OpenFusion Implementation Repository

Version 1.0

User Guide

PRISMTECH

OpenFusion
Implementation Repository

User Guide

PrismTech

Part Number: OFIMR-IMRG

Doc Issue 15, 11 April 2006

Notices

Copyright Notice

Preface

About the OpenFusion ImR Guide

The *OpenFusion ImR Guide* is included with the OpenFusion Implementation Repository product. The OpenFusion Implementation Repository (OF ImR for short) can be used with either or both of the TAO and JacORB ORB implementations: it provides a simpler, more integrated IMR solution than other, previously available Implementation Repository implementations.

The *OpenFusion ImR Guide* is intended to be used with the standard documentation provided with either the OpenFusion TAO or OpenFusion JacORB products, as appropriate.

Intended Audience

The *OpenFusion ImR Guide* is intended to be used by users and developers who wish to use OpenFusion Implementation Repository with TAO or JacORB. Readers who use this guide should have a good understanding of the ORBs and programming languages they are using (such as Java, C++, or IDL, for example), plus understand implementation repository concepts and principles.

Organisation

The *OpenFusion ImR Guide* is organised into the following sections:

- an *Introduction* which describes the OpenFusion Implementation Repository's features and benefits
- *Using the OF ImR* describes how to use the OF ImR's command line utilities to run, configure and use the OF ImR

Conventions

The conventions listed below are used to guide and assist the reader in understanding the OpenFusion ImR Guide.

Item *Under Construction* and subject to change.

Item of special significance or where caution needs to be taken.

Item contains helpful hint or special information.

WIN Information applies to Windows (e.g. NT, 2000) only.

UNIX Information applies to Unix based systems (e.g. Solaris) only.

Hypertext links are shown as *blue italic underlined.*

On-Line (PDF) versions of this document: Items shown as cross references to other parts of the document, e.g. *Contacts* on page v, behave as hypertext links: users can jump to that section of the document by clicking on the cross reference.

```
%  Commands or input which the user enters on the
   command line of their computer terminal
```

`Courier`, **`Courier Bold`**, or *`Courier Italic`* fonts are used to indicate programming code. The `Courier` font can also be used to indicate file names (in order to distinguish the file name from the standard text).

Extended code fragments are shown as `Courier` font in shaded boxes:

```
NameComponent newName[] = new NameComponent[1];

// set id field to “example” and kind field to an empty string
newName[0] = new NameComponent (“example”, “”);

rootContext.bind (newName, demoObject);
```

Italics and ***Italic Bold*** indicate new terms, or emphasise an item.

Arial Bold indicates user related actions, e.g. **File | Save** from a menu.

Step 1: One of several steps required to complete a task.

Contacts

PrismTech can be contacted at the following contact points for information and technical support.

Corporate Headquarters	**European Head Office**
PrismTech Corporation 6 Lincoln Knoll Lane Suite 100 Burlington, MA 01803 USA	PrismTech Limited PrismTech House 5th Avenue Business Park Gateshead NE11 0NG UK
Tel: +1 781 270 1177 Fax: +1 781 238 1700	Tel: +44 (0)191 497 9900 Fax: +44 (0)191 497 9901

Web:	*http://www.prismtech.com*
General Enquiries:	*info@prismtech.com*

Contents

Table of Contents

List of Tables

Introduction

Introduction

Background and General Description

An Implementation Repository plays a key role in building scalable, flexible CORBA systems: it helps clients to bind requests to the appropriate object implementations.

Implementation Repositories (IMR) can bind CORBA clients to *persistent* CORBA servers and their associated persistent Interoperable Object References (IOR): the IMR enables these servers and their object implementations to be located when a server is stopped and subsequently restarted - even if the server is started on a different host.

The Object Management Group (OMG) does not provide a standard for Implementation Repositories. Consequently, specific IMR features my vary between different vendors. Nonetheless, an IMR must guarantee client side interoperability between different vendors ORBs: a client using one vendors ORB can make requests to a server registered with another vendors IMR. However, servers can only use or be registered with their own ORB's Implementation Repository since each ORB uses proprietary mechanisms to communicate with the IMR.

The OpenFusion Implementation Repository (OF ImR) can be used seamlessly by servers developed using either or both of PrismTech's TAO or JacORB ORBs. In addition, the OF ImR provides advanced features over alternative IMRs for improved usability, reliability, availability and scalability, including support for load balancing, fail-over and auto-activation of servers.

Features

The OpenFusion Implementation Repository provides the features listed below. These features are generally available to both OpenFusion TAO and OpenFusion JacORB[1]:

- a suite of command line utilities for running, configuring and managing OF ImR instances
- binding client requests, to appropriate object implementations, for persistent object IORs

1. See *Using the OF ImR* on page 7 for details of specific features supported on particular ORBs for this release.

- persistence of configuration information using file-based persistence plug-ins
- automatic server activation, including activation of servers on remote hosts (i.e. servers located on a different host to the ImR)
- co-location of the ImR and Activator in a single process
- configurable load balancing
- fail-over protection for ImR instances and stateless replica servers

The OpenFusion ImR

1 Using the OF ImR

1.1 Overview

Specific Features

The OpenFusion Implementation Repository provides the specific features listed below. Most of these features are available on both TAO and JacORB. However some features are currently only available on one ORB: these are shown with a symbol, **TAO** or **JacORB**, indicating which ORB it is available on.

- command line utilities for running, configuring, and managing OF ImR instances
- binding client requests to appropriate object implementations for persistent object IORs
- more than one OF ImR instance can be run on a single host: servers can register with a specific ImR instance
- variable level, ORB driven logging facility and support
- configurable server checking to ensure that a server is available when a client sends a request
- configurable ImR checking to ensure that an ImR continues to be available to a registered server
- automatic, configurable registration with, and discovery of, an ImR by CORBA servers that create persistent IORs
- automatic, configurable activation of shared servers
- automatic activation of servers on remote hosts (i.e. for servers located on a different host to the ImR)
- configurable timeout duration of automatic server activation for determining if a server has started or not
- automatic activation of registered servers
- persistence of configuration information using file-based persistence
- **JacORB** configurable load balancing
- **JacORB** fail-over protection for ImR instances through automatic discovery and re-registration with other ImR instances running on the same or different hosts

JacORB
- basic fail-over protection for stateless replica servers

TAO
- automatic activation of non-shared servers

TAO
- co-location of the ImR and Activator processes when they are running on the same host

Command Line Utilities

The OF ImR provides command line utilities for starting, configuring and managing the ImR, servers, load balancing and other features.

i

Two sets of matching utilities are provided: one set is for use with TAO, the other set is for use with JacORB. The utilities use a naming convention whereby each utility name contains a prefix which corresponds to the ORB they are used with, namely *`tao_`* for the TAO ORB and *`jac_`* for the JacORB ORB. For example, the version of ImR Launcher utility which is used with TAO is called *`tao_imrd`*; the version used with JacORB is *`jac_imr`*. Aside from the difference of naming prefix, the usage and options are identical for each version.

The utilities include:

- **ImR Launcher** (*`jac_imr`*, *`tao_imrd`*) - starts and configures the ImR
- **Implementation Repository Manager** (*`jac_imr_mgr`*, *`tao_imrd_mgr`*) - provides information about the ImR and is used to register, remove, and deactivate servers.
- **Activator Launcher** (*`jac_activator`*, *`tao_activator`*) - enables servers to be automatically activated for use with JacORB or TAO clients, respectively
- **Activator Manager** (*`jac_activator_mgr`*, *`tao_activator_mgr`*) - adds and removes start-up commands associated with specific servers, as well as starting a server after the associated command is registered and obtaining details of registered commands
- **ImR Locator** (*`jac_locator`*) - locates and displays details of any ImR running in the domain on a given UDP port

JacORB

In addition, the OF ImR can also be configured on JacORB by setting property values in JacORB's *`jacorb.properties`* file.

PrismTech

1.2 Running and Configuration

The OpenFusion Implementation Repository is started with the ImR Launcher command line utility, ***jac_imr*** or ***tao_imrd***, as appropriate. The ImR is configured by using either the ImR Launcher's command line options or, when used with JacORB, by setting property entries in the ***jacorb.properties*** file or using a combination of the two.

JacORB

The `jacorb.properties` file is located in the *<install>/classes* directory, where *<install>* is the OpenFusion installation directory. Table 6, *JacORB ImR Configuration Properties*, on page 23 lists the properties.

Performing general management tasks, adding servers, setting the load balancing policy, and other tasks are performed by the remaining utilities (as listed above under *Command Line Utilities* on page 8). All of the utilities and how to perform the configuration tasks associated with them are described below.

On-line Help is available for each utility by running it with the *-h* or *--help* option, for example:

```
%  jac_imr -h
```

i

Please note that although all examples show the *jac_** version of the utility, the command syntax shown should also be used for the *tao_** version and is identical to it.

ImR Launcher

The ImR Launcher (*jac_imr*, *tao_imrd*) starts the ImR, and optionally configures it, using:

```
%  jac_imr [options...]
```

where *[options...]* is one or more of the options described in *Table 1*.

JacORB

The JacORB *jacorb.properties* file property associated with each option is shown in {}'s underneath the command line option.

Table 1 ImR Launcher

Option	Description
`-p, --port <number>` `{jacorb.imr.port_number}`	The fixed port number which the ImR will use.The default number is not set and the port number for the POA is assigned by the system.
`-i, --imr_ior <filename>` `{jacorb.imr.ior_file}`	A file where the ImR's IOR is to be stored.
`-a, --aliveness <value>` `{jacorb.imr.aliveness_policy}`	Determines when the ImR should ping servers to decide if they are active. The `aliveness_policy` can be set to: JacORB ***0*** (`PING`) - The ImR will *ping* each live server, a server will ping the ImR, or both, at regular intervals as specified by the *heartbeat* property. If an exception occurs when the ImR is pinging, then it is assumed that the server is no longer active and it is de-registered within the ImR. ***1*** (`ON LOOKUP`) - The ImR will ping the selected server (selected after application of any load balancing policy), when a request to locate a specified object is received. If an exception occurs, indicating that the server is no longer active, then the server may be reactivated if it's auto-activation flag is set to true and the Start-up Daemon is running. **Note**: for this release the auto-activation can only be applied if just one server exists; if multiple servers exist, then auto-activation is not allowed. JacORB ***2*** (`ON EXPIRY`) - An activity table is updated to record the time a server last pinged the ImR. The ImR will check the activity table at regular intervals according to the value of the timeout property[a]. If the timeout has elapsed and the server has not pinged the ImR, then the ImR pings the server. It is assumed that the server is no longer active if a failure occurs and the server is de-registered within the ImR. ***3*** (`NONE`) - The ImR will not ping it's servers (no ping mechanism is specified). The default value for the `aliveness_policy` is ***3*** (`NONE`).

Table 1 ImR Launcher (Continued)

Option	Description
`-t, --timeout <value> {jacorb.imr.active_server_ timeout or jacorb.imr.heartbeat}`	The *`--timeout`* option performs different tasks depending on the value of the *`aliveness_policy`* (see the *`--aliveness`* option above). This option either: • sets the frequency (in milliseconds) that the ImR checks the *activity table* when and only when the *`aliveness_policy`* is set to *ON EXPIRY* or • sets the frequency (in milliseconds) of the *`heartbeat`* property. The *`heartbeat`* property is the frequency that a the ImR pings a server and/or a server pings the ImR. The default `active_server_timeout` value is 120000 milliseconds. The default `heartbeat` value is 0 (zero). Servers will ping when the heartbeat value is greater than 0 (zero). ImRs will ping when the heartbeat value is greater than 0 (zero) **and** the *aliveness* policy for the ImR is set to ping. The *`-t`* command line option sets the heartbeat for the **ImR only**, if the *aliveness* policy is set to ping. The `jacorb.imr.heartbeat` property entry sets the heartbeat for servers and is also used by the ImR if the *`aliveness_policy`* is set to ping (0) and the *`-t`* command line option has **not** been set. **Note**: • if the ImR is set to ping and `heartbeat` is 0, then the ImR cannot ping and will throw an error • if the `aliveness_policy` is set to a value greater than 0, then `heartbeat` is ignored • if the servers are not required to ping the ImR and the ImR aliveness policy is set to ping (*`0`*), then the heartbeat for the ImR can be set using the *`-t`* command line option when the ImR is started

Table 1 ImR Launcher (Continued)

Option	Description
`-u, --udp_port <number>` `{jacorb.imr.udp_port}`	The port number which the multicast server will use. The multicast server is used to locate any ImRs running within the domain. **Note**: *udp_port* must be set before the multicast server can be started: if it is not set, then a multicast server will not run. TAO Servers which wish to register with the ImR must be able to find it. If the ImR's default port number (10018) is not used, by setting *--udp_port* to 1 (one), then the server's ORB must be explicitly given the port number using either of the following methods: 1) From the command line using the server's *-ORBUseOFIMR* and *-ORBImplReposServicePort* command line options in conjunction with tao_imrd, e.g. *tao_imrd --udp_port 12345* *my_server -ORBUseOFIMR 1 -ORBImplReposServicePort 12345* where 12345 is the example assigned port number 2) Set the *ImplRepoServicePort* environment variable to the port number. Note that the setting the port number via the command line method will override the value set in *ImplRepoServicePort*.

Table 1 ImR Launcher (Continued)

Option	Description
`-s, --auto <on | off>` `{jacorb.imr.allow_auto_register}`	Indicates whether a server can be registered automatically with the ImR or if it must be pre-registered before it can be activated. If *`allow_auto_register`* is not set or set to *`off`*, then all servers must be pre-registered using the *`ImR Manager`* utility. The default value is *on*. If multiple servers with the same name are required, then the server name must be registered, with the required load balancing policy, using the *`ImR Manager`* utility (see *ImR Manager* on page 13), before the actual servers can be registered. If a server is not pre-registered and *allow_auto_register* is set to on, then a flag will be set preventing registration of any additional servers at the point when the server is registered with the ImR.
`-v, --activator_timeout <value>` `{jacorb.imr.activator.start_timeout}`	Sets the length of time (in milliseconds) that the ImR will wait for the activator to start a server. The default value is *`10000`*.
`-f, --imr_data <filename>` `{jacorb.imr.imr_data_file}`	The filename where the repository data is stored when using file-based persistence.
`-b, --imr_backup <filename>` `{jacorb.imr.imr_backup_file}`	The filename where repository data is stored on shutdown of the ImR when using file-based persistence.
`-n, --imr_new`	If provided, data will ***not*** be loaded on ImR start-up, from persistent storage.
`-h, --help`	Displays descriptions of the options listed in this table.

a.{*`jacorb.imr.active_server_timeout`*}

ImR Manager

The ImR Manager (*`jac_imr_mgr`*, *`tao_imrd_mgr`*) provides information about the ImR and is used to register and remove server names. Note that the ImR must be running for the ImR Manager to work. The manager is run using:

```
%  tao_imrd_mgr [options...]
```

where *[options...]* is one or more of the options and sub-options listed in *Table 2*. These options are described, in detail, under separate headings after the table.

Table 2 ImR Manager

Options and Sub-options	Description
`-ORBInitRef ImplementationRepository=<ior>`	Standard method of passing the ImR's IOR to the ImR Manager, where *<ior>* is the ImR's IOR. The IOR can be in a file, its location specified as, for example, *file://imr.ior*.
`-a, --add`	Registers (adds) a server name using the configurations set by the sub-options shown below.

`-e, --mode <mode>`	The activator mode, where
	`0 = Normal`
	`1 = Manual`
	`2 = Per_Client`
	`3 = Auto_start`

`-n, --name <name>`	JacORB Name of the server to be added (used with JacORB servers only)
`-p, --poa <name>`	TAO POA name of the server to be added (used with TAO servers only).
`-m, --multiples <on | off>`	Sets whether multiple servers assigned with the same server name are allowed (*ON*) or not (*OFF*). The default is OFF, i.e. multiple servers are not allowed.
`-l, --lb_policy <policy>`	The load balancing policy to be used for servers identified by the server name (*-name <name>*). The policy values, <*policy*>, are: **0** - Random **1** - Round Robin **2** - FIFO (First In First Out) **3** - LIFO (Last in First Out) **Note**: • if the policy (*-l*) is not set and allowing multiple servers (*-m*) is set to *ON*, then the default policy of Random (0) is used • if the policy (*-l*) is set and the allowing of multiple servers (*-m*) is not set or set to *OFF*, then the policy will be ignored since load balancing is impossible with only one server

PrismTech

Table 2 ImR Manager (Continued)

Options and Sub-options	Description
`-v, --act_ior <ior>`	The IOR of the Activator to be used to start server under this name.
`-d, --deactivate`	Deactivates all of the POAs for a running server.
`-s, --server <id>`	The id of the server.
`-k, --shutdown`	Shuts down a server name using the configurations set by the sub-options shown below.
`-n, --name <name>`	JacORB Name of the server to be shutdown (used with JacORB servers only)
`-p, --poa <name>`	TAO POA name of the server to be shutdown (used with TAO servers only).
`-s, --server <id>`	The id of the server which is to be shutdown. This option is only required when more than one instance of a named server is running.
`-i, --imr`	JacORB Remotely shuts down the ImR process
`-l, --list`	Lists names or POAs which are associated with the ImR. This option will list all registered servers when is it used without a sub-option.
`-n, --name <name>`	JacORB Lists all registered servers identified by <*name*> (used with JacORB servers only).
`-p, --poa <name>`	TAO Lists all POAs for the server identified by *--server <id>* (used with TAO servers only).
`-s, --server <id>`	Lists all POAs for the server identified by *--server* <id>
`-v, -activators`	Lists all registered Activators.
`-r, --remove`	Removes registration details specified by sub-option
`-n, --name <name>`	JacORB Identifies the name of the server (used with JacORB servers only).
`-p, --poa <name>`	TAO Identifies the server's POA (used with TAO servers only).
`-h, --help`	Displays descriptions of the options listed in this table.

Server Registration and Configuration

Server names can be registered (also referred to as being *added*) and configured using the ImR Manager's --*add* option. Configuration of load balancing policy, auto-activation and allowing the use of multiple servers is set when the server name is registered.

Server names are registered and configured using the *add* option of the *jac_imr_mgr* or *tao_imrd_mgr* command, as appropriate:

```
%  jac_imr_mgr --add <-n <name>> [-m <ON | OFF>]
   [-l <policy>] [-v <ActivatorIOR>]
```

where:

<> indicates mandatory items, *[]* indicates optional items (see *Table 2* for list of options)

If auto-activation is required, an Activator must be provided and *multiples* must be set to *OFF*.

Auto-activation can only be applied when the server is registered and if:

- there is only one server (auto-activation with multiple servers is not supported in this release),
- the Activator IOR for the server has been registered
- the *Activator Launcher* is running at the location provided by the Activator IOR
- the start-up command for the server has been registered with the Activator and
- the start-up mode is *auto-start*.

If an Activator IOR is provided and multiple servers are set to be allowed, then auto-activation will be ignored.

Example 1

To register a server name of *NotificationService*, allowing multiple servers, using a Round Robin load balancing policy, and disabling auto-activation use:

```
%  jac_imr_mgr add -n OpenFusion.NotificationService -m ON
-p 1
```

Example 2

To register a server name of *NotificationService* for a single server and enabling auto-activation use:

```
%  jac_imr_mgr add -n OpenFusion.NotificationService -v
IOR:..... -e 3
```

⚠ If multiple servers are allowed, then the server name must be registered ***before*** the server is started. When registering the server name, it must be specified that multiple servers are allowed, along with the load balancing policy that is to be used. Individual servers can also be registered.

Load balancing will only be applied if multiple servers are allowed. If multiple servers are not allowed, then only one server can be registered and returned (by the ImR).

When a server which is registered with the ImR becomes inactive, a second server can be registered without <*multiples*> being set to *true*: the second server overwrites the first server in this situation.

Deactivating a Server Name

All of the POAs for a running server can be deactivated with the ImR Manager's *deactivate* command and any of the server's POA names:

```
%  jac_imr_mgr --deactivate <-s <server_id>>
```

Removing a Server Name

A server name can be removed (de-registered) using the jac_imr_mgr's or tao_imrd_mgr's *remove <name>* option, where <*name*> is the server name to be removed:

```
%  jac_imr_mgr --remove <-n <example_server>>
```

Obtaining Information About the ImR

The ImR Manager can provide a list of server names, servers, POAs and Activators which are associated with the ImR. For example:

```
%  jac_imr_mgr --list
```

Activator Launcher

The Activator Launcher (*jac_activator*, *tao_activator*) allows automatic activation of a server. The activation is based on the start up command linked with the server. The start up command is registered using the Activator Manager.

Table 3 Activator Launcher

Option	Description
`-i, --act_ior <filename>` `{jacorb.imr.activator_ior_file}`	The location of the file used to store the Activator's IOR.
`-f, --act_data <filename>` `{jacorb.imr.act_data_file}`	The location of the file used to store the data held by the Activator where persistence is file-based.
`-b, --act_backup <filename>` `{jacorb.imr.act_backup_file}`	The location of the backup file used to store the data held by the Activator on shutdown when using file-based persistence.
`-n, --act_new`	If provided, then persisted data will not be loaded from the backup file.
`-h, --help`	Displays descriptions of the options listed in this table.

JacORB

The Activator's IOR is referenced by the *ORBInitRef.IMRActivator* property in the jacorb.properties file: the Activator Manager will not work if this property is not set.

Single Process ImR and Activator

TAO

The ImR and Activator can be run in a single process using the *tao_imrd_activator* command.

tao_imrd_activator uses the same command line options as the ImR Launcher and Activator Launcher, with the exception of the *-i* (*--act_ior*) and *-f* (*--act_data*) options.

Regards these two options, the long *GNU*-style options must be used instead to identify which of the two co-located components the option is targeted at. For example:

```
% tao_imrd_activator --act_ior activator.ior --imr_ior
imr.ior --imr_data ./imr_data.file --act_data
./act_data.file
```

PrismTech

Activator Manager

The Activator Manager (*jac_activator_mgr*, *tao_activator_mgr*) adds and removes start-up commands associated with specific servers. A server can also be started using the Activator Manager. The Start-up Activator must be running for this tool to work.

JacORB Unless the *-ORBInitRef* option is used (see *Table 4* below), the *ORBInitRef.IMRActivator* property in the jacorb.properties file must be set with Activator's IOR or the Activator Manager will not work.

The Activator Manager uses the command line options shown in *Table 4*.

Table 4 Activator Manager

Options and Sub-options	Description
`-ORBInitRef ImRActivator=<ior>`	Standard method of passing the Activator's IOR to the Activator Manager, where *<ior>* is the Activator's IOR. The IOR can be in a file, its location specified as, for example, *file://activator.ior*.
`-l, --list`	Lists registered commands.
`-a, --add`	Adds a new command using the sub-options.
`    -n, --name <name>`	JacORB Identifies the server name to use.
`    -p, --poa <name>`	TAO Identifies the POA name to use.
`    -c, --`*`command <command>`*	Specifies the command to use.
`    -f, --comm_file <filename>`	JacORB Uses the command stored in <filename>. This option is an alternative to the *-c* option. The `-f` option must be used when the command string is longer than the maximum length allowed by the operating system, such as on Windows NT and 2000 systems when the command string is greater than 2K. See Appendix A, *Using a Command File*, on page 29 for instructions on creating and using a command file.
`-r, --remove`	Removes the server identified by the `--`*`name`* sub-option.
`    -n, --name <name>`	JacORB Identifies the server name to remove.
`    -p, --poa <name>`	TAO Identifies the POA name to remove.
`-s, --start`	Starts the server identified by the `--`*`name`* sub-option.
`    -n, --name <name>`	JacORB dentifies the server name to start.

Table 4 Activator Manager (Continued)

Options and Sub-options	Description
`-p, --poa <name>`	TAO Identifies the POA name to start.
`-h, --help`	Displays descriptions of the options listed in this table.

Alternative Activator Location Method

TAO An alternative method to using

tao_activator_mgr `-ORBInitRef ImRActivator=file://activator.ior`

is to set the *ImRActivatorIOR* environment variable with the Activator's IOR object reference, noting that using *tao_activator_mgr* from the command line will override the value set in `ImRActivatorIOR`.

Activating a Server Name

If an auto-activate command has been registered for a server, then the server can be activated by using the `jac_activator_mgr`'s or `tao_activator_mgr`'s *--start* command:

```
%  jac_activator_mgr --start <-n <example_server>>
```

ImR Locator

JacORB The ImR Locator utility (*jac_locator*) locates and displays details of any ImR running in the domain on a given UDP port (jacorb.imr.udp_port).

The *jac_locator* utility uses the command line options shown in *Table 4*.

Table 5 ImR Locator

Option	Description
`-u, --udp_port <port>`	Attempts to locate the ImR running on the port number specified by *port*.
`-h, --help`	Displays a help screen.

JacORB JacORB can use the port number value set in the *jacorb.properties* file if the port number is not specified with *--udp_port* command line option.

Server Auto-activation Example

A server can be auto-activated by following the steps shown below.

i Note that either the Activator or the ImR can be started first, however they both must be started ***before*** their respective managers are used. Similarly, the Activator must be running in order to register the Server with the ImR.

Step 1: Start the Activator.

```
%  jac_activator
```

Step 2: Register a start up command for the Server using the Activator Manager.

```
%  jac_activator_mgr -a -n <server name> -c <start up
   command>
```

Where

- `server name` is the name of the server, for example `StandardNS`, for the JacORB `NameService`
- `start up command` is the command that is used to start the server, for example `jaco org.jacorb.naming.NameServer`, and including any command line parameters that `NameServer` takes, if required

  WIN If running on a Windows NT and 2000 platforms, then the entire start up command must be entered either:

  a) on the command line, but only if the command is less than 2K long, or

  b) using the `-f` or `--comm_file` command line option and using a file containing the start up command: this option can be used when the start up command is greater than 2K in length.
- `mode` is the activation mode for JacORB servers and will be auto-activated if and only if the mode is set to `3`

If the command line string exceeds the maximum length allowed by the operating system, then a command file must be used instead of using a start up command on the command line (with the `-a` option). For example, the start up will fail on Windows NT and 2000 systems if the start up command string is greater than 2K and is used with the `-a` option and is entered on the command line: in these situations the `-f` option must be used with a file containing the required command string. See Appendix A, *Using a Command File*, on page 29 for instructions on creating and using a command file.

Step 3: Start the ImR.

```
%  jac_imr
```

Step 4: Register the Server with the ImR

```
%  jac_imr_mgr -a -n <server name> -v <Activator IOR> -e
<mode>
```

Where

- *server name* is the name of the server, for example *StandardNS* for the JacORB *NameService*
- *Activator IOR* is the IOR of the Activator started at *Step 1:*
- *mode* is the activation mode for JacORB servers and will be auto-activated if and only if the mode is set to *3*

Configuring Failover Scenarios

JacORB

There are several possible failover scenarios that may be configured. This section gives a brief overview of each one.

Automatic Server Restart

To ensure that another server is started in the event of a server dying, set the *jacorb.imr.aliveness_policy* property (see *Table 6*) to *LOOKUP*. This policy will allow the IMR to constantly check that the server is running.

Automatic IMR Switching

There are two methods for allowing a server to automatically switch to another running IMR:

- To allow the server to switch to any IMR running within the same subnet, use the *jacorb.imr.udp_port* property (see *Table 6*).
- To allow the server to switch to an IMR running outside the server's subnet, use the *jacorb.imr.other_imrs* property (see *Table 6*) to preset a list of known IMRs. If you preset a list of known IMRs with IOR references, it is essential that those IORs do not change if the IMR is restarted. To accomplish this it is recommended that a unique port (*jacorb.imr.port_number*) and a unique identifier (*jacorb.imr.identifier*) are set for each IMR.

In a failover situation, the value of the *jacorb.imr.other_imrs* property takes precedence.

Automatic Server Switching

By configuring multiple servers and a suitable load balancing policy within the IMR manager it is possible to distribute any client calls between the different servers. If a server does go down, it will be transparent to the client as the remaining servers will be used to manage the calls.

1.3 JacORB ImR Properties

JacORB

Table 6 lists JacORB's ImR configuration properties.

Table 6 JacORB ImR Configuration Properties

Property	Description	Type
`jacorb.use_imr`	Switch on to contact the Implementation Repository (IMR) on every server start-up. Default is off.	boolean
`jacorb.use_imr_endpoint`	Switch off to prevent writing the IMR address into server IORs. This property is ignored if *jacorb.use_imr = off*. Default is off.	boolean
`jacorb.imr.allow_auto_register`	If set to on servers that don't already have an entry on their first call to the IMR, will get automatically registered. Otherwise, an `UnknownServer` exception is thrown. Default is off.	boolean
`jacorb.imr.check_object_liveness`	If set on the IMR will try to ping every object reference that it is going to return. If the reference is not alive, then `TRANSIENT` is thrown. Default is off.	boolean
`ORBInitRef.ImplementationRepository`	The initial reference for the IMR.	URL
`jacorb.imr.table_file`	File in which the IMR stores data.	file
`jacorb.imr.backup_file`	Backup data file for the IMR.	file
`jacorb.imr.ior_file`	File to which the IMR writes its IOR. This is usually referred to by the initial reference for the IMR (configured above).	file

Table 6 JacORB ImR Configuration Properties (Continued)

Property	Description	Type
`jacorb.imr.timeout`	Time in milliseconds that the implementation will wait for a started server to register. After this timeout is exceeded the IMR assumes the server has failed to start. Default is 12000 milliseconds (2 minutes).	millisec.
`jacorb.imr.no_of_poas`	Initial number of POAs that can be registered with the IMR. This is an optimization used to size internal data structures. This value can be exceeded. Default is 100.	integer
`jacorb.imr.no_of_servers`	Initial number of servers that can be registered with the IMR. This is an optimization used to size internal data structures. This value can be exceeded. Default is 5.	integer
`jacorb.imr.host`	Host for IMR.	node
`jacorb.imr.port_number`	Starts the IMR on a fixed port (equivalent to the -p option).	integer
`jacorb.imr.endpoint_port_number`	Port number for IMR endpoint.	integer
`jacorb.imr.connection_timeout`	Time in milliseconds that the IMR waits until a connection from an application client is terminated. Default is 2000.	millisec.

`jacorb.imr.aliveness_policy`	If or when the ImR is to check whether registered servers are still alive values:
	`0 = PING`
	`1 = ON LOOKUP`
	`2 = ON EXPIRY`
	`3 = NONE.`
	Default is 3.
		integer

`jacorb.imr.udp_port`	Port on which the multicast server is run.	integer.
`jacorb.imr.heartbeat`	Heartbeat interval.	millisec.
`jacorb.imr.active_server_timeout`	Active server timeout interval.	millisec.
`jacorb.implname`	The implementation name for persistent servers. (See the *JacORB Implname and CORBA Objects* section of the *Programming Guide*).	name
`jacorb.java_exec`	Command used by the IMR to start servers.	command

PRISMTECH

Table 6 JacORB ImR Configuration Properties (Continued)

Property	Description	Type
`jacorb.imr.identifier`	Uses the *`USER_ID`* POA policy to allow the user to generate unique, constant identifiers for the IMR. If this property is not set, the *`SYSTEM_ID`* POA policy is used to generate random unique information.	string
`jacorb.imr.other_imrs`	A comma-separated list of file URLs or IOR values, used to hard code other IMRs. This is useful if UDP is not being used or the IMRs are configured across subnets. This property takes precedence over the *`udp_port`* property and therefore will be checked first.	list of file URLs and IORs

Appendices

Appendices Section

Appendix A

Using a Command File

When using the Activator Manager there can be situations where the length of the start up command string exceeds the length that a particular operating system can cope with. For example, Windows NT and 2000 platforms can not cope with command line strings which are longer than 2k (see *Activator Manager* on page 19).

This restriction can be overcome by placing the start up command string into a file, then using the *-f* (*--comm_file*) option to pass the file, and the command it contains, to the Activator Manager.

Obtaining a Command String

The start up command string which is used by the Activator Manager to start an OpenFusion service can be obtained by using one of the alternative methods, *Using the Admin Manager* or *Using the jaco Batch File*, shown below.

Using the Admin Manager

The following the steps describe how to use the Admin Manager to obtain the start up command string.

Step 1: Install OpenFusion, if it is not already installed.

Step 2: Start the *admin manager* GUI or use *adminMgrTool* on the command line and set the logging level to debug for the particular service that you need.

Step 3: Start the service at the command line using *server -start* and piping the output to a file. For example:

```
%  server -start NotificationService > command.log
```

The information piped to the *command.log* file by the server command is shown below under *Example 1*. This start command strings appear in the output as a series of *command args[]*. The start up command which will be used by the Activator Manager is constructed by concatenating these `command args[]` into a single string, as shown under *Example 2*.

Example 1 Server command output

```
[configuration loaded from classpath resource
file:/D:/PrismTech/OpenFusion/classes/jacorb.properties]
0 [main] DEBUG root  - Starting service: NotificationService
110 [main] DEBUG root  - Command args[0] = "C:\Program Files\Java\j2re1.4.2_04\bin\javaw.exe"
110 [main] DEBUG root  - Command args[1] =
-Djava.class.path=d:/PrismTech/OpenFusion\classes;d:/PrismTech/OpenFusion\lib\avalon-framew
ork.jar;d:/PrismTech/OpenFusion\lib\castor-0.9.2.jar;d:/PrismTech/OpenFusion\lib\classes12.
jar;d:/PrismTech/OpenFusion\lib\connector.jar;d:/PrismTech/OpenFusion\lib\EccpressoAll.jar;
d:/PrismTech/OpenFusion\lib\excalibur-of.jar;d:/PrismTech/OpenFusion\lib\flexlm.jar;d:/Pris
mTech/OpenFusion\lib\fusion.jar;d:/PrismTech/OpenFusion\lib\hsqldb.jar;d:/PrismTech/OpenFus
ion\lib\idl.jar;d:/PrismTech/OpenFusion\lib\jaas.jar;d:/PrismTech/OpenFusion\lib\jacorb.jar
;d:/PrismTech/OpenFusion\lib\jdom.jar;d:/PrismTech/OpenFusion\lib\jhall.jar;d:/PrismTech/Op
enFusion\lib\jmxri.jar;d:/PrismTech/OpenFusion\lib\jmxtools.jar;d:/PrismTech/OpenFusion\lib
\jndi.jar;d:/PrismTech/OpenFusion\lib\jta-spec1_0_1.jar;d:/PrismTech/OpenFusion\lib\log4j.j
ar;d:/PrismTech/OpenFusion\lib\logkit.jar;d:/PrismTech/OpenFusion\lib\ntp.jar;d:/PrismTech/
OpenFusion\lib\onlinehelp.jar;d:/PrismTech/OpenFusion\lib\ots-jts_1.0.jar;d:/PrismTech/Open
Fusion\lib\tyrex-1.0.1.jar;d:/PrismTech/OpenFusion\lib\xalan.jar;d:/PrismTech/OpenFusion\li
b\xercesImpl.jar;d:/PrismTech/OpenFusion\lib\xmlParserAPIs.jar;d:/PrismTech/OpenFusion\lib\
cosnaming.jar;.;d:\PrismTech\OpenFusion\classes
110 [main] DEBUG root  - Command args[2] = -Djava.endorsed.dirs=d:/PrismTech/OpenFusion\lib
130 [main] DEBUG root  - Command args[3] = -DOF.Install.Dir=d:/PrismTech/OpenFusion
130 [main] DEBUG root  - Command args[4] =
-DOF.Domains.URL=file:d:/PrismTech/OpenFusion/domains
130 [main] DEBUG root  - Command args[5] =
-DOF.Node.URL=file:d:/PrismTech/OpenFusion/domains/OpenFusion/localhost
130 [main] DEBUG root  - Command args[6] =
-DOF.Domain.URL=file:d:/PrismTech/OpenFusion/domains/OpenFusion
130 [main] DEBUG root  - Command args[7] = -DOF.License.File=d:/PrismTech/OpenFusion\etc
130 [main] DEBUG root  - Command args[8] = -DSecurityEnabled=false
130 [main] DEBUG root  - Command args[9] =
-Djava.security.auth.login.config=file:d:/PrismTech/OpenFusion/etc/security/jaas_config
130 [main] DEBUG root  - Command args[10] = com.prismt.openfusion.orb.Service
130 [main] DEBUG root  - Command args[11] =
file:d:/PrismTech/OpenFusion/domains/OpenFusion/localhost/NotificationService/NotificationS
ervice.xml
130 [main] DEBUG root  - Server NotificationService: Starting
```

Example 2 Concatenating Command String

```
javaw.exe
-Djava.class.path=d:/PrismTech/OpenFusion\classes;d:/PrismTech/OpenFusion\lib\avalon-framew
ork.jar;d:/PrismTech/OpenFusion\lib\castor-0.9.2.jar;d:/PrismTech/OpenFusion\lib\classes12.
jar;d:/PrismTech/OpenFusion\lib\connector.jar;d:/PrismTech/OpenFusion\lib\EccpressoAll.jar;
d:/PrismTech/OpenFusion\lib\excalibur-of.jar;d:/PrismTech/OpenFusion\lib\flexlm.jar;d:/Pris
mTech/OpenFusion\lib\fusion.jar;d:/PrismTech/OpenFusion\lib\hsqldb.jar;d:/PrismTech/OpenFus
ion\lib\idl.jar;d:/PrismTech/OpenFusion\lib\jaas.jar;d:/PrismTech/OpenFusion\lib\jacorb.jar
;d:/PrismTech/OpenFusion\lib\jdom.jar;d:/PrismTech/OpenFusion\lib\jhall.jar;d:/PrismTech/Op
enFusion\lib\jmxri.jar;d:/PrismTech/OpenFusion\lib\jmxtools.jar;d:/PrismTech/OpenFusion\lib
\jndi.jar;d:/PrismTech/OpenFusion\lib\jta-spec1_0_1.jar;d:/PrismTech/OpenFusion\lib\log4j.j
ar;d:/PrismTech/OpenFusion\lib\logkit.jar;d:/PrismTech/OpenFusion\lib\ntp.jar;d:/PrismTech/
OpenFusion\lib\onlinehelp.jar;d:/PrismTech/OpenFusion\lib\ots-jts_1.0.jar;d:/PrismTech/Open
Fusion\lib\tyrex-1.0.1.jar;d:/PrismTech/OpenFusion\lib\xalan.jar;d:/PrismTech/OpenFusion\li
b\xercesImpl.jar;d:/PrismTech/OpenFusion\lib\xmlParserAPIs.jar;d:/PrismTech/OpenFusion\lib\
cosnaming.jar;.;d:\PrismTech\OpenFusion\classes
```

```
-Djava.endorsed.dirs=d:/PrismTech/OpenFusion\lib -DOF.Install.Dir=d:/PrismTech/OpenFusion
-DOF.Domains.URL=file:d:/PrismTech/OpenFusion/domains
-DOF.Node.URL=file:d:/PrismTech/OpenFusion/domains/OpenFusion/localhost
-DOF.Domain.URL=file:d:/PrismTech/OpenFusion/domains/OpenFusion
-DOF.License.File=d:/PrismTech/OpenFusion\etc -DSecurityEnabled=false
-Djava.security.auth.login.config=file:d:/PrismTech/OpenFusion/etc/security/jaas_config
com.prismt.openfusion.orb.Service
file:d:/PrismTech/OpenFusion/domains/OpenFusion/localhost/NotificationService/NotificationS
ervice.xml
```

Note that the *d:/PrismTech/OpenFusion* appearing in the last line of the above command string is the directory where OpenFusion was installed on the system which was used to create the examples: this directory path will be substituted with the path used on your system.

Using the jaco Batch File

If *jaco.bat* is normally be used to run the required program, then the command string should be as taken from the jaco.bat file. For example:

```
java -Xbootclasspath/p:"d:/PrismTech/OpenFusion/classes;d:/PrismTech/OpenFusion/lib/
jacorb.jar;d:/PrismTech/OpenFusion/lib/logkit.jar;d:/PrismTech/OpenFusion/lib/avalon-framew
ork.jar;d:/PrismTech/OpenFusion/lib/dnsjava-1.3.2.jar;%CLASSPATH%"
-Dorg.omg.CORBA.ORBClass=org.jacorb.orb.ORB
-Dorg.omg.CORBA.ORBSingletonClass=org.jacorb.orb.ORBSingleton %*
```

Where:

the *d:/PrismTech/OpenFusion* is the location of the jacorb installation used by example: you should use the actual path used on your system.

the *%** is replaced by the name of the program which is to be run, along with any command line parameters it uses, for example if running the jacorb demo examples this would be *demo.hello.Server <ior-file>*
.