MICRO FOCUS

Micro Focus OpenFusion Implementation Repository Version 1.1

User Guide
for JacORB 2.3 or JacORB 3.0

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

http://www.microfocus.com

2016-04-19

Contents

Preface

About the OpenFusion Implementation Repository User Guide

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

The ***OpenFusion Implementation Repository User Guide*** is intended to be used with the standard documentation provided with the OpenFusion JacORB product.

Intended Audience

The ***OpenFusion Implementation Repository User Guide*** is intended to be used by users and developers who wish to use OpenFusion Implementation Repository with 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), and should understand implementation repository concepts and principles.

Organization

The ***OpenFusion Implementation Repository User Guide*** is organized 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
- "Using a Command File" describes how to deal with long command strings by using a command file.

Conventions

The conventions listed below are used to guide and assist the reader in understanding the ***OpenFusion Implementation Repository User Guide***.


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

i
Item contains helpful hint or special information.

WIN
Information applies to Windows systems only.

UNIX
Information applies to Unix based systems (such as Solaris) only.

C
C language specific

C++
C++ language specific

Java
Java language specific

Hypertext links are shown as blue.

Items shown as cross references, such as "Introduction", are as hypertext links: click on the reference to go to the item.

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

`Courier` fonts indicate programming code and file names.

Extended code fragments are shown 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", "");
```

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

Bold indicates Graphical User Interface (GUI) elements and commands, for example, **File | Save** from a menu.

Steps in a task are numbered:

1 One of several steps required to complete a task.

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 *Product Updates* section of the Micro Focus SupportLine Web site, where you can download fixes and documentation updates.
- The *Examples and Utilities* section of the Micro Focus SupportLine Web site, including demos and additional product documentation.

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

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.

Also, visit:

- The Micro Focus Community Web site, where you can browse the Knowledge Base, read articles and blogs, find demonstration programs and examples, and discuss this product with other users and Micro Focus specialists.
- The Micro Focus YouTube channel for videos related to your product.

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. You can find this by either logging into your order via the Electronic Product Distribution email or via the invoice with the order.

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 *Product Updates* section of the Micro Focus SupportLine Web site, 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, then click *Support*.

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:

- 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

Introduction

General Description

An Implementation Repository is key 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 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 may vary between different vendors. Nonetheless, an IMR must guarantee client-side interoperability between different vendors' ORBs: a client using one vendor's ORB can make requests to a server registered with another's 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 the Micro Focus JacORB ORB. 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.

This version of the OpenFusion Implementation Repository is compatible with:

- All OpenFusion CORBA Services versions prior to version 5.x;
- OpenFusion CORBA Services version 5.x when used with JacORB 2.3 or JacORB 3.0.

Features

The OpenFusion Implementation Repository provides the features listed below and which are generally available to OpenFusion JacORB:

- 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
- Persistence of configuration information using file-based persistence plug-ins
- Automatic server activation, including activation of servers on remote hosts (that is, servers located on a different host to the ImR)
- Co-location of the OF ImR and Activator in a single process
- Configurable load balancing
- Fail-over protection for ImR instances and stateless replica servers

Using the OF ImR

Overview

Specific Features

The OpenFusion Implementation Repository provides the specific features listed below:

- 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
- Configurable load balancing
- Fail-over protection for ImR instances through automatic discovery and re-registration with other ImR instances running on the same or different hosts
- Basic fail-over protection for stateless replica servers

Command Line Utilities

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

i

The utilities are use a naming convention whereby each utility name contains the prefix *jac_* for the JacORB ORB.

The utilities include:

- **ImR Launcher** (*jac_imr*) - starts and configures the ImR
- **Implementation Repository Manager** (*jac_imr_mgr*) - provides information about the ImR and is used to register, remove, and deactivate servers.

- **Activator Launcher** (*jac_activator*) - enables servers to be automatically activated for use with JacORB clients.
- **Activator Manager** (*jac_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

In addition, the OF ImR can also be configured on JacORB by setting property values in JacORB's *jacorb.properties* file, as described in "Using a Command File".

Running and Configuration

The OpenFusion Implementation Repository is started with the ImR Launcher command line utility, *jac_imr*. 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.

The jacorb.properties file is located in the *<install>/classes* directory, where *<install>* is the OpenFusion installation directory. Table 6, *JacORB ImR configuration properties* 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"). 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
```

ImR Launcher

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

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

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

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: *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. *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 `jacorb.imr.active_server_timeout`. 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 its servers (no ping mechanism is specified). The default value for the `aliveness_policy` is *3* (`NONE`).

Table 1 ImR Launcher

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
`-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.

Table 1 ImR Launcher

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"), 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.

ImR Manager

The ImR Manager (*`jac_imr_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:

```
%  jac_imrd_mgr [options...]
```

where *`[options...]`* is one or more of the options and sub-options listed in Table 2, *ImR Manager*. 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

Table 2 ImR Manager

Options and Sub-options	Description
-n, --name <name>	Name of the server to be added (used with JacORB 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, that is, 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
-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>	Name of the server to be shutdown (used with JacORB 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	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>	Lists all registered servers identified by *<name>* (used with JacORB 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>	Identifies the name of the server (used with JacORB 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* command:

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

where:

<> indicates mandatory items, *[]* indicates optional items (see Table 2, *ImR Manager* 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 all the following apply:

- 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.
- 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 *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*) 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.

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.

Activator Manager

The Activator Manager (*jac_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.

Unless the *-ORBInitRef* option is used (see Table 4, *Activator Manager* 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, *Activator Manager*.

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>`	Identifies the server name to use.
`-c, --command <command>`	Specifies the command to use.
`-f, --comm_file <filename>`	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 "Using a Command File" for instructions on creating and using a command file.
`-r, --remove`	Removes the server identified by the *--name* sub-option.
`-n, --name <name>`	Identifies the server name to remove.
`-s, --start`	Starts the server identified by the *--name* sub-option.
`-n, --name <name>`	Identifies the server name to start.
`-h, --help`	Displays descriptions of the options listed in this table.

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 *--start* command:

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

ImR Locator

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, *Activator Manager*.

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 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.

1 Start the Activator.

```
%  jac_activator
```

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 platform, then the entire start up command must be entered either:

  - On the command line, but only if the command is less than 2K long, or
  - 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 "Using a Command File" for instructions on creating and using a command file.

1 Start the ImR.

```
%  jac_imr
```

2 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 *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, *JacORB ImR configuration properties*) 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, *JacORB ImR configuration properties*).
- 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, *JacORB ImR configuration properties*) 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 fail-over 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.

JacORB ImR Properties

Table 6, *JacORB ImR configuration properties* 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

Table 6 JacORB ImR configuration properties

Property	Description	Type
`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
`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
`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

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 platforms can not cope with command line strings which are longer than 2k (see "Activator Manager").

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.

1. Install OpenFusion, if it is not already installed.
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.
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. The 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:/MicroFocus/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:/MicroFocus/OpenFusion\
classes;d:/MicroFocus/OpenFusion\lib\avalon-framework.jar;d:/MicroFocus/OpenFusion\
lib\castor-0.9.2.jar;d:/MicroFocus/OpenFusion\lib\classes12.jar;d:/MicroFocus/
OpenFusion\lib\connector.jar;d:/MicroFocus/OpenFusion\lib\EccpressoAll.jar;d:/
MicroFocus/OpenFusion\lib\excalibur-of.jar;d:/MicroFocus/OpenFusion\lib\
flexlm.jar;d:/MicroFocus/OpenFusion\lib\fusion.jar;d:/MicroFocus/OpenFusion\lib\
hsqldb.jar;d:/MicroFocus/OpenFusion\lib\idl.jar;d:/MicroFocus/OpenFusion\lib\
jaas.jar;d:/MicroFocus/OpenFusion\lib\jacorb.jar;d:/MicroFocus/OpenFusion\lib\
jdom.jar;d:/MicroFocus/OpenFusion\lib\jhall.jar;d:/MicroFocus/OpenFusion\lib\
jmxri.jar;d:/MicroFocus/OpenFusion\lib\jmxtools.jar;d:/MicroFocus/OpenFusion\lib\
jndi.jar;d:/MicroFocus/OpenFusion\lib\jta-spec1_0_1.jar;d:/MicroFocus/OpenFusion\
lib\log4j.jar;d:/MicroFocus/OpenFusion\lib\logkit.jar;d:/MicroFocus/OpenFusion\lib\
ntp.jar;d:/MicroFocus/OpenFusion\lib\onlinehelp.jar;d:/MicroFocus/OpenFusion\lib\
ots-jts_1.0.jar;d:/MicroFocus/OpenFusion\lib\tyrex-1.0.1.jar;d:/MicroFocus/
OpenFusion\lib\xalan.jar;d:/MicroFocus/OpenFusion\lib\xercesImpl.jar;d:/MicroFocus/
OpenFusion\lib\xmlParserAPIs.jar;d:/MicroFocus/OpenFusion\lib\cosnaming.jar;.;d:\
MicroFocus\OpenFusion\classes
110 [main] DEBUG root  - Command args[2] = -Djava.endorsed.dirs=d:/MicroFocus/
OpenFusion\lib
130 [main] DEBUG root  - Command args[3] = -DOF.Install.Dir=d:/MicroFocus/OpenFusion
130 [main] DEBUG root  - Command args[4] = -DOF.Domains.URL=file:d:/MicroFocus/
OpenFusion/domains
130 [main] DEBUG root  - Command args[5] = -DOF.Node.URL=file:d:/MicroFocus/
OpenFusion/domains/OpenFusion/localhost
130 [main] DEBUG root  - Command args[6] = -DOF.Domain.URL=file:d:/MicroFocus/
OpenFusion/domains/OpenFusion
130 [main] DEBUG root  - Command args[7] = -DOF.License.File=d:/MicroFocus/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:/
MicroFocus/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:/MicroFocus/OpenFusion/domains/
OpenFusion/localhost/NotificationService/NotificationService.xml
130 [main] DEBUG root  - Server NotificationService: Starting
```

Example 2 Concatenating Command String

```
javaw.exe -Djava.class.path=d:/MicroFocus/OpenFusion\classes;d:/MicroFocus/
OpenFusion\lib\avalon-framework.jar;d:/MicroFocus/OpenFusion\lib\castor-
0.9.2.jar;d:/MicroFocus/OpenFusion\lib\classes12.jar;d:/MicroFocus/OpenFusion\
lib\connector.jar;d:/MicroFocus/OpenFusion\lib\EccpressoAll.jar;d:/MicroFocus/
OpenFusion\lib\excalibur-of.jar;d:/MicroFocus/OpenFusion\lib\flexlm.jar;d:/
MicroFocus/OpenFusion\lib\fusion.jar;d:/MicroFocus/OpenFusion\lib\hsqldb.jar;d:/
MicroFocus/OpenFusion\lib\idl.jar;d:/MicroFocus/OpenFusion\lib\jaas.jar;d:/
MicroFocus/OpenFusion\lib\jacorb.jar;d:/MicroFocus/OpenFusion\lib\jdom.jar;d:/
MicroFocus/OpenFusion\lib\jhall.jar;d:/MicroFocus/OpenFusion\lib\jmxri.jar;d:/
MicroFocus/OpenFusion\lib\jmxtools.jar;d:/MicroFocus/OpenFusion\lib\jndi.jar;d:/
MicroFocus/OpenFusion\lib\jta-spec1_0_1.jar;d:/MicroFocus/OpenFusion\lib\
log4j.jar;d:/MicroFocus/OpenFusion\lib\logkit.jar;d:/MicroFocus/OpenFusion\lib\
ntp.jar;d:/MicroFocus/OpenFusion\lib\onlinehelp.jar;d:/MicroFocus/OpenFusion\lib\
ots-jts_1.0.jar;d:/MicroFocus/OpenFusion\lib\tyrex-1.0.1.jar;d:/MicroFocus/
OpenFusion\lib\xalan.jar;d:/MicroFocus/OpenFusion\lib\xercesImpl.jar;d:/
MicroFocus/OpenFusion\lib\xmlParserAPIs.jar;d:/MicroFocus/OpenFusion\lib\
cosnaming.jar;.;d:\MicroFocus\OpenFusion\classes
-Djava.endorsed.dirs=d:/MicroFocus/OpenFusion\lib -DOF.Install.Dir=d:/MicroFocus/
OpenFusion -DOF.Domains.URL=file:d:/MicroFocus/OpenFusion/domains -
DOF.Node.URL=file:d:/MicroFocus/OpenFusion/domains/OpenFusion/localhost
-DOF.Domain.URL=file:d:/MicroFocus/OpenFusion/domains/OpenFusion -
DOF.License.File=d:/MicroFocus/OpenFusion\etc -DSecurityEnabled=false
```

```
-Djava.security.auth.login.config=file:d:/MicroFocus/OpenFusion/etc/security/
jaas_config com.prismt.openfusion.orb.Service
file:d:/MicroFocus/OpenFusion/domains/OpenFusion/localhost/NotificationService/
NotificationService.xml
```

Note that *d:/MicroFocus/OpenFusion* is the directory where OpenFusion CORBA Services 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:/MicroFocus/OpenFusion/classes;d:/MicroFocus/
OpenFusion/lib/ jacorb.jar;d:/MicroFocus/OpenFusion/lib/logkit.jar;d:/MicroFocus/
OpenFusion/lib/avalon-framework.jar;d:/MicroFocus/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:

```
java -Djava.endorsed.dirs="c:/MicroFocus/OpenFusion/lib/endorsed" -Djacorb.home="c:/
MicroFocus/OpenFusion" -Dorg.omg.CORBA.ORBClass=org.jacorb.orb.ORB -
Dorg.omg.CORBA.ORBSingletonClass=org.jacorb.orb.ORBSingleton -classpath "c:/
MicroFocus/OpenFusion/classes;c:/MicroFocus/OpenFusion/lib/endorsed/jacorb.jar;c:/
MicroFocus/OpenFusion/lib/endorsed/jacorb-omgapi.jar;c:/MicroFocus/OpenFusion/lib/
endorsed/slf4j-api-1.7.6.jar;%CLASSPATH%" %*
```

- *d:/MicroFocus/OpenFusion* is the location of the jacorb installation used by the 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>*.