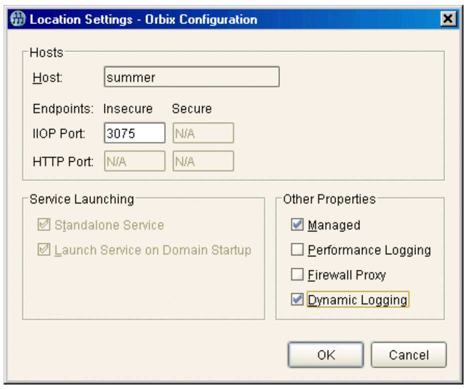
Components
- ☑ Demos
- ☐ Custom Components — Select

* This service is unavailable or has no valid license.

Figure 25: *Select Services Screen*

Note: If you do not check **Demos**, the demo programs included with the installation will not run properly.

12. If you wish to deploy a service using non-default settings, click the **Edit** button next to the service's name. This displays a dialog similar to Figure 26. This enables you to configure options specifically for the service (for example, activation modes, replication settings, and optional properties). When you have selected the settings, click **OK** to return to the **Select Services** screen.


Location Settings - Orbix Configuration

Hosts
Host: summer

Endpoints:	Insecure	Secure
IIOP Port:	3075	N/A
HTTP Port:	N/A	N/A

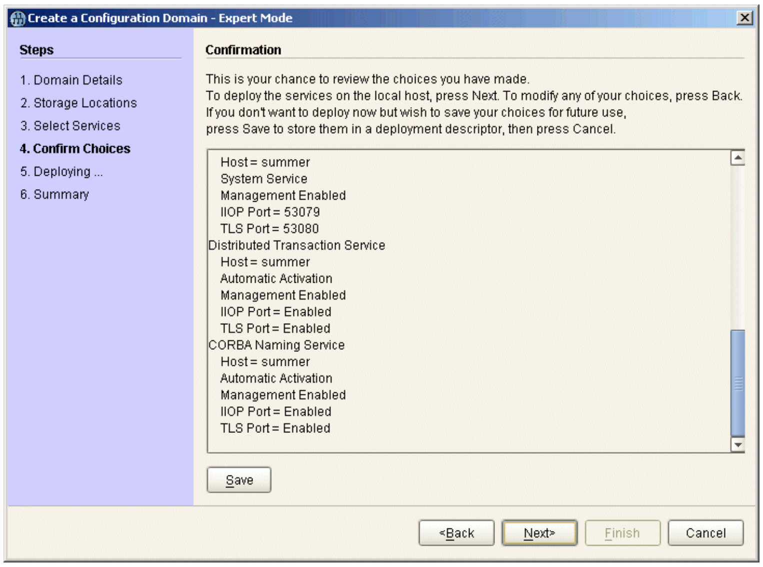
Service Launching
- ☑ Standalone Service
- ☑ Launch Service on Domain Startup

Other Properties
- ☑ Managed
- ☐ Performance Logging
- ☐ Firewall Proxy
- ☑ Dynamic Logging

Figure 26: *Location Settings Dialog*

Note: Some options may not be available for all services (for example, replication is available for the locator, CFR, security and naming services only).

13. After selecting and configuring the desired services, click **Next** to view the configuration options that you have chosen. This displays a **Confirmation** screen similar to Figure 27.
14. If you have configured replicas for this domain or have configured services to be run on different hosts you must save a domain descriptor. To save a descriptor, click **Save**.



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 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.

```
   Host = summer
   System Service
   Management Enabled
   IIOP Port = 53079
   TLS Port = 53080
Distributed Transaction Service
   Host = summer
   Automatic Activation
   Management Enabled
   IIOP Port = Enabled
   TLS Port = Enabled
CORBA Naming Service
   Host = summer
   Automatic Activation
   Management Enabled
   IIOP Port = Enabled
   TLS Port = Enabled
```

Figure 27: *Confirmation Screen*

15. If you want to proceed to deploy, click **Next** to create the domain and deploy any local services.
16. When the domain is successfully created, the **Finish** button becomes enabled. Click **Finish** to exit.

Configuring a Machine with no GUI

You may need to occasionally configure and deploy an Orbix domain on a machine with no GUI capabilities (for example, a server on a remote site). The Orbix Configuration tool supports this by enabling you to create a domain deployment descriptor on one host, and then deploy it on another host.

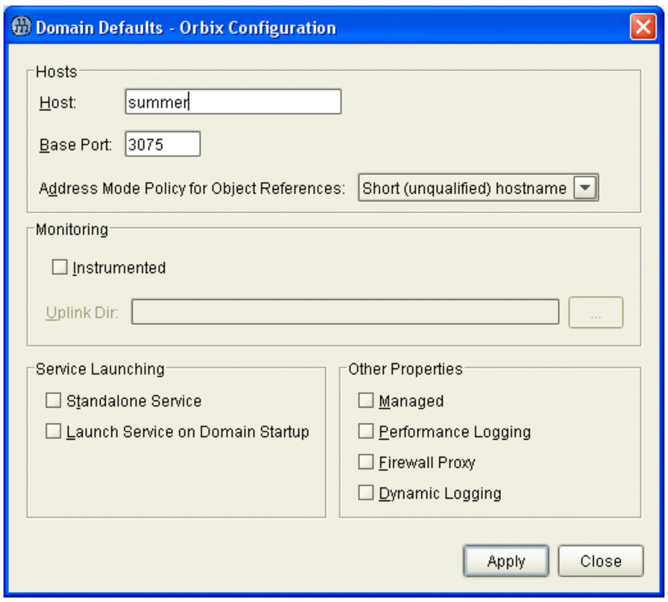
Orbix provides a command-line version of the configuration tool (`itconfigure -nogui`) for users who cannot deploy using a GUI application. This parses a pre-existing deployment descriptor and deploys the specified configuration domain.

Creating the deployment descriptor

The recommended method of generating the deployment descriptor is to first run the Orbix configuration tool on a GUI-enabled machine, and then save the deployment descriptor for later use in command-line mode. This ensures that the generated XML document is valid.

To create the deployment descriptor, complete the following steps:

1. On a machine with GUI capabilities, run the configuration GUI and select **File**>**New**>**Expert** from main menu.
2. To design the domain, follow the steps outlined in "Creating a Domain in Expert Mode".
3. On the **Select Services** screen, click **Settings**. This displays a dialog box similar to the one shown in Figure 28.
4. In the **Host** text box, enter the name of the remote host, and click **Apply**.
5. In the **Confirmation** screen, click **Save** to save the deployment descriptor.


Domain Defaults - Orbix Configuration

Hosts
Host: summer
Base Port: 3075
Address Mode Policy for Object References: Short (unqualified) hostname

Monitoring
☐ Instrumented
Uplink Dir:

Service Launching
☐ Standalone Service
☐ Launch Service on Domain Startup

Other Properties
☐ Managed
☐ Performance Logging
☐ Firewall Proxy
☐ Dynamic Logging

Figure 28: *Domain Defaults Dialog*

Deploying on the command line

To deploy a deployment descriptor on the command line, perform the following steps:

1. Copy your deployment descriptor file to the machine without GUI capabilities.
2. At the command prompt, change directory to the location of your domain deployment descriptor, for example:

```
<install-dir>\etc\domains\sample-domain
```

3. Enter the following command:

```
 itconfigure -load sample-domain_dd.xml -nogui
```

   itconfigure reads the specified deployment descriptor, finds the profile matching the current host's IP address and deploys the services in this profile. If no such match is found, itconfigure prints an information message and exits.
4. Repeat this process on any other hosts for which you have configured services.

Localizing the domain

If the descriptor contains exactly one profile/node, and that node does not match the local host, use the following command:

```
itconfigure -load sample-domain_dd.xml -nogui -localize
```

This replaces the name and IP address of the node specified in the deployment descriptor with name and IP address of the local host.

Changing the domain name

If you wish to change the name of the configuration domain, use following command:

```
itconfigure -load sample-domain_dd.xml -nogui -name my-domain
```

The name specified using -name overrides the name specified in the descriptor.

Note: On AIX, if you wish Orbix services to be launched at start up, the domain name must be no longer than seven characters.

Deploying on Multi-homed Machines

You may need to configure and deploy an Orbix domain on a *multi-homed* server machine. This is a machine that has more than one IP address and corresponding hostname, and where only one address/hostname is configured as the default.

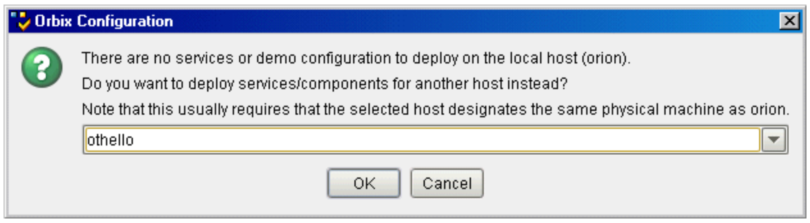
If you want to use a non-default Orbix hostname, you can supply it explicitly to the **Orbix Configuration** tool. This provides a -multihome command-line option, where the alternate hostname can be specified.

Specifying the -multihome option to itconfigure instructs Orbix-supplied services, such as the locator, to use the specified hostname when listening on a *host*:*port* and embedding a host into an IOR.

Multi-homed deployment process

When you click the configuration tool's **Finish** button, the behavior in GUI mode is identical with command-line mode. The configuration tool has a deployment descriptor loaded in memory, and must decide which services to deploy. The selection process is as follows:

- The configuration tool looks for the host node that matches the localhost IP address, using `InetAddress.getLocalHost().getHostAddress()`.
- If a matching host is found, the configuration tool deploys the services specified in this node's profile.
- If no matching host is found, the configuration tool displays a dialog that enables you to deploy on another hostname. For example, Figure 29 shows the dialog displayed when there are more than two nodes.



There are no services or demo configuration to deploy on the local host (orion).
Do you want to deploy services/components for another host instead?
Note that this usually requires that the selected host designates the same physical machine as orion.
othello

Figure 29: *Dialog for More than Two Nodes*

You can choose to deploy the services for one of these nodes. This only succeeds if you are on a multi-homed machine, and the selected node's IP address/name maps to another non-default network interface on this machine. The exception is when there is no interaction between the services on the selected node.

For more information on deployment descriptor nodes and profiles, see "Orbix Deployment Descriptors".

Deploying on the command line

On the command line, you can specify the host to consider as the local node to `itconfigure`. This avoids the step described in "Multi-homed deployment process".

If the deployment descriptor has a `<dd:node>` `name` attribute set to the name of a virtual or additional network adapter (for example, `othello`), you can deploy the services for this node using the following command:

```
itconfigure -nogui -multihome othello -load
    <descriptor-name>
```

The `-multihome` option instructs `itconfigure` to look for a node matching the specified `-multihome` value instead of finding one that matches the localhost's default IP address.

Deploying with the GUI

In GUI mode, you can also use the `-multihome` option when starting the GUI to avoid the step described in "Multi-homed deployment process". When you click **Finish**, `itconfigure` initializes the default host with the one specified by `-multihome`. Unless you manually alter the host field(s), no dialog appears when you click **Finish** because there is no conflict to resolve.

To configure and deploy a domain on a multi-homed machine using the GUI, complete the following steps:

1. On a multi-homed machine, run the configuration tool with the `-multihome` command-line option, and specify the alternate hostname. For example, on a multi-homed machine with primary hostname `orion`, and an alternate hostname `othello`, use the following command:

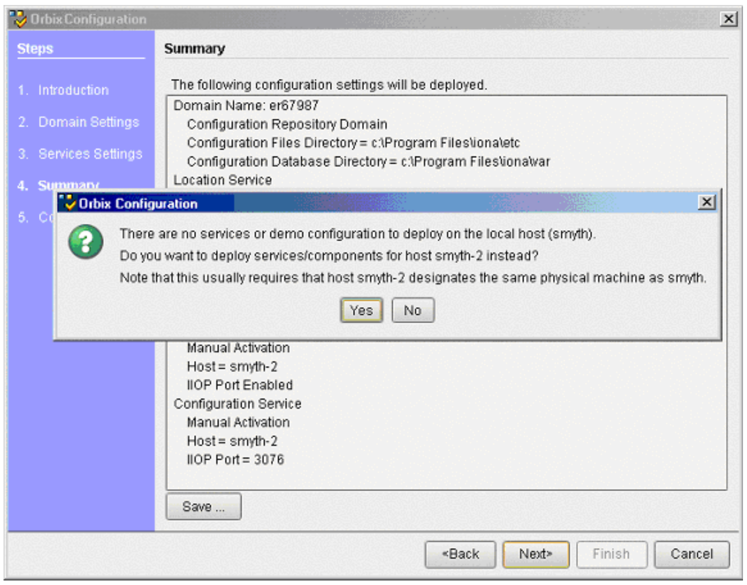
```
itconfigure -multihome othello
```

2. Select **File**>**New**>**Expert** from main menu, and follow the steps outlined in "Creating a Domain in Expert Mode" on page 35.
3. In the **Select Services** dialog, click **Settings**. This displays a dialog box similar to the one shown in Figure 28 on page 40.
4. The **Host** box should match the hostname specified by the multihome parameter on the command line.
5. Click **Apply**.
6. Because the configuration tool has been informed of the alternate hostname, deployment can then proceed as normal (see "Creating a Domain in Expert Mode" on page 35).

Modifying hostnames without the -multihome option

If you do not use the `-multihome` option, and you modify the content of the hosts field, `itconfigure` displays the dialog shown in Figure 30.

If you click **Yes**, the services are deployed. The last sentence in this message shows that this conflict can also arise when the host is a truly remote machine, and forcing local deployment would not make sense.



Summary

The following configuration settings will be deployed.

Domain Name: er67987
Configuration Repository Domain
Configuration Files Directory = c:\Program Files\iona\etc
Configuration Database Directory = c:\Program Files\iona\var
Location Service

Orbix Configuration

There are no services or demo configuration to deploy on the local host (smyth).
Do you want to deploy services/components for host smyth-2 instead?
Note that this usually requires that host smyth-2 designates the same physical machine as smyth.

Manual Activation
Host = smyth-2
IIOP Port Enabled
Configuration Service
Manual Activation
Host = smyth-2
IIOP Port = 3076

Figure 30: *Multi-homed Message*

Configuring your Orbix applications for multi-homed hosts

Specifying the `-multihome` option to `itconfigure` instructs Orbix-supplied services, such as the locator, to use the specified hostname when listening on a *host*:*port*, and embedding a host into an IOR. However, by default, Orbix applications written by the user still use the default hostname.

To use a multi-homed hostname with your Orbix application, you must specify values for the following configuration variables in your application configuration scope:

- `policies:iiop:client_address_mode_policy:local_hostname`
- `policies:iiop:server_address_mode_policy:local_hostname`

For example, the following `itadmin` commands show examples of setting both these variables:

```
itadmin variable create -scope ApplicationScope \
  -type string -value Hostname \
 policies:iiop:client_address_mode_policy:local_hostname

itadmin variable create -scope ApplicationScope \
  -type string -value Hostname \
 policies:iiop:server_address_mode_policy:local_hostname
```

These settings configure applications listening on random ports to use the specified hostname. The most common use case is when the domain is for a single host. In this case, you can create these variables in the global configuration scope.

Configuring Orbix in a Cluster

A cluster is a group of two or more machines, in which the following applies:

- Each machine has its own hostname.
- There is also a virtual hostname/IP address, but this is assigned to only one machine at a time.
- A shared disk drive can be mounted by only one machine at a time

A cluster is a means of providing failover. Software, configuration files, databases, and so on, can be stored on the shared disk. While server applications always advertise themselves using the virtual hostname.

Clustering software is used to monitor the health of software running on the active computer. If something goes wrong, a failover is made to another computer in the cluster. This section explains how to set up a cluster in an Orbix environment.

Configure a multi-homed Orbix host

On the currently active machine in the cluster, install Orbix on the shared disk, and run the following command:

```
itconfigure -multihome virtual-hostname
```

This configures Orbix services such as the locator for use on a multi-homed host. For full details, see "Deploying on Multi-homed Machines" on page 41.

Configure your Orbix applications as multi-homed

To use a multi-homed hostname with Orbix applications you have written, specify values for the following configuration variables in your application configuration scope:

- `policies:iiop:client_address_mode_policy:local_hostname`
- `policies:iiop:server_address_mode_policy:local_hostname`

This instructs your applications listening on random ports to use the specified hostname. For more details, see "Configuring your Orbix applications for multi-homed hosts" on page 44.

Write shell scripts for Orbix services and applications

Write shell scripts that do the following:

- Start Orbix services and user-written Orbix server applications.
- Stop your Orbix server applications and Orbix services.
- Check the health of Orbix services and Orbix server applications (for example, using `ps` and `grep`).

For more details, see “Starting and Stopping Orbix Services” on page 31.

Register your shell scripts

Finally, you must register your Orbix shell scripts with the clustering software used in your environment.

Configuring Services to Listen on Network Interfaces

By default, `itconfigure` generates domains in which services are configured to listen on all interfaces on that host. This behavior can be changed by explicitly specifying a list of listen addresses *as a comma-separated list of hostnames or IP addresses*. Using dedicated listen addresses is useful for example if you want to set up to domains on the same host listening on the same ports, where each domain's services would receive client requests from independent networks.

Specifying address lists using the Expert wizard

You can specify a listen address list on the first page of the **Expert** wizard. The value entered in this text field will be used in set the following configuration variable:

```
wka:transport:addr_list = ["publish address(listen address
   list):port"];
```

In this syntax, the publish address is the address (as hostname or IP address, depending on the selection of the address mode policy) the server uses when it publishes object references. See also section “Specifying Address Mode Policies” on page 54.

Example

When specifying a listen address list of `10.2.3.57` and deploying a locator, using insecure transports only and the default IIOP port, on host `orion`, the following variable will be defined for the locator:

```
plugins:locator:iiop:addr_list =
   ["orion(10.2.3.57):3075"];
```

Specifying address lists on the command line

You can also specify the listen addresses on the command line, using the `-listen_address_list` *`List`* option:

For example:

```
itconfigure -nogui -load sample_dd.xml
   -listen_address_list 10.2.3.57
```

In either case, the addresses are used only if and when deployment takes place. They are not stored in the deployment descriptor because they are not portable (i.e. they usually become meaningless if the descriptor is localized on another host).

Specifying Custom Locations for Domain Files

This section explains how to specify custom locations for all your configuration domain's files by passing properties to `itconfigure`. It includes the following topics:

- "Configuration domain files"
- "Command-line options for custom locations"
- "Setting all locations on the command line"
- "Partially setting custom locations"
- "Redeploying an existing domain"

Note: You can also set these custom locations using the Orbix Configuration GUI. For more details, see the **Advanced Settings** in "Creating a Domain in Expert Mode" on page 35.

Configuration domain files

Orbix configuration domain files include start, stop, and `_env` scripts, domain databases, domain log files, and configuration (`.cfg`) files.

Specifying custom locations for these domain files enables you to use a directory structure such as the following:

- `domains/bin/*_env|start*|stop*`
- `domains/config/*.cfg|cfr-*.cfg`
- `domains/dbs/`*`<domain>`*`/`*`<service>`*`/...`
- `domains/logs/`*`<domain>`*`/...`

Command-line options for custom locations

By default, domain start/stop and environment scripts are stored in the `bin` subdirectory of your *<config_dir>*. Domain configuration files are stored in the `domains` subdirectory in your *<config_dir>*.

By default, database files are stored in the *<domain_name>*`/dbs` subdirectory of your *<var_dir>*. Service log files are stored in the *<domain_name>*`/logs` subdirectory of your *<var_dir>*.

The default locations for *<config_dir>* and *<var_dir>* are shown in Table 1. These locations can be overwritten using the command-line options to `itconfigure` and properties displayed in Table 1.

Table 1: *Properties and Options for Custom Directory Locations*

Location for	Property	Command line option	Default location
Configuration files and scripts for all domains (*<config_dir>*)	`com.iona.deploy.config.dir`	`-etc`	**Windows**: `%IT_PRODUCT_DIR%\etc` **UNIX**: `/opt/etc/iona`, `$IT_PRODUCT_DIR/etc` or `$HOME/etc`
Database and log files for all domains (*<var_dir>*)	`com.iona.deploy.data.dir`	`-var`	**Windows**: `%IT_PRODUCT_DIR%\var` **UNIX**: `/opt/var/iona`, `$IT_PRODUCT_DIR/var` or `$HOME/var`

Setting all locations on the command line

The `itconfigure` command enables you to specify the custom locations for the domain log, data, script and configuration files. The configuration GUI also provides feedback on locations that are passed to `itconfigure` as properties. If all four configuration file locations are set, the GUI does not prompt for the `config` and `var` directories. Instead, it displays the values for these four directories in non-editable text fields.

To deploy your custom locations, specify your custom locations to `itconfigure` on the command line for example:

```
e:\Program Files\iona\asp\version\bin>itconfigure –name -nogui d1\
-Dcom.iona.deploy.config.bin.dir=e:\domains\bin \
-Dcom.iona.deploy.config.domains.dir=e:\domains\config \
-Dcom.iona.deploy.domain.db.dir=e:\domains\dbs\d1 \
-Dcom.iona.deploy.domain.log.dir=e:\domains\log\d1
```

This provides more fine-grained control. The command-line properties are explained in Table 2.

Table 2: *Properties for Custom File Locations*

Location for	Property	Default location
Domain start/stop and env scripts	`com.iona.deploy.config.bin.dir`	<*config_dir*>/bin
Domain configuration files	`com.iona.deploy.config.domains.dir`	<*config_dir*>/domains
Domain data files	`com.iona.deploy.domain.db.dir`	<*var_dir*>/<domain_name>/dbs
Domain log files	`com.iona.deploy.domain.log.dir`	<*var_dir*>/<domain_name>/logs

Note: If all four properties are specified, values for the `etc` and `var` directories do not need to be specified (their default values are not relevant). However, if any of these values is not specified, it defaults to a subdirectory of the `var` or the `etc` directory.

Partially setting custom locations

If not all four custom locations have been set, a value for the configuration and/or data directories is required, so that the missing value can be replaced with a subdirectory of the configuration or data directory. The GUI displays the configuration and data directories in editable text fields, and displays the directories that have already been set in non-editable text fields.

For example, specify the following on the command line:

```
e:\Program Files\iona\asp\version\bin>itconfigure \
-Dcom.iona.deploy.domain.db.dir=e:\domains\dbs\d1 \
-Dcom.iona.deploy.domain.log.dir=e:\domains\log\d1
```

This will be displayed in the **Storage Locations** screen. You can select the default configuration directory (for example, `e:\program_files\iona`), or overwrite this value with a custom location. If you click **Next** and continue to select and deploy services, your domain files will be located as follows:

scripts	`e:\Program Files\iona\etc\bin`
configuration files	`e:\Program Files\iona\etc\domains`
databases	`e:\domains\dbs\d1`
service and deployer log files	`e:\domains\log\d1`

Note: If the `etc` directory does not exist and needs to be created, `itconfigure` requires your confirmation. However, it does not require confirmation to create the domain log and domain database directories.

Redeploying an existing domain

Before deploying, the configuration tool checks for existing scripts in the `bin` directory, configuration files (and sub-directories named *<domain_name>*) in the `domains` directory, databases in the `dbs` directory, and logs in the `log` directory.

If any such files exist, this indicates that a domain with the same name already exists. The configuration tool only continues and deletes the existing files after your confirmation. This has the same effect as in the default case. For example, domain log files and domain databases are located in *<var directory>*/*<domain_name>*/dbs and *<var_directory>*/*<domain_name>*/logs. Only the sub-directories are deleted, leaving the *<var directory>*/*<domain_name>* directory.

Specifying Custom Library Paths

This section explains how to specify a custom library path on the command-line or using the configuration GUI.

This enables you to put shared libraries in different directories and still deploy, without needing to change system defaults that may need root/administrator permissions.

Using the command line

The `-libs` (shorthand `-L`) option to the `itconfigure` command has the following syntax:

`-libs` *<library-path>*

or

`-L` *<library-path>*

Specifying this option causes `itconfigure` to pass the supplied library path to the deployer. The deployer then prepends the path to the built-in path used when preparing and running Orbix services.

The library path argument is a list of directories to be searched for shared libraries when a service is run. The syntax of the list is the same as the platform-specific path syntax, as shown in the following examples.

UNIX:

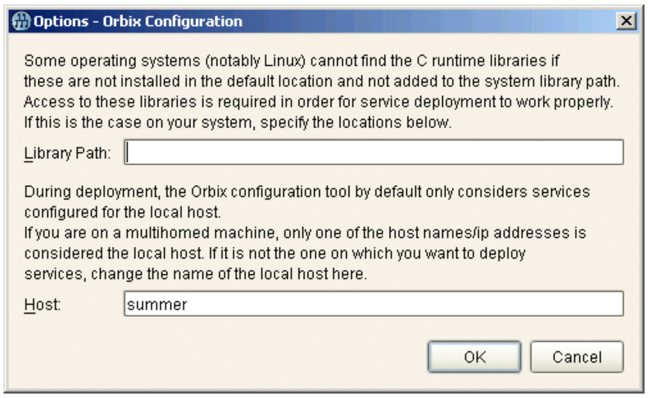
```
itconfigure -load sample_dd.xml -libs
   /usr/my_libs:/home/me/lib -nogui
```

Windows:

```
itconfigure -load sample_dd.xml -libs
   c:\usr\my_libs;d:\me\lib -nogui
```

Using the configuration tool

You can also use the **Orbix Configuration** tool to specify library paths to the deployer.



Some operating systems (notably Linux) cannot find the C runtime libraries if these are not installed in the default location and not added to the system library path. Access to these libraries is required in order for service deployment to work properly. If this is the case on your system, specify the locations below.

Library Path:

During deployment, the Orbix configuration tool by default only considers services configured for the local host.
If you are on a multihomed machine, only one of the host names/ip addresses is considered the local host. If it is not the one on which you want to deploy services, change the name of the local host here.

Host: summer

Figure 31: *Options dialog*

To specify a library path, perform the following steps:

1. Select **Tools**>**Options** to display the **Options** dialog.
2. Enter your path in the **Library Path** field, for example:

`c:\usr\my_libs;d:\me\lib`

If you are using a multi-homed machine, you can enter the required machine name in the **Host** field.

Using Custom XML Files

This section explains how to automate the process of deploying an Orbix configuration domain, and subsequently add or modify some of its configuration data (for example, adding a scope for a service developed at your site).

In previous versions of Orbix (for example 5.x), you could only do this by manually modifying the `ABDriver.dtd` and `<`*`domain_name`*`>_driver.xml` files generated by the `itconfigure` tool. Orbix 6.0.2 and higher enable you to do this by passing a system property to the `itconfigure` command. This section includes the following topics:

- “Passing custom XML to itconfigure”
- “Deploying custom XML with the GUI”
- “Custom XML example”
- “Rules for writing XML files”

Passing custom XML to itconfigure

To use custom XML files, you must first supply the path to the directory containing your files to the `itconfigure` tool. You can do this by passing a system property to `itconfigure`, for example:

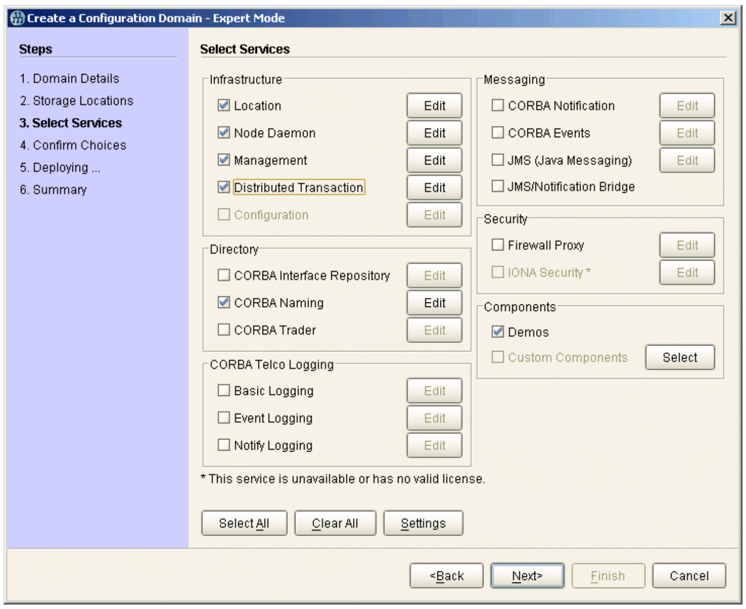
```
itconfigure -
   Dcom.iona.deploy.custom.xml.dir=e:\custom\conf
```

The specified directory should exist and contain at least one file with the `.xml` extension.

Deploying custom XML with the GUI

To deploy custom XML files, perform the following steps:

1. Select **File**>**New**>**Expert** from main menu.
2. Click **Next** to display the **Storage Locations** screen
3. Click **Next** to display then **Select Services** screen, shown in Figure 32. The **Custom Components** checkbox at the bottom right of the screen is disabled. This is unchecked when no custom components are selected.


Create a Configuration Domain - Expert Mode

Steps
1. Domain Details
2. Storage Locations
3. Select Services
4. Confirm Choices
5. Deploying ...
6. Summary

Select Services

Infrastructure
✓ Location
✓ Node Daemon
✓ Management
✓ Distributed Transaction
☐ Configuration

Directory
☐ CORBA Interface Repository
✓ CORBA Naming
☐ CORBA Trader

CORBA Telco Logging
☐ Basic Logging
☐ Event Logging
☐ Notify Logging

Messaging
☐ CORBA Notification
☐ CORBA Events
☐ JMS (Java Messaging)
☐ JMS/Notification Bridge

Security
☐ Firewall Proxy
☐ IONA Security *

Components
✓ Demos
☐ Custom Components

* This service is unavailable or has no valid license.

Figure 32: *Custom Components in Select Services*

4. Click the **Select** button on the right of the **Custom Components** checkbox to display the **Select Custom Components** dialog, shown in Figure 33. This enables you to select components from your specified directory.
5. Click **OK**. The **Custom Components** checkbox is then displayed as checked.

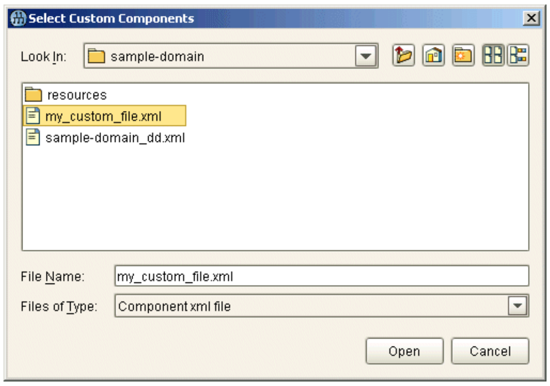


Figure 33: *Select Custom Components*

Custom XML example

For example, if you select the custom XML file with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ABDeploy SYSTEM "ABDeploy.dtd">
<ABDeploy>
    <service>
        <dataId>example_using_custom_xml_files</dataId>
    </service>

    <process>
      <stage action="filePopulate">
         <source>
              <Dsection>main</Dsection>
         </source>
      </stage>
    </process>

    <section name="main">
        <configScope>
            <dataId>custom</dataId>
        </configScope>

        <configData scope="custom">
            <dataId>custom:example:var</dataId>
            <dataType>list</dataType>
            <dataValue>This</dataValue>
            <dataValue>is</dataValue>
            <dataValue>just</dataValue>
            <dataValue>an</dataValue>
            <dataValue>example!</dataValue>
        </configData>
    </section>
</ABDeploy>
```

Then the generated configuration will include the following fragment:

```
custom
{
  custom:example:var = ["This", "is", "just", "an", "example!"];
};
```

Note: If you select more than one custom component, the order in which they are deployed is non-deterministic. Do not make any assumptions about the order in which custom components are deployed, except that they are deployed after all Orbix services and components.

Rules for writing XML files

If you write your own XML files, you should obey the following rules:

- Only use a simple service element (one with just a `dataId` child).
- Use simple process elements and stages with one of the following actions only: `filePopulate`, `configPopulate`.
- Do not use constraints.
- Use `configData` elements with a `dataType` of `list`, `string`, or `long`.
- Do not use external entities.

WARNING: The schema for the Orbix deployer XML files is not fully documented. A subset of the complete DTD is supported and documented. Unsupported features are subject to change without notice. For details, see "Orbix Deployment DTD".

Specifying Address Mode Policies

This section explains how to use address mode policies to control the way in which host names and/or IP addresses are published in IORs. In previous versions of Orbix, you could do this by specifying the host DNS alias or IP address. Orbix 6.0 and later use policies. These are portable and enable you design your configuration domain on one host (run `itconfigure` in GUI mode and save the descriptor), and deploy it elsewhere, without the need to supply actual hostnames or IP addresses at that later stage.

This section includes the following topics:

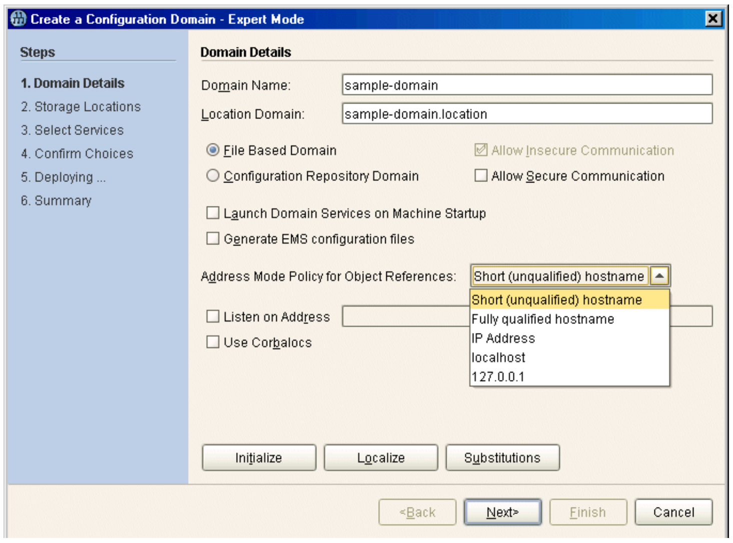
- "Selecting an address mode"
- "Specifying a fully-qualified hostname"
- "Persistence of address mode policies"
- "Restrictions and special cases"
- "Node daemon address mode policies"
- "Leaving the node daemon name unset"

- “Assigning a node daemon name other than the hostname”

Selecting an address mode

To select an address mode, perform the following steps:

1. Run the configuration GUI using the `itconfigure` command.
2. Select **File**>**New**>**Expert** from main menu. This displays the **Domain Details** screen.
3. Select your preferred policy using the **Address mode policy for Object References** drop-down box, shown in Figure 34.


Create a Configuration Domain - Expert Mode

Steps
1. Domain Details
2. Storage Locations
3. Select Services
4. Confirm Choices
5. Deploying ...
6. Summary

Domain Details

Domain Name: sample-domain
Location Domain: sample-domain.location

(●) File Based Domain
( ) Configuration Repository Domain
[✓] Allow Insecure Communication
[ ] Allow Secure Communication
[ ] Launch Domain Services on Machine Startup
[ ] Generate EMS configuration files

Address Mode Policy for Object References: Short (unqualified) hostname
- Short (unqualified) hostname
- Fully qualified hostname
- IP Address
- localhost
- 127.0.0.1

[ ] Listen on Address
[ ] Use Corbalocs

Initialize | Localize | Substitutions

<Back | Next> | Finish | Cancel

Figure 34: *Selecting an Address Mode Policy*

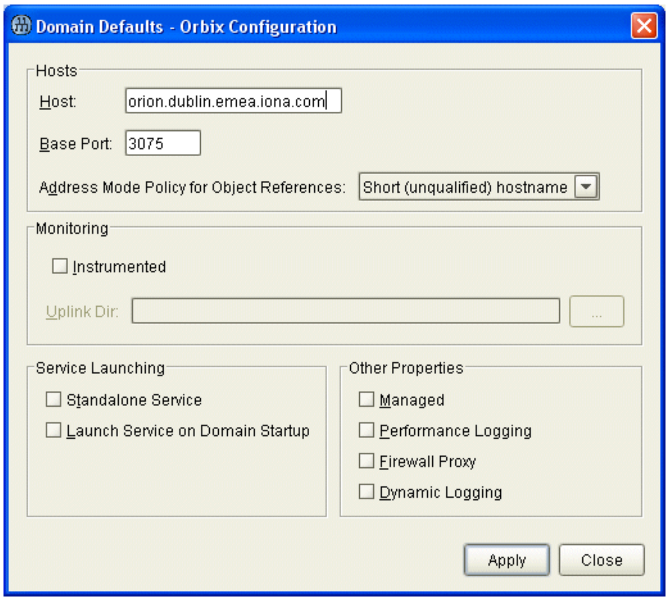
Specifying a fully-qualified hostname

To use fully qualified hostnames in IORs, you must ensure that `itconfigure` knows the fully qualified host name. Depending on your network configuration, this cannot always be obtained with JDK 1.3 APIs.

However, you can do this by invoking the `itconfigure` command using the `-host` option, for example:

```
itconfigure -host orion.dublin.emea.iona.com
```

Alternatively, you can edit the host field in the **Domain Defaults** dialog shown in Figure 35. This dialog is displayed opens when you click **Settings** on the **Service Settings** screen, shown in Figure 35:



Host: orion.dublin.emea.iona.com
Base Port: 3075
Address Mode Policy for Object References: Short (unqualified) hostname
Instrumented: ☐
Uplink Dir:
Standalone Service: ☐
Launch Service on Domain Startup: ☐
Managed: ☐
Performance Logging: ☐
Firewall Proxy: ☐
Dynamic Logging: ☐

Figure 35: *Specifying a Hostname*

Persistence of address mode policies

If you chose not to deploy now, and save the descriptor to deploy on other hosts, you can still use the selected address mode policy on the other hosts because the policy is persisted by the descriptor.

The descriptor stores addresses as policies (instead of literal string IP addresses or names). This enables you to apply the same policy on other hosts, using the `-localize` option to `itconfigure`. For more information, see "Replicating Services in a Domain" on page 25.

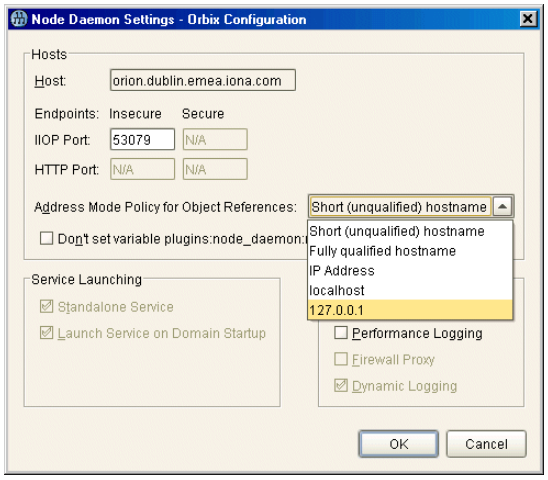
Restrictions and special cases

While the deployment descriptor schema supports node-specific address mode policies, the Orbix configuration GUI only allows you to specify the address mode policy on a global level—for all nodes.

If you must use different policies on different nodes, please refer to "Orbix Deployment Descriptors", and manually edit the descriptor. The same applies if you want one more level of granularity and specify address mode policies on a per-service basis. There is one case, however, where you can specify address mode policies on a per-service basis. The Orbix configuration tool enables you to set service-specific address mode policies for the node daemon.

Node daemon address mode policies

The **Node Daemon Settings** dialog, shown in Figure 36, enables you to specify the address mode policy for node daemons:


Node Daemon Settings - Orbix Configuration

Hosts

Host: orion.dublin.emea.iona.com

Endpoints:	Insecure	Secure
IIOP Port:	53079	N/A
HTTP Port:	N/A	N/A

Address Mode Policy for Object References: Short (unqualified) hostname

☐ Don't set variable plugins:node_daemon:

Service Launching

☑ Standalone Service

☑ Launch Service on Domain Startup

☐ Performance Logging

☐ Firewall Proxy

☑ Dynamic Logging

Figure 36: *Node Daemon Settings Dialog*

Therefore, if you want all services, except for the node daemon, to publish fully qualified host names, you must first change the global address mode policy to `fully qualified hostname`. For the node daemon, you can override this with the localhost IP policy (`127.0.0.7`).

Note: This policy is used for all node daemons in the domain. `itconfigure` does not allow you to interactively specify the node daemon's address mode policy on a per instance basis.

Leaving the node daemon name unset

It is also possible (but not recommended) to avoid giving the node daemon an explicit name. The default is node daemon name is `iona_services.node_daemon.`*`<hostname>`*, which corresponds to the ORB name and configuration scope. For example:

```
iona_services.node_daemon.my_hostname {
   plugins:node_daemon:name =
   "iona_services.node_daemon.my_hostname";
   ...
}
```

To leave the node daemon name unset, check **Don't set variable plugins:node_daemon:name**, displayed in Figure 36.

If a node daemon does not have a name configured, at startup, it registers itself with the locator, and identifies itself as a node daemon named *<host>*, where *<host>* is obtained by `gethostname()`. This value depends on the host on which the node daemon is started.

Note: This may break the mapping between a process and node daemon. A process that is registered to be monitored and started on demand by node daemon *<activating host>* can only be activated if a node daemon with the name *<activating host>* exists. In addition, generated start and stop scripts will not stop such a node daemon.

Assigning a node daemon name other than the hostname

You can also assign a node-daemon name that is not related to the hostname. You can do this by editing the node details in the deployment descriptor file. The recommended approach is to deploy using `itconfigure`, update the deployment descriptor, and then re-deploy with the new descriptor file.

The following example shows these steps in more detail:

1. Deploy as normal using `itconfigure`. However, you must choose the IP address mode policy (see "Selecting an address mode"). For example, for a machine named `spring`, the deployment descriptor would include:

```
<dd:nodes>
    <dd:node name="spring" ip="10.5.1.101"
   profile="spring">
        <dd:policies>
            <dd:policy name="address_mode" value="ip"/>
        </dd:policies>
   </dd:node>
</dd:nodes>
<dd:profile id="spring">
```

2. Update the deployment descriptor to remove any relation between the hostname (`spring`) and the node name (in this case, `summer`). For example:

```
<dd:nodes>
   <dd:node name="summer" ip="10.5.1.101"
   profile="summer">
       <dd:policies>
          <dd:policy name="address_mode" value="ip"/>
       </dd:policies>
   </dd:node>
</dd:nodes>
<dd:profile id="summer">
```

The values of the `profile` and `node name` attributes can be different, for example:

```
<dd:nodes>
   <dd:node name="summer" ip="10.5.1.101"
   profile="profile_1">
       <dd:policies>
          <dd:policy name="address_mode" value="ip"/>
       </dd:policies>
   </dd:node>
</dd:nodes>
<dd:profile id="profile_1">
```

In addition, for a node on which a locator is running, the value of the `ip` attribute can be an IP alias instead of an IP address, for example:

```
<dd:node name="1" ip="001my.intra.net"
   profile="profile_loc_1">
   <dd:policies>
      <dd:policy name="address_mode" value="ip" />
      <dd:policy name="use_corbalocs" value="true" />
   </dd:policies>
   <dd:resource name="iiop" value="17701" />
   <dd:resource name="iiop_secure" value="17702" />
</dd:node>
```

3. Finally, re-deploy the domain with the updated deployment descriptor:

```
itconfigure -nogui -load <deployment_descriptor_name_dd.xml>
```

Specifying Custom Certificates

The **Orbix Configuration** tool can process custom values for any substitution variables. These are variables that have been defined in the `substitute_vars` section of the XML files in your *install-dir*/`asp`/*version*/`etc/conf` directory. This section shows how to use substitution variables to specify custom security certificates. It includes the following:

- “Substitution variable file”
- “Using the GUI”
- “Using the command line”

Substitution variable file

When using substitution variables, you must pass a substitution variable file to `itconfigure`. This file contains replacement values for all defined substitution variables. This file should be specified in property file format, for example:

```
ROOT_TRUSTED_CA_LIST_POLICY="C:\\orbix\\custom_certs\\calist.pem"

ITADMIN_ADMIN_CERT_ROOT_DIR="C:\\orbix\\custom_certs"

SERVICES_AUTH_METHOD_DATA=["filename=C:\\orbix\\custom_certs\\administrator.p12",
   "password_file=C:\\orbix6\\custom_certs\\administrator.pwf"]

UTILITIES_AUTH_METHOD_DATA=["filename=C:\\orbix\\custom_certs\\utilities.p12",
   "password_file=C:\\orbix6\\custom_certs\\utilities.pwf"]

KDM_AUTH_METHOD_DATA=["filename=C:\\orbix\\custom_certs\\kdmadministrator.p12",
   "password_file=C:\\orbix6\\custom_certs\\kdmadministrator.pwf"]

KDM_CERT_CONSTRAINTS=["C=US,O=ABigBank*,CN=abc*","C=US,O=ABigBank*,CN=xyz*"]

KDM_ADM_CERT_CONSTRAINTS=["C=US,O=ABigBank*,CN=abc*"]
```

Each value must be enclosed in quotes as shown in this example file.

Using the GUI

In the **Orbix Configuration** GUI tool, the **Substitutions** button is displayed at the bottom of the **Expert** panel (see Figure 34 on page 55). When the **Substitutions** button is clicked, this launches a dialog that enables you to enter the location of the substitution file.

Using the command line

On the command line, you can use the `-substitutions` option to pass the substitution file location to `itconfigure`, for example:

```
itconfigure -nogui -load descriptor -substitutions
   properties file
```

WARNING: When using the `-substitutions` option to configure custom certificates for a deployed system, the substitutions file should *not* contain the pass phrase directly.

Instead, it should contain the location of the password files containing the pass phrases (see "Substitution variable file" on page 60).

Because the password file(s) stores the pass phrase in plain text, such files should not be readable by anyone except the administrator.

Orbix Deployment Descriptors

This chapter explains the data structure and grammar of the Orbix domain deployment descriptor.

Deployment Descriptor Structure

The Orbix domain deployment descriptor (`domain-name_dd.xml`) describes the contents of a configuration domain. This section outlines the overall structure of this file. It includes the following topics:

- "Document structure"
- "Recommended deployment descriptor generation"
- "Validating manual changes to a deployment descriptor"

Document structure

The `<domain-name>_dd.xml` file must conform to the following document structure:

Example 1: *Deployment Descriptor Structure*

```
   <?xml version="1.0" encoding="UTF-8"?>
1  <dd:descriptor xmlns:dd="http://ns.iona.com/aspdd">
     <!--This deployment descriptor version 1.0 has been generated
      by Orbix tools-->
2    <dd:configuration>
       <dd:domain>domain-name</dd:domain>
        ...
     </dd:configuration>

     <!--Concrete node information for this deployment-->
3    <dd:nodes>
        <dd:node name="hostname" ip="ip-address" profile="hostname"
           <dd:resource name="some-resource" value="some-value" />
           ...
           <dd:policies>
                <dd:policy name="some-policy" value="some-value" />
           </dd:policies>
           ...
        </dd:node>
        ...
     </dd:nodes>

4    <dd:feature id="feature-name">
        <dd:resource type="directory" name="some-resource" />
     </dd:feature>

     <!--The following profiles will be deployed-->
5    <dd:profile id="hostname">
```

Example 1: *Deployment Descriptor Structure*

```
6      <dd:service name="service-name" ... >
       ...
       </dd:service>
       ...
7    <dd:component />
       ...
   </dd:profile>
 </dd:descriptor>
```

This deployment descriptor structure is described as follows:

1. The `<dd:descriptor>` element is the containing root element of the deployment descriptor XML vocabulary. It specifies an XML namespace named `dd`. This element indicates what version of the deployment descriptor XML vocabulary is being used. In this case, the absence of a version attribute indicates that this is version 1.0.
2. The `<dd:configuration>` element specifies the general configuration information for the domain (for example, its name, type, and location domain).
3. The `<dd:nodes>` element specifies information about the host machines included in the domain. Each `<dd:nodes>` element one or more `<dd:node>` element, one for each host machine. A `<dd:node>` element can include optional `<dd:resource>` and `<dd:policies>` elements. A `<dd:resource>` element specifies resources used by domain-level features; while `dd:policies` specifies policies that apply to all services on that node.
4. The `<dd:feature>` element specifies information about domain-level features.
5. The `<dd:profile>` element specifies a logical group of services and components that maps to a particular node.
6. The `<dd:service>` element specifies the details for a particular service (for example, the naming service).
7. The `<dd:component>` element specifies the details for a particular component (for example, Orbix demos). The difference between a component and a service is that services maintain live database information as part of the domain state, whereas a component does not.

These elements are described in more detail with examples in the sections that follow.

Recommended deployment descriptor generation

The recommended method of generating a deployment descriptor is to run the Orbix configuration tool on a GUI-enabled machine, and, if necessary, save the deployment descriptor for later use in command-line. Generating the descriptor in GUI mode ensures that the generated XML document is valid, and checked for dependencies.

Certain combinations of services and features are not permitted. For example, a descriptor that contains an indirect persistent, on-demand naming service, but no node-daemon, is invalid. Using different transports for different services is also invalid. Lastly, a descriptor with a node daemon that has secure endpoints only, and a locator with insecure endpoints only is not valid. This is because the locator would not be able to communicate with the node daemon.

Validating manual changes to a deployment descriptor

You can edit the domain deployment descriptor file to meet your requirements using any text editor. However, any changes you make must be checked for validity and dependencies.

Running the Orbix configuration tool enforces consistency on a deployment descriptor that has inconsistent relationships between services, or has incorrect container descriptions. You can validate manual changes to a deployment descriptor by running the following command:

```
itconfigure -nogui -load descriptor.xml -save
   somefile.xml
```

If the descriptor is correct, `descriptor.xml` and `somefile.xml` will be identical in structure. Otherwise, the configuration tool reports an error message, and exits without saving to the specified document (`somefile.xml`).

Domain Configuration Elements

This section explains the domain-specific information contained in an example deployment descriptor file. It includes the following topics:

- “Example descriptor”
- “Domain elements”

Example descriptor

The following extract from a deployment descriptor file named `my-domain_dd.xml` shows some example domain-specific elements:

Example 2: *Domain-Specific Configuration*

```
<?xml version="1.0" encoding="UTF-8"?>
<dd:descriptor xmlns:dd="http://ns.iona.com/aspdd">
   <!--This deployment descriptor version 1.0 has been
   generated by Orbix tools-->
  <dd:configuration>
    <dd:domain>my-domain</dd:domain>
    <dd:source>file</dd:source>

   <dd:location_domain>my-domain.location</dd:location_do
   main>
  </dd:configuration>
  <!--Concrete node information for this deployment-->
  <dd:nodes>
    <dd:node name="summer" ip="10.2.4.82"
   profile="summer" />
  </dd:nodes>
  <!--The following profiles will be deployed-->
  <dd:profile id="summer">
  ...
  </dd:profile
</dd:descriptor>
```

Domain elements

The following table explains all the domain-specific elements:

Table 3: *Domain-Specific Elements*

Element	Description
`<dd:descriptor>`	Specifies the XML namespace details for the deployment descriptor.
`<dd:configuration>`	Specifies the general configuration information for the domain (for example, its name, type, and location domain)
`<dd:domain>`	Specifies the configuration domain name (in this case, `my-domain`).
`<dd:source>`	Specifies the configuration domain type. Can be either `file`, `cfr`, or `link` (`.cfg` text file, Configuration Repository, or a link domain).
`<dd:location_domain>`	Specifies the location domain name. This takes the form `<`*`domain-name`*`>.location` (for example, `my-domain.location`). A location domain is a group of servers that are registered with the same locator daemon.

Table 3: *Domain-Specific Elements*

Element	Description

`<dd:nodes>`	This is a container for all host machines in a configuration domain that belong to the same `dns` domain. It has a single `dns` attribute (for example, `dns="dublin.emea.myco.com"`).
	There can be multiple `<dd:nodes>` in one deployment descriptor. For example:
	`<dd:nodes dns="dublin.emea.myco.com">`
	`  <dd:node name="summer" ip="10.2.4.82"`
	`   profile="summer.dublin.emea.myco.com"/>`
	`  <dd:node name="onion" ip="10.2.1.101"`
	`   profile="onion.dublin.emea.myco.com" />`
	`</dd:nodes>`
	
	`<dd:nodes dns="boston.amer.mycorp.com">`
	`  <dd:node name="jupiter" ip="10.5.3.18"`
	`   profile="jupiter.boston.amer.mycorp.com" />`
	`</dd:nodes>`

`<dd:node>`	Specifies the identity of a particular host machine in the domain. It has three attributes: • `name` specifies the hostname. • `ip` specifies the IP address. • `profile` specifies a logical group of services and components to deploy on the specified node. A `<dd:node>` element can also include optional `<dd:resource>` and `<dd:policies>` elements.
`<dd:profile>`	Specifies a logical group of services and components. Its `id` attribute corresponds to the `<dd:node profile>` attribute. In this version of Orbix, only one profile per node is supported.

`<dd:feature>`	Specifies information about optional domain-level features. These are implemented separately from the deployer and invoked at the end of the deployment process. The following example is for integration with IBM Tivoli management:
	`<dd:descriptor ...`
	`  <dd:feature`
	`   xmlns:dd="http://ns.iona.com/aspdd"`
	`   id="tivoli-integration">`
	`      <dd:resource type="directory"`
	`         name="configuration-files" />`
	`  </dd:feature>`
	`  ...`
	`</dd:descriptor>`

`<dd:resource>`	Specifies resources used by domain-level features. For example:
	`<dd:resource type="directory"`
	`         name="configuration-files" />`
	This specifies a resource that is a file system directory named `configration-files`.

Table 3: *Domain-Specific Elements*

Element	Description

`<dd:policies>`	As a child of the `<dd:node>` element, specifies policies that apply to all services on that node. Currently, there is only one available policy:
	`address_mode`
	For example:
	`<dd:nodes>`
	` <dd:node name="orion2" ip="10.2.1.101">`
	`    <dd:policies>`
	`      <dd:policy name="address_mode"`
	`         value="ip" />`
	`    </dd:policies>`
	`  </dd:node>`
	`</dd:nodes>`
	For more details on this example, see "Conversion Process from Orbix 5.1 to an Orbix 6.x Descriptor" on page 73.
	Policies can also be specified on a per-service bases (see "Profile Configuration Elements" on page 66). Service-specific policies override node-specific policies.

Profile Configuration Elements

A profile specifies a group of configured services and components for a particular node. This section explains the profile-specific information contained in an example deployment descriptor file. It includes the following topics:

- "Example descriptor"
- "Service elements"
- "Service and component XML Files"

Example descriptor

The following is a complete listing of a deployment descriptor file named `my-domain_dd.xml`. It shows an entire profile configured for a default domain:

Example 3: *Profile Configuration*

```
<?xml version="1.0" encoding="UTF-8"?>
<dd:descriptor xmlns:dd="http://ns.iona.com/aspdd">
  <!--This deployment descriptor has been generated by ASP
   tools-->
  <dd:configuration>
    <dd:domain>my-domain</dd:domain>
    <dd:source>file</dd:source>

   <dd:location_domain>my-domain.location</dd:location_do
   main>
  </dd:configuration>

  <!--Concrete node information for this deployment-->
  <dd:nodes>
    <dd:node name="summer" ip="10.2.4.83"
   profile="summer" />
  </dd:nodes>
  <!--The following profiles will be deployed-->
  <dd:profile id="summer">
    <dd:service name="locator">
      <dd:activation mode="manual" />
      <dd:run mode="direct_persistent" proxified="false"
          managed="true" authenticated="false" />
      <dd:endpoint protocol="iiop" port="3075" />
    </dd:service>

    <dd:service name="node_daemon">
      <dd:activation mode="manual" />
      <dd:run mode="direct_persistent" proxified="false"
         managed="true" authenticated="false" />
      <dd:endpoint protocol="iiop" port="53079" />
    </dd:service>
    <dd:service name="naming">
      <dd:activation mode="on_demand" />
     <dd:run mode="indirect_persistent" proxified="false"
         managed="true" authenticated="false" />
      <dd:endpoint protocol="iiop" port="0" />
    </dd:service>
    <dd:service name="management">
      <dd:activation mode="manual" />
      <dd:run mode="direct_persistent" proxified="false"
         managed="true" authenticated="false" />
      <dd:endpoint protocol="iiop" port="53085" />
      <dd:endpoint protocol="http" port="53185" />
    </dd:service>
    <dd:component name="demos" />
  </dd:profile>
</dd:descriptor>
```

Service elements

The following table explains the profile-specific elements

Table 4: *Profile-Specific Elements*

Element	Description
`<dd:service>`	Specifies the identity of a service. Its `name` attribute is the service name (for example, `locator`).
`<dd:activation>`	Specifies how a service is activated. Its single `mode` attribute has the following possible values: • `manual` specifies that it must be activated using a start command or a script. • `on_demand` means that the node daemon starts the service when requested by a client. • `system_service` specifies that the service will be started at boot time. On Windows, the service will be installed as an NT service. On Unix, appropriate run control scripts will be created. For more details, see the *Orbix Administrator's Guide*.
`<dd:run>`	Specifies how a service is run. It has the following attributes, all of which are optional: • `mode` specifies whether the service uses the locator to resolve persistent object references (indirect persistence), or its IOR contains a well-known address for the server process (direct persistence). Possible values are `indirect_persistent` or `direct_persistent`. Defaults to `indirect_persistent`. • `proxified` specifies whether service is registered with the Firewall Proxy Server. Possible values are `true` or `false`. This attribute is optional. Defaults to `false`. • `managed` specifies whether service is registered with the management service. Possible values are `true` or `false`. Defaults to `false`. • `authenticated` specifies whether the service is registered with the security service. Possible values are `true` or `false`. Defaults to `false`. • `perflog` specifies whether the service is configured for performance logging. This is necessary for integration with Enterprise Management Systems (for example, IBM Tivoli). Possible values are `true` or `false`. Defaults to `false`. • `dynlog` specifies whether the service is configured for dynamic logging. For details on dynamic logging, see the *Orbix Administrator's Guide*.

Table 4: *Profile-Specific Elements*

Element	Description
`<dd:endpoint>`	Specifies details of a service communication endpoint. It has three attributes: • `protocol` specifies the protocol used by the service. Possible values are `iiop` and `http`, as well as `fps<n>`, where `<n>` is the number of the proxy group. The `fps<n>` protocol is only used by the Firewall Proxy Service to indicate its proxy ports. • `port` specifies the port number used by the service (for example, `9000`). • `secure` specifes if the endpoint is secure. Values are `true` or `false`. A secure endpoint is one that includes TLS (Transport Layer Security). For example, if `secure="true"` is set on an endpoint where `protocol="http"`, a `https` endpoint is configured.

`<dd:configuration>`	Specifies configuration overrides for the service. This enables you to change a small number of configuration settings in your domains, at the scope of a service, without modifying the shared description.
	`<dd:service name="..." ... >`
	`    <dd:configuration`
	`        name="variable-name"`
	`        value="value" action="set"`
	`        stage="preprepare" />`
	`    <dd:configuration`
	`         name="variable-name"`
	`         action="unset" />`
	`    ...`
	`</dd:service>`
	Available actions are `set` and `unset`. The default is `set`, so the `action` attribute can be omitted. Configuration overrides only change the value at the service instance scope.

`<dd:policies>`	Specifies information about any policy overrides for that service. Currently, there is only one available policy:
	`address_mode`
	Specified values must match those already specified in the `<dd:node>` element (see "Domain Configuration Elements" on page 63).
	The following example shows policy overrides for address modes and ORB hostnames:
	` <dd:service ... >`
	`   <dd:policies>`
	`     <dd:policy name="address_mode"`
	`        value="ip" />`
	`     </dd:policies>`
	`</dd:service>`
	For more details on this example, see "Conversion Process from Orbix 5.1 to an Orbix 6.x Descriptor" on page 73.

Table 4: *Profile-Specific Elements*

Element	Description
`<dd:component>`	Specifies a component for the profile. It has a single `name` attribute. An example value is `demos`.

Service and component XML Files

`<dd:service>` and `<dd:component>` elements have corresponding XML source documents containing the data needed to deploy the configuration domain. Many of these XML source documents correspond to Orbix services. Other XML documents contain core information that is needed for all configurations.

Note: These XML source documents are proprietary IONA documents. These XML source documents and their XML schema are not fully documented and subject to change without notice.

However, to enable you to write and use your own custom XML source documents, a subset of the schema is documented and supported. Custom XML files that comply with this partial schema will continue to work with future versions of Orbix, even though the overall schema may change. For details of the partial schema, see "Orbix Deployment DTD".

Migrating Orbix Deployments

For users who have modified Orbix 5.1 driver files, this chapter shows how to migrate to Orbix 6.x, and explains the automated conversion process in detail. For users with existing Orbix 6.x deployments, this chapter explains how to import existing domain information into an Orbix 6.3 domain.

Migrating from Orbix 5.1 Driver Files

This section explains how to migrate from Orbix 5.1 driver files to an Orbix 6.x deployment descriptor. This applies to customers who have modified `ABDriver.dtd` and/or `<domain>_driver.xml` files. It includes the following topics:

- "Approach to migration"
- "Using the itconfigure command line"
- "Using the itconfigure GUI"
- "Migrating custom XML"
- "Adding new Orbix 6.x features"

Approach to migration

The approach used is to generate an Orbix 6.x deployment descriptor by retrieving the domain topology (selected domain services) from the driver file, and the service details (for example, port numbers) from the `ABDriver.dtd` file. This descriptor is then passed to the `itconfigure` tool, as if it had been created by `itconfigure`. This enables migration of domain topology only, no domain data is migrated (for example, database and log files).

The implementation is limited to driver files for domains without replicated services. Driver file entries with the component attribute `role=replica` result in an exception. The deployer also rejects driver files for link domains (links can always be recreated), and driver files for domains that include a J2EE application server.

Using the itconfigure command line

For example, to generate an Orbix 6.x deployment descriptor using the command line, enter the following:

```
E:\Program Files\IONA\asp\version\bin>itconfigure -nogui
-compatible \ -load e:\drivers\my-domain_driver.xml \
-entities e:\drivers\ABDriver.dtd \
-etc e:\etc -var e:\var
```

Using the itconfigure GUI

For example, to generate an Orbix 6.x deployment descriptor using the configuration GUI, enter the following:

```
E:\Program Files\IONA\asp\version\bin>itconfigure
   -compatible \
-load e:\drivers\my-domain_driver.xml \
-entities e:\drivers\ABDriver.dtd \
-etc e:\etc -var e:\var
```

The services specified in the driver file are displayed as selected in the GUI, with their service details as specified in the `ABDriver.dtd` file. You can subsequently add more services, or change the details for the pre-selected services in the GUI, before proceeding to deploy the domain.

Migrating custom XML

Migration can also be used in conjunction with custom component files (see "Using Custom XML Files" on page 51).

If your Orbix 5.1 driver files specify one or more components that are not recognized as Orbix components, and you pass the directory containing these XML files using the `–Dcom.iona.deploy.custom.xml.dir` property, the deployment will also include your custom components.

If you use configuration tool in GUI mode, and save the descriptor, this descriptor also includes your custom components.

Adding new Orbix 6.x features

Because address mode policies (and hostname policies for the ORB) are now persisted in the deployment descriptors, you can migrate 5.1 domains, and also add Orbix 6.x features and services to your domains, without losing what has been extracted from the driver and entities files.

The following steps show how to migrate and add new features at the same time:

1. Convert the driver and/or entities file to a descriptor, without deploying the services, as follows:

   ```
   itconfigure –nogui –compatible –load <driver> -entities <entities>
   ```

2. Process the descriptor using proprietary tools to add the new feature (for example, a security service).
3. Deploy the extended descriptor using the following command:

```
itconfigure -nogui -load <extended_descriptor> -etc <etc_dir> -var
   <var_dir>
```

Conversion Process from Orbix 5.1 to an Orbix 6.x Descriptor

This section explains the Orbix 5.1 to Orbix 6.x conversion process in more detail. It shows how the Orbix configuration tool constructs an Orbix 6.x deployment descriptor from an Orbix 5.1 `<domain_name>_driver.xml` and `ABDriver.dtd` file. It includes the following topics:

- "Stage 1—Constructing an empty descriptor"
- "Stage 2—Parsing of driver files and construction of node profiles"
- "Stage 3—Obtaining the service details from ABDriver.dtd"
- "Stage 4—Obtaining the address mode policy"
- "Rules for inferring the address mode policy"
- "Ensuring ORB name compatibility"
- "Example conversion"
- "Conversion for virtual hosts"

Stage 1—Constructing an empty descriptor

An empty deployment descriptor is constructed with a domain name and location domain name, as found in the `ABDriver.dtd` file.

If no definition for the `config.domain.name` entity is found, an exception is thrown. If no value for the `location_domain_name` entity is found, the Orbix 6.x default is used (`<domain_name>.local`). Initially, the domain type is file-based.

Stage 2—Parsing of driver files and construction of node profiles

The `<domain_name>_driver.xml` files are parsed to enable the construction of service entries for the deployment descriptor's local node profile. Any constraints and the ordering of the driver file entries are ignored. Orbix 6.x does not depend on the order of the entries in a deployment descriptor when deploying a domain—it automatically constructs it correctly. Driver component entries are processed as follows:

CFR domains	A component named `config_rep.xml` causes the descriptor's domain type to be changed to CFR based, and adds a service element into the descriptor's local node profile.

Ignored components	Components named `init.xml`, `init_svcs.xml`, `file_core.xml`, `file_svcs.xml`, `comet.xml`. `admin.xml`, `tool_corba.xml` are ignored
Link domains	A component named `link.xml` results in an exception (no conversion of driver files for link domains).
Replicas	A component with the `role` attribute set to `replica` results in an exception (no conversion of driver files for domains with replicas).
Demos	A component named `demos.xml` results in a `component` element being added to the descriptor's local node profile.
Others	All other components, provided they are known Orbix components, result in a `service` element being added to the descriptor's local node profile. If they are not known Orbix components (for example, `custom.xml`), a `component` element is added to the descriptors local node profile.

Stage 3—Obtaining the service details from ABDriver.dtd

For every driver component entry for which a corresponding service element has been added to the descriptor's local node profile, `ABDriver.dtd` is consulted to determine the service details:

Direct/Indirect Persistence: `cfr`, `management`, `locator` and `node_daemon` service elements are always set to be direct persistent—regardless of the constraints in the driver component element and the content of `ABDriver.dtd`.

For all other services, if the `<`*`service_name`*`>.direct_persistence` entity is defined in `ABDriver.dtd`, and if its value is `true` or `yes`, the service is set to be direct persistent. The default for a service element is indirect persistent.

Start Mode: `cfr`, `management`, `locator` and `node_daemon` service elements are always set to be started manually—regardless the constraints of the driver component element and the content of `ABDriver.dtd`.

For all other services, if the `<`*`service_name`*`>.mode` entity is defined in `ABDriver.dtd`, and if its value is `manual` or `boot`, the service is set to be started manually (default for a service element is on demand).

Subsequently, if the `config.daemon.install` entity is defined in `ABDriver.dtd` and if its value is `true`, the startup mode of a service is promoted to system service, if it had been manual. On Windows it is installed as an NT service.

Ports: If the component's security attribute in the `<`*`domain_name`*`>_driver.xml` file is set to `iiopOnly` or `iiopTls`, and if the `<`*`service_name`*`>.port` entity is defined (is a number and not zero), an endpoint element is created in the corresponding service element in the descriptor.

If the component's security attribute in the `<domain_name>_driver.xml` file is set to `iiopTls` or `tlsOnly`, and if the entity `<service_name>.tls.port` is defined (is a number and not zero), a secure endpoint element is created in the corresponding service element in the descriptor.

If no port entities can be found for a service (other than the management service) that is marked as direct persistent, an exception is thrown.

For the management service, the `<domain_name>_driver.xml` and `ABDriver.dtd` files may have specified this as an indirect persistent service, and therefore no non-zero IIOP ports for the management service are defined in `ABDriver.dtd`. Instead of throwing an exception, default endpoints elements are created in the descriptor (IIOP port `53086`, IIOP TLS port `53086`, HTTP port `53185`, HTTPS port `53186`). This is necessary because the management service in Orbix 6.x is always direct persistent.

Lastly, if the `manage_services` entity is defined in `ABDriver.dtd` and if its value is `true`, or if the `<service_name>.managed` entity is defined and its value is `true`, the corresponding service element in the descriptor is set to be managed.

Stage 4—Obtaining the address mode policy

The default behavior of the deployer towards address mode policies (whether hostnames or IP addresses used in IORs) is to use the unqualified host name, and to assume all services and components are to be deployed on the localhost. The name and IP address of the localhost are obtained by `InetAddress.getLocalHost()`.

If the `host.hostname_for_iors` entity is present in `ABDriver.dtd`, this default behavior is overwritten as follows:

- If the deployer fails to obtain the `InetAddress` of the host identified by the value of `host.hostname_for_iors` (`InetAddress.getByName()` throws an `UnknownHostException`), the conversion fails.
- Otherwise the converter creates a `dd:nodes` element in the descriptor, and sets its `dns` attribute set to the DNS domain name. This is obtained from the `InetAddress` object's hostname, after stripping off the first part of the name, so this may be an empty string.

For example, the following entry in `ABDriver.dtd`:

```
<!ENTITY host.hostname_for_iors = "orion.dublin.emea.iona.com">
```

results in: `<dd:nodes dns="dublin.emea.iona.com">`.

If the entity value is an IP address, or an unqualified host name, it depends on your network configuration whether a DNS name is specified.

Next, a `dd:node` element is created as a child of the `dd:nodes` element. The value for the `name` attribute of `dd:node` is obtained as the hostname member of the above `InetAddress` object, and the value for the ip attribute as the host address member of the InetAddress object. For example, the following entry in ABDriver.dtd:

```
<!ENTITY host.hostname_for_iors "10.2.1.101">
```

results in:

```
<dd:nodes>
  <dd:node name="orion" ip="10.2.1.101" profile="orion" />
</dd:nodes>
```

Rules for inferring the address mode policy

By comparing the value of the dns attribute (of dd:nodes), and the values of the name and ip attributes (of dd:node) with the original entity value, the address mode policy is inferred. If this is not short, it is stored as a dd:policy element under the dd:node element. The rules for this process are as follows:

- If the entity value is the literal localhost, the address mode policy is set to localhost.
- Otherwise, if the entity value is the literal 127.0.0.1, the address mode policy is set to localhost_ip.
- Otherwise, if the entity value matches the value for ip attribute on the dd:node element, the address mode policy is set to ip.
- Otherwise, if the entity value matches the string obtained by concatenating the value of the name attribute on the dd:node element with (a dot and) the value of the dns attribute of the dd:nodes element, the address mode policy is set to long.
- Otherwise, the address mode policy is short.

If the entity value specifies the IP address of the localhost, the value of the name attribute on the dd:node element may not be identical with the default name of the localhost. This is the case for example, if on the network, IP address 10.2.1.101 is known to belong to host orion, but the DNS resolution on orion has a different virtual name for this host (for example, orion-2).

Ensuring ORB name compatibility

By default, the value dd:node element's name attribute is used to determine host-qualified service ORB names. This may result in different ORB names in the 6.x domain than those in the 5.1 domain. To prevent this—and to allow for hostnames used in ORB names that are not the name of an existing host (5.1 accepted any string entered in the **What is the unqualified hostname?** text box)—the converter also checks if any of the following entities are defined:

```
cfr.orbname
locator.orbname
node_daemon.orbname
naming.orbname
```

To ensure ORB name compatibility between Orbix 5.1 and Orbix 6.x, the last part of the name in the value of the first entity found—if different from the `dd:node` element's name attribute—is also recorded as a policy under the `dd:node` element.

Example conversion

Assume the following contents of `c:\winnt\system32\drivers\etc\hosts` on host `orion` (IP address `10.2.1.101`):

```
127.0.0.1       localhost
orion2
```

and the following in the `ABDriver.dtd` file:

```
<!ENTITY host.hostname_for_iors "10.2.1.101">
<!ENTITY naming.orbname "iona_services.naming.orion">
```

In this case, `InetAddress.getByName("10.2.1.101").getHostName()` returns `orion2`.

And `InetAddress.getByName("10.2.1.101").getHostAddress()` returns `10.2.1.101`.

To ensure that in Orbix 6.x the same address mode policy and ORB names are used as were used in the Orbix 5.1 domain, the descriptor has the following entries:

```
<dd:nodes>
  <dd:node name="orion2" ip="10.2.1.101">
    <dd:policies>
        <dd:policy name="address_mode" value="ip" />
       <dd:policy name="hostname_for_orbs" value="orion"
   />
    </dd:policies>
  </dd:node>
</dd:nodes>
```

Conversion for virtual hosts

Changes in the conversion process for hostnames and address mode policies ensure that you can migrate 5.1 driver and entity files that used virtual hostnames/IP addresses. See "Deploying on Multi-homed Machines" on page 41 for more details.

One important difference however is that—while the actual conversion of the driver and entities files from a remote host may succeed as it did in Orbix 6.0.2—subsequent deployment can fail because services might not be able to communicate with each other. For example, a locator is prepared and subsequently started on the localhost (for example, `orion`), but when the node daemon is started it fails to communicate with the locator, which listens on a network address on the remote host. In practice, you should avoid such conversions, because they will not yield the expected results.

Note: All other entities (apart from those needed to resolve references in `<`*`domain_name`*`>_driver.xml`) are ignored. All path related entities (`<`*`service_name`*`>.bin.dir`, and the associated parameter entity `%binDir`) are ignored. Address list entities are ignored because the deployer reconstructs that information when processing the generated descriptor.

Migrating from Orbix 6.x

This section explains how to import existing Orbix 6.x service databases into an Orbix 6.3 domain. You can do this using the **Expert** mode of the **Orbix Configuration** tool. This enables you to initialize a new Orbix 6.3 domain with existing service database files.

Note: Only databases created using Orbix 6.0 SP1 or later can be imported.

Before you begin

You must ensure that all service databases that use the Persistent State Service (PSS) have been check pointed. This applies to the following services:

- locator daemon
- node daemon
- naming service
- interface repository (IFR)
- configuration repository (CFR)

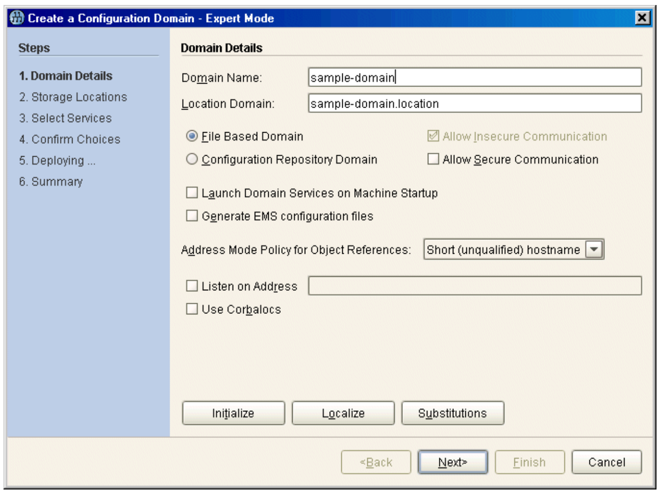
For details on check pointing databases, see the *Managing Orbix Service Databases* chapter in the *Orbix Administrator's Guide*.

Note: When importing data from a CFR-based domain, you must ensure that the ports used for the CFR, locator, and naming service remain unchanged.

Importing an Orbix 6.x descriptor

To import Orbix 6.x service database and log files into Orbix 6.3, perform the following steps:

1. Start the Orbix configuration tool using the `itconfigure` command.
2. Select **File**>**New**>**Expert** from the main menu. This displays the **Domain Details** screen shown in Figure 37.



Create a Configuration Domain - Expert Mode

Steps
1. Domain Details
2. Storage Locations
3. Select Services
4. Confirm Choices
5. Deploying ...
6. Summary

Domain Details

Domain Name: sample-domain
Location Domain: sample-domain.location

[✓ File Based Domain] Configuration Repository Domain

[✓ Allow Insecure Communication]

[ ] Allow Secure Communication

[ ] Launch Domain Services on Machine Startup

[ ] Generate EMS configuration files

Address Mode Policy for Object References: Short (unqualified) hostname

[ ] Listen on Address

[ ] Use Corbalocs

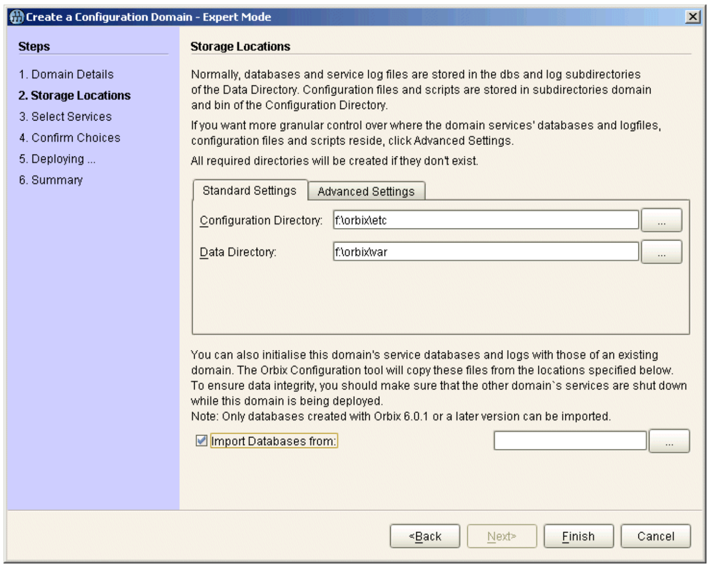
Figure 37: *Domain Details Screen*

3. Click **Initialize**, located at the bottom left of the screen. This opens a file selection dialog, which enables you to browse to the domain descriptor from your existing domain.
4. Click **Open** to select your existing domain descriptor.

Note: You must use the same location domain name and transports to ensure that indirect persistent object references continue to work. Services such as the security service or firewall proxy service should not be added or removed. In general, you should not make changes to the domain.

5. When you have initialized your domain, click **Next**. This displays the **Storage Locations** screen, shown in Figure 38.

6. Select the **Import Databases From** checkbox at the bottom left of the screen.



Storage Locations

Normally, databases and service log files are stored in the dbs and log subdirectories of the Data Directory. Configuration files and scripts are stored in subdirectories domain and bin of the Configuration Directory.

If you want more granular control over where the domain services' databases and logfiles, configuration files and scripts reside, click Advanced Settings.

All required directories will be created if they don't exist.

Configuration Directory: f:\orbix\etc

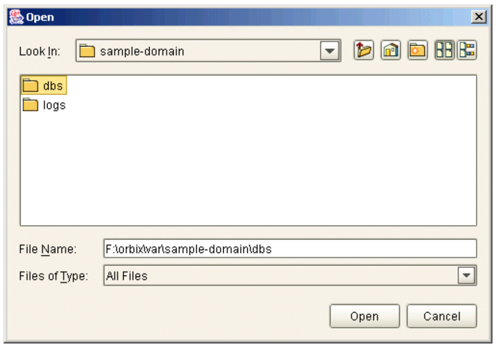
Data Directory: f:\orbix\var

You can also initialise this domain's service databases and logs with those of an existing domain. The Orbix Configuration tool will copy these files from the locations specified below. To ensure data integrity, you should make sure that the other domain`s services are shut down while this domain is being deployed.
Note: Only databases created with Orbix 6.0.1 or a later version can be imported.

☑ Import Databases from:

Figure 38: *Storage Locations Screen*

7. Click the button on the bottom right to browse to the location of the existing service database (`dbs`) directory. Click **Open**. Figure 39 shows an example.



Look In: sample-domain

File Name: F:\orbix\var\sample-domain\dbs

Files of Type: All Files

Figure 39: *Open dialog*

This initializes your new domain’s service database with those of the existing domain. These domain files are copied from the locations specified in the **Import Databases from** field.

Note: To ensure data integrity, all services in the existing domain must be shutdown when the new domain is deployed.

8. Click **Finish** to deploy the new domain based on your existing database and log files.

   Alternatively, click **Next**, and follow the steps in the wizard.

Orbix Deployment DTD

This appendix lists the supported DTD for the Orbix component XML templates. These XML template files are used to deploy Orbix components and services. The supported DTD is a subset of the complete DTD. Unsupported features are not documented.

Orbix Component Template Structure

The Orbix component XML template documents use a Document Type Definition (DTD) document to define the tags and values that make up the data and the structure the data takes. The DTD defining the configuration data is `ABDeploy.dtd`.

All XML documents used as source for an Orbix configuration must specify `ABDeploy.dtd` as its DTD, and conform to the structure it defines.

WARNING: The schema for the Orbix deployer XML files is not fully documented. Only a subset of the complete DTD is supported and documented. Unsupported features are not documented, and are subject to change without notice.

ABDeploy.dtd

Example 4 shows the subset the `ABDeploy.dtd` file that is supported by Micro Focus. This file defines the structure of configuration XML component templates.

Example 4: *The DTD defining Orbix configuration source documents.*

```
  <!-- Application Builder Data Deployment Definition -->

  <!ENTITY % ABDriver SYSTEM "ABDriver.dtd">
  <!ENTITY % DynamicDriver SYSTEM "dynamic_deploy.dtd">

1 <!ELEMENT ABDeploy (service?, process?, section*)>
  <!ELEMENT configData (dataType, (dataValue?)>
  <!ATTLIST configData
               scope CDATA #IMPLIED>
  <!ELEMENT dataId        (#PCDATA)>
  <!ELEMENT dataType      (#PCDATA)>
  <!ELEMENT dataValue     (#PCDATA)>

2 <!-- service  -->
  <!ELEMENT service (data)>

3 <!-- process  -->
  <!ELEMENT process (stage*)>
```

Example 4: *The DTD defining Orbix configuration source documents.*

```
4     <!ELEMENT stage  (source*)>
      <!ATTLIST stage
                  action ( populate | prepare | ntInstall | ntUninstall
                   | removeReplica | run | templates | filePopulate
                    | configPopulate | populateHandler ) #REQUIRED
                   store ( environment | properties | domain |
         bootDomain
                    | adminDomain | cfrDomain | imr | tmp ) "domain"
                  domain ( domain | cfrDomain | adminDomain ) "domain">
5     <!ELEMENT source        (file?, Dsection*)>
      <!ELEMENT file          (#PCDATA)>
6     <!ELEMENT Dsection      (#PCDATA)>
      <!ATTLIST Dsection
                   os NMTOKENS #IMPLIED
                   os_family ( unix | windows ) #IMPLIED
                   security ( iiopOnly | iiopTls | tlsOnly | is2_iiop |
                                        is2_semi | is2_tls) #IMPLIED

7     <!-- section  -->
      <!ELEMENT section ((configScope | configData)*)>
      <!ELEMENT configScope (dataId)>

      <!--   -->
      %DynamicDriver;
      %ABDriver;
```

The numbered elements in Example 4 are explained as follows:

1. `<ABDeploy>` is the root element of every Orbix configuration document. It must be the first tag and is required for the document to be valid.

   `<ABDeploy>` can contain one `<service>` element, `<process>` element, and any number of `<section>` elements.
2. `<service>` specifies information about a service that the deployer needs to deploy it.

   This element must be present in custom XML files to satisfy the more general syntax. Aside from using its `id` attribute to identify the custom component for your own documentation purposes, it is of no further relevance to custom XML files.
3. `<process>` specifies when and how certain `<section>` elements are processed. Can contain any number of `<stage>` elements.
4. A `<stage>` element can reference one or multiple `section`s that can reside in one or more XML files. A `<stage>` element has one or more `<source>` elements.

   The `action` attribute of the `stage` determines the target location of the configuration data processed in the `<stage>`. It decides where the configuration data specified in the `configData` elements in the stage's `section`s will be placed.

The attributes `store` and `domain` have default values.

> **Note:** The values `filePopulate`, `configPopulate` and `populateHandler` for attribute `action` are now deprecated. They are still supported to protect investment in any custom XML files that may have been written against the former DTD, and map internally to the following combination of values for the `action` and `store` attributes:
>
> ```
> filePopulate - action="populate" store="domain"
> configPopulate - action="populate" store="domain"
> populateHandler - action="populate" store="bootDomain".
> ```
>
> Newly written custom XML files should use these combinations instead of the deprecated values.

A custom XML file's `<stage>` element should rarely have more than one `<section>` child element.

5. `<source>` has an optional attribute file, the value of which, if specified, indicates in which XML file the sections referred to in the CDATA of the`<Dsection>` child elements can be found. Custom XML files most likely specify their sections locally, so this attribute is not needed.

   A `<source>` element can have one or several `<Dsection>` child elements.
6. A `<Dsection>` element is a reference to a set of `<configData>` elements which is itself contained in a `<section>`. The `Dsection`'s CDATA provides the mapping.

   It is an error if a `<Dsection>` element references a `<section>`that cannot be found (in the local file, or in the file denoted by its parent source element.

   While `DScection`s in XML files can have `constraint` attributes (meaning the data in the references sections is processed only if the constraint is met), custom XML files should not use these constraint attributes.
7. A `<section>` element is a container for a set of `<configScope>` and/or `<configData>` elements. It has one mandatory `name` attribute, which is used to map to `<Dsection>` elements appearing in the as child elements of a `<process>` element. A `<section>` element must contain at least one `<configScope>` or `<configData>` element.

   `<section>` elements are used to support multiple installation and configuration scenarios.

Summary

In practice the full complexity described in Example 4 is rarely needed. Most custom XML files will provide sufficient functionality if the following conditions are met:

- The `<process>` element contains one `<stage>` element (the `action` attribute of which is set to `configPopulate`).
- The `<stage>` element contains one `<source>` element, without the `file` attribute being set.

- The `<source>` element contains one `<Dsection>` element (without any attributes), and this `<Dsection>` element's CDATA is the same as the name of a `<section>` to be found further on in the document.
- The document contains one `<section>` element.
- The `<section>` element contains any number of `<configData>` elements.
- A `<configData>` element and its child elements hold the equivalent information to an `itadmin variable create` command—they specify variable scope, name, type and value(s).

Glossary

A

administration

All aspects of installing, configuring, deploying, monitoring, and managing a system.

ART

Adaptive Runtime Technology. A modular, distributed object architecture, which supports dynamic deployment and configuration of services and application code. ART provides the foundation for ORBIX software products.

ATLI2

Abstract Transpot Layer Interface, version 2. The current transport layer implementation used in ORBIX.

C

Certificate Authority

Certificate Authority (CA). A trusted third-party organization or company that issues digital certificates used to create digital signatures and public-private key pairs. The role of the CA in this process is to guarantee that the individual granted the unique certificate is, in fact, who he or she claims to be. CAs are a crucial component in data security and electronic commerce because they guarantee that the two parties exchanging information are really who they claim to be.

CFR

See configuration repository.

client

An application (process) that typically runs on a desktop and requests services from other applications that often run on different machines (known as server processes). In CORBA, a client is a program that requests services from CORBA objects.

configuration

A specific arrangement of system elements and settings.

configuration domain

Contains all the configuration information that Orbix ORBs, services and applications use. Defines a set of common configuration settings that specify available services and control ORB behavior. This information consists of configuration variables and their values. Configuration domain data can be implemented and maintained in a centralized Orbix configuration repository or as a set of files distributed among domain hosts. Configuration domains let you organize ORBs into manageable groups, thereby bringing scalability and ease of use to the largest environments. See also configuration file and configuration repository.

configuration file

A file that contains configuration information for Orbix components within a specific configuration domain. See also configuration domain.

configuration repository

A centralized store of configuration information for all Orbix components within a specific configuration domain. See also configuration domain.

configuration scope

Orbix configuration is divided into scopes. These are typically organized into a root scope and a hierarchy of nested scopes, the fully-qualified names of which map directly to ORB names. By organizing configuration properties into various scopes, different settings can be provided for individual ORBs, or common settings for groups of ORB. Orbix services, such as the naming service, have their own configuration scopes.

CORBA

Common Object Request Broker Architecture. An open standard that enables objects to communicate with one another regardless of what programming language they are written in, or what operating system they run on. The CORBA specification is produced and maintained by the OMG. See also OMG.

CORBA naming service

An implementation of the OMG Naming Service Specification. Describes how applications can map object references to names. Servers can register object references by name with a naming service repository, and can advertise those names to clients. Clients, in turn, can resolve the desired objects in the naming service by supplying the appropriate name. The Orbix naming service is an example.

CORBA objects

Self-contained software entities that consist of both data and the procedures to manipulate that data. Can be implemented in any programming language that CORBA supports, such as C++ and Java.

CORBA transaction service

An implementation of the OMG Transaction Service Specification. Provides interfaces to manage the demarcation of transactions and the propagation of transaction contexts. Orbix OTS is such as service.

CSIv2

The OMG's Common Secure Interoperability protocol v2.0, which can be used to provide the basis for application-level security in both CORBA and J2EE applications. The IONA Security Framework implements CSIv2 to transmit user names and passwords, and to assert identities between applications.

D

deployment
The process of distributing a configuration or system element into an environment.

H

HTTP
HyperText Transfer Protocol. The underlying protocol used by the World Wide Web. It defines how files (text, graphic images, video, and other multimedia files) are formatted and transmitted. Also defines what actions Web servers and browsers should take in response to various commands. HTTP runs on top of TCP/IP.

I

IDL
Interface Definition Language. The CORBA standard declarative language that allows a programmer to define interfaces to CORBA objects. An IDL file defines the public API that CORBA objects expose in a server application. Clients use these interfaces to access server objects across a network. IDL interfaces are independent of operating systems and programming languages.

IFR
See interface repository.

IIOP
Internet Inter-ORB Protocol. The CORBA standard messaging protocol, defined by the OMG, for communications between ORBs and distributed applications. IIOP is defined as a protocol layer above the transport layer, TCP/IP.

implementation repository
A database of available servers, it dynamically maps persistent objects to their server's actual address. Keeps track of the servers available in a system and the hosts they run on. Also provides a central forwarding point for client requests. See also location domain and locator daemon.

IMR
See implementation repository.

installation
The placement of software on a computer. Installation does not include configuration unless a default configuration is supplied.

Interface Definition Language
See IDL.

interface repository
Provides centralized persistent storage of IDL interfaces. An Orbix client can query this repository at runtime to determine information about an object's interface, and then use the Dynamic Invocation Interface (DII) to make calls to the object. Enables Orbix clients to call operations on IDL interfaces that are unknown at compile time.

invocation
A request issued on an already active software component.

IOR
Interoperable Object Reference. See object reference.

L

location domain
A collection of servers under the control of a single locator daemon. Can span any number of hosts across a network, and can be dynamically extended with new hosts. See also locator daemon and node daemon.

locator daemon
A server host facility that manages an implementation repository and acts as a control center for a location domain. Orbix clients use the locator daemon, often in conjunction with a naming service, to locate the objects they seek. Together with the implementation repository, it also stores server process data for activating servers and objects. When a client invokes on an object, the client ORB sends this invocation to the locator daemon, and the locator daemon searches the implementation repository for the address of the server object. In addition, enables servers to be moved from one host to another without disrupting client request processing. Redirects requests to the new location and transparently reconnects clients to the new server instance. See also location domain, node daemon, and implementation repository.

N

naming service
See CORBA naming service.

node daemon
Starts, monitors, and manages servers on a host machine. Every machine that runs a server must run a node daemon.

O

object reference
Uniquely identifies a local or remote object instance. Can be stored in a CORBA naming service, in a file or in a URL. The contact details that a client application uses to communicate with a CORBA object. Also known as interoperable object reference (IOR) or proxy.

OMG
Object Management Group. An open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications, including CORBA. See www.omg.com.

ORB
Object Request Broker. Manages the interaction between clients and servers, using the Internet Inter-ORB Protocol (IIOP). Enables clients to make requests and receive replies from servers in a distributed computer environment. Key component in CORBA.

OTS
See CORBA transaction service.

P

POA

Portable Object Adapter. Maps object references to their concrete implementations in a server. Creates and manages object references to all objects used by an application, manages object state, and provides the infrastructure to support persistent objects and the portability of object implementations between different ORB products. Can be transient or persistent.

protocol

Format for the layout of messages sent over a network.

S

server

A program that provides services to clients. CORBA servers act as containers for CORBA objects, allowing clients to access those objects using IDL interfaces.

SSL

Secure Sockets Layer protocol. Provides transport layer security—authenticity, integrity, and confidentiality—for authenticated and encrypted communications between clients and servers. Runs above TCP/IP and below application protocols such as HTTP and IIOP.

SSL handshake

An SSL session begins with an exchange of messages known as the SSL handshake. Allows a server to authenticate itself to the client using public-key encryption. Enables the client and the server to co-operate in the creation of symmetric keys that are used for rapid encryption, decryption, and tamper detection during the session that follows. Optionally, the handshake also allows the client to authenticate itself to the server. This is known as mutual authentication.

T

TCP/IP

Transmission Control Protocol/Internet Protocol. The basic suite of protocols used to connect hosts to the Internet, intranets, and extranets.

TLS

Transport Layer Security. An IETF open standard that is based on, and is the successor to, SSL. Provides transport-layer security for secure communications. See also SSL.

Index