

ProActive Parallel Suite

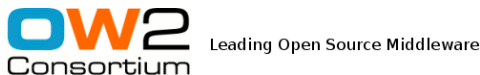


An Open Source Middleware For Parallel, Distributed, Multicore Computing

ProActive Resource Manager

Version 3.1.0

The OASIS Research Team and ActiveEon Company



ProActive Resource Manager v3.1.0 Documentation

Legal Notice

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation; version 3 of the License.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

If needed, contact us to obtain a release under GPL Version 2 or 3 or a different license than the AGPL.

Contact: proactive@ow2.org or contact@activeeon.com

Copyright 1997-2011 INRIA/University of Nice-Sophia Antipolis/ActiveEon.

Mailing List

proactive@ow2.org

Mailing List Archive

<http://www.objectweb.org/wws/arc/proactive>

Bug-Traking System

<http://bugs.activeeon.com/browse/PROACTIVE>

Contributors and Contact Information

Team Leader

Denis Caromel
INRIA 2004, Route des Lucioles, BP 93
06902 Sophia Antipolis Cedex
France
phone: +33 492 387 631
fax: +33 492 387 971
e-mail: Denis.Caromel@inria.fr

Contributors from OASIS Team

Brian Amedro
Francoise Baude
Francesco Bongiovanni
Borelli Elvio
Yu Feng
Ludovic Henrio
Fabrice Huet
Virginie Legrand Contes
Eric Madelaine
Laurent Pellegrino
Guilherme Peretti-Pezzi
Franca Perrina
Marcela Rivera
Christian Ruz
Bastien Sauvan
Mathieu Schnoor
Doglov Sergei
Oleg Smirnov
Marc Valdener
Fabien Viale

Contributors from ActiveEon Company

Vladimir Bodnartchouk
Arnaud Contes
Cédric Dalmasso
Christian Delbé
Jean-Michel Guillaume
Clément Mathieu
Emil Salageanu
Jean-Luc Scheefer

Former Important Contributors

Laurent Baduel (Group Communications)
Vincent Cave (Legacy Wrapping)
Alexandre di Costanzo (P2P, B&B)
Abhijeet Gaikwad (Option Pricing)
Mario Leyton (Skeleton)
Matthieu Morel (Initial Component Work)
Romain Quilici
Germain Sigety (Scheduling)
Julien Vayssiere (MOP, Active Objects)

Table of Contents

List of figures iv

Part I. ProActive Resource Manager

Chapter 1. Overview 3

- 1.1. Overview 3
- 1.2. Basic concepts 3
 - 1.2.1. Server and client parts 3
 - 1.2.2. Computing node 3
 - 1.2.3. Node source 4
- 1.3. Architecture 4

Chapter 2. Installation guide 6

- 2.1. Requirements 6
- 2.2. Downloading and unpacking 6

Chapter 3. User guide 7

- 3.1. Connecting to the resource manager 7
- 3.2. Getting and releasing nodes 8
- 3.3. Nodes topology 9
- 3.4. Limitations 11

Chapter 4. Administration guide 12

- 4.1. Configuration guide 12
 - 4.1.1. Main configuration file 12
 - 4.1.2. Users authentication 13
 - 4.1.3. User authorization 17
- 4.2. The command line interface 19
 - 4.2.1. Launching the resource manager 19
 - 4.2.2. Interacting with the resource manager 20
- 4.3. Organizing your nodes 24
 - 4.3.1. Default infrastructure 26
 - 4.3.2. Local infrastructure 26
 - 4.3.3. GCM customized infrastructure 26
 - 4.3.4. SSH infrastructure 26
 - 4.3.5. Command Line infrastructure 27
 - 4.3.6. Windows HPC infrastructure 27
 - 4.3.7. Amazon EC2 Infrastructure 28
 - 4.3.8. Load Sharing Facility (LSF) infrastructure 29
 - 4.3.9. Portable Batch System (PBS) infrastructure 30
 - 4.3.10. Generic Batch Job infrastructure 31
 - 4.3.11. Virtualized infrastructure 31
 - 4.3.12. Static policy 36
 - 4.3.13. Time slot policy 36
 - 4.3.14. Cron policy 37
 - 4.3.15. "Remove nodes when scheduler is idle" policy 37
 - 4.3.16. "Scheduler loading" policy 38
 - 4.3.17. "Cron load based" policy 38

4.3.18. "Cron slot load based" policy	39
4.3.19. Amazon EC2 policy	40
4.3.20. Custom infrastructure/policy	41
4.4. Administration with Java API	41
4.4.1. Start and stop the resource manager	41
4.4.2. Register an existing node	42
4.4.3. Remove a node from the resource manager	42
4.4.4. Create a node source	43
4.5. Accounting	43
4.6. The JMX interface	44
4.7. Resource Manager Graphical User Interface	47
4.7.1. Launching the Resource Manager GUI	47
4.7.2. Connect to an existing Resource Manager	48
4.7.3. Resource Manager main views	49
4.7.4. Main actions	55
Chapter 5. ProActive Windows Agent	61
5.1. Context	61
5.2. Functionalities	61
5.3. Example	61
5.4. Installation	63
5.5. Configuration	64
5.6. Start the agent	68
5.7. Other information	69

List of Figures

1.1. The ProActive Resource Manager architecture	5
4.1. Credentials encryption	14
4.2. Virtual Infrastructure NodeSource creation dialog	33
4.3. Structure of the Resource Manager JMX interface	44
4.4. Connection using JConsole	46
4.5. Browse MBean attributes	47
4.6. Resource Manager startup screen	48
4.7. Connection screen	48
4.8. Tree view	50
4.9. Tab view	51
4.10. Compact view	52
4.11. Compact view	53
4.12. Resource Explorer view tool bar	54
4.13. Node Source creation dialog box	55
4.14. SSH infrastructure configuration panel	57
4.15. Scheduler loading policy configuration panel	58
4.16. Node Source removal dialog box	59
4.17. Add Node by URL	59
4.18. Node removal dialog	60
4.19. RM shutdown dialog box	60
5.1. Configuration during installation	63
5.2. ProActive Agent Control window	65
5.3. Configuration Editor window - General Tab	66
5.4. Configuration Editor window - Connection Tab (Resource Manager Registration)	67
5.5. Configuration Editor window - Planning Tab	68

Part I. ProActive Resource Manager

Table of Contents

Chapter 1. Overview	3
1.1. Overview	3
1.2. Basic concepts	3
1.2.1. Server and client parts	3
1.2.2. Computing node	3
1.2.3. Node source	4
1.3. Architecture	4
Chapter 2. Installation guide	6
2.1. Requirements	6
2.2. Downloading and unpacking	6
Chapter 3. User guide	7
3.1. Connecting to the resource manager	7
3.2. Getting and releasing nodes	8
3.3. Nodes topology	9
3.4. Limitations	11
Chapter 4. Administration guide	12
4.1. Configuration guide	12
4.1.1. Main configuration file	12
4.1.2. Users authentication	13
4.1.3. User authorization	17
4.2. The command line interface	19
4.2.1. Launching the resource manager	19
4.2.2. Interacting with the resource manager	20
4.3. Organizing your nodes	24
4.3.1. Default infrastructure	26
4.3.2. Local infrastructure	26
4.3.3. GCM customized infrastructure	26
4.3.4. SSH infrastructure	26
4.3.5. Command Line infrastructure	27
4.3.6. Windows HPC infrastructure	27
4.3.7. Amazon EC2 Infrastructure	28
4.3.8. Load Sharing Facility (LSF) infrastructure	29
4.3.9. Portable Batch System (PBS) infrastructure	30
4.3.10. Generic Batch Job infrastructure	31
4.3.11. Virtualized infrastructure	31
4.3.12. Static policy	36
4.3.13. Time slot policy	36
4.3.14. Cron policy	37
4.3.15. "Remove nodes when scheduler is idle" policy	37
4.3.16. "Scheduler loading" policy	38
4.3.17. "Cron load based" policy	38
4.3.18. "Cron slot load based" policy	39
4.3.19. Amazon EC2 policy	40

4.3.20. Custom infrastructure/policy	41
4.4. Administration with Java API	41
4.4.1. Start and stop the resource manager	41
4.4.2. Register an existing node	42
4.4.3. Remove a node from the resource manager	42
4.4.4. Create a node source	43
4.5. Accounting	43
4.6. The JMX interface	44
4.7. Resource Manager Graphical User Interface	47
4.7.1. Launching the Resource Manager GUI	47
4.7.2. Connect to an existing Resource Manager	48
4.7.3. Resource Manager main views	49
4.7.4. Main actions	55
Chapter 5. ProActive Windows Agent	61
5.1. Context	61
5.2. Functionalities	61
5.3. Example	61
5.4. Installation	63
5.5. Configuration	64
5.6. Start the agent	68
5.7. Other information	69

Chapter 1. Overview

1.1. Overview

The ProActive Resource Manager is the software for coupling distributed resources in order to solve large-scale problems. The distributed resources are represented by Java virtual machines (JVM) which can be launched on desktop computers, clusters or clouds. The resource manager provides a single point of access to all resources enabling an effective way of selecting them for computations with the criteria you need.

More precisely, the resource manager does the following:

- Deploys nodes (launch JVMs) automatically to different types of infrastructure (i.e. grid, cloud, another resource manager, etc).
- Maintains and monitors the list of resources and manages their states (i.e. deploying, lost, configuring, free, busy, down).
- Supplies computing nodes to users based on user criteria (i.e. specific operating system, available resources or licenses and many others).



Note

In the following sections, the term "ProActive Resource Manager" will be used to describe the whole application (including the client interface for using the provided nodes) whereas the term "resource manager" will be used only to describe the server part (for managing resources) of the ProActive Resource Manager.

1.2. Basic concepts

1.2.1. Server and client parts

ProActive Resource Manager is a client/server application where each part can be described as follows:

- **Server** - The server part of ProActive Resource Manager, called just "resource manager" from now on, selects nodes according to user requests, restricts an access to these nodes to other users, monitors nodes states and provides up-to-date information about resources utilization. Once nodes are selected and given to a user, it has a direct access to them. When user finishes its computation it releases nodes by contacting the resource manager.
- **Client** - The person or entity requesting nodes for computation from the resource manager. The main client example is ProActive Scheduler which requests nodes when it has jobs to execute. However, it could be anyone who wants to directly run a distributed application on already existing nodes having an exclusive access to them. Clients also may trigger automatic nodes deployment, manually add and remove nodes to the resource manager.

1.2.2. Computing node

- **Node** - The resource manager is written in JAVA using ProActive Parallel Suite and is able to communicate with remote JVMs that are also running ProActive (the same version of ProActive that is used in the resource manager). A node inside such a JVM is just a logical container for activities and it can be seen as an entry point to the JVM. Each ProActive JVM may have one or more nodes to perform computations and it is just a matter of how the JVM is exposed to the external world (in term of number of entry points). The number of running JVMs per host and nodes per JVM are up to the resource providers but here there is always a tradeoff between performance and reliability. The maximum performance can be achieved when only one JVM is running on host and all users run their computation on this JVM simultaneously having an access to the same address space in memory. If it is not acceptable, only one node per JVM can be exposed. In this case, two users cannot run their computation in parallel on a single JVM.
- **Nodes deployment** - The process of launching JVMs with ProActive nodes for further utilization for computations. At this moment, users can configure proper number of JVMs per host and nodes per JVM and many other parameters like infrastructure type.
- **Adding nodes** - Once the JVMs launched, all nodes have to be registered in the resource manager in order to be used. To do that, they send a request to add them providing their urls.

- **Acquiring nodes** - When the resource manager receives the request of adding nodes, it contacts them and obtains the basic information about each node.
- **Getting nodes** - When a client of the resource manager needs to run computations, it asks for nodes providing nodes criteria. Nodes which are found, are marked as busy which means that other clients cannot use them.
- **Releasing nodes** - Once the computations completed, client releases nodes and they become available for others.
- **Removing nodes** - At any time, nodes could be removed from the resource manager. At this moment, JVM will be killed unless there is some activity still running there. Administrator can also remove nodes preemptively without waiting for computations to be completed.
- **Nodes states** - In order to provide an access to shared resources, the resource manager maintains different node states:
 - **Deploying** - The deployment of the node has been triggered by the resource manager but it has not yet been added.
 - **Lost** - The deployment of the node has failed for some reason. The node has never been added to the resource manager and won't be useable.
 - **Configuring** - Node has been added to the resource manager and is under configuration. The resource manager computes several information about the node. This step can be time consuming depending.
 - **Free** - Node is available for computations.
 - **Busy** - Node has been given to user to execute computations.
 - **To be removed** - Node is busy but requested to be removed. So it will be removed once the client will release it.
 - **Down** - Node is unreachable or down and cannot be used anymore.

1.2.3. Node source

- **Node Source** - A node source is defined by an infrastructure and a policy. All the nodes belonging to the same node source will be launched on the same infrastructure, with the same manner (protocol, job submission, ...) and at the time defined by the policy (for details see [Section 4.3, "Organizing your nodes"](#)).
- **Infrastructure manager** - The part of node source responsible for node deployment to the particular infrastructure. For instance, it may launch JVM over ssh or by submitting a specific job to the native scheduler of the system.
- **Policy** - The part of node source defining rules and limitations of node source utilization. All policies require to define administrator of the node source and a set of its users, so that you can limit nodes utilization. Beside, the policy defines rules of nodes deployment, like static deployment (all nodes are launched at the moment of node source creation and never removed) or time slot deployment (nodes are deployed for particular time) or others.

1.3. Architecture

Basically, the ProActive Resource Manager is a client/server application. Clients are remote and may administrate or monitor the resource manager state by means of command line and graphical user interface. They also may request nodes for computations using Java API. The communication between clients, server and nodes are performed by ProActive. Administrative actions, like node deployment, are delegated by the resource manager core to node sources and infrastructure managers. User requests of getting nodes are processed by selection manager which may contact nodes at the request time and execute some code there in order to know whether the node is suitable. Once user has obtained nodes, he contacts them directly without involving the resource manager. Yet, it is required that a user who runs computations on nodes remains connected to the resource manager, otherwise these nodes will be released.

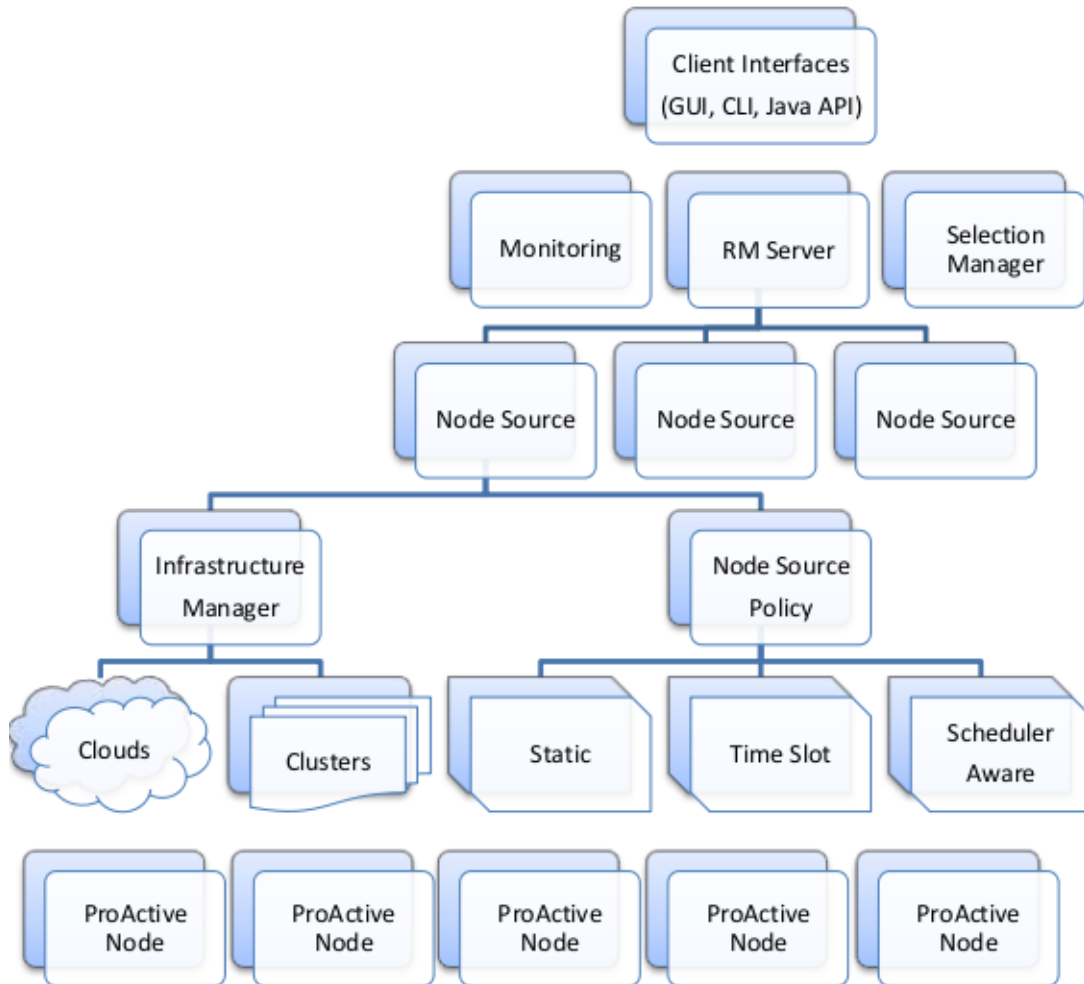


Figure 1.1. The ProActive Resource Manager architecture

Chapter 2. Installation guide

2.1. Requirements

The ProActive Parallel Suite consists of the following products

- **ProActive Programming** enabled the creation of parallel and distributed applications. If you decided to implement your parallel computations in the application running simultaneously on nodes it has all the means to do it. The nodes in this case cannot be shared among different users.
- **ProActive Resourcing** helped to organize your resources. This product includes ProActive Programming and expands its capabilities to deploy nodes on different infrastructures and share them among users.
- **ProActive Scheduling** allowed to schedule your task flow. The scheduler includes the ProActive Resourcing and, as a consequence, ProActive Programming and allows to represent your parallel computations in sequence of jobs and tasks with dependencies.

Before downloading one of this product, just think of how your computations will be represented and where it will be executed.

ProActive is written in Java and requires Java 1.6 (or higher). It is strongly recommended to use [Java from Oracle](#)¹ as all the testing is done with it. Once Java intalled on your system, it is required to define **JAVA_HOME** environment variable.



Warning

If you need a Java 1.5 version, please contact ActiveEon at contact@activeeon.com.

Depending on your network topology, firewall configuration and your needs, it may be required to open some ports for the resource manager. For instance, when you are deploying nodes to the cloud, the resource manager has to be visible from outside.

2.2. Downloading and unpacking

Actually, ProActive does not require installation. Just [download](#)² one of the products according to your needs and **unzip** the archive. That's it. The distribution contains sources and binaries with all dependencies. To add your cluster to the resource manager just **copy** this folder to each host. By doing this you will be able to start ProActive nodes from the resource manager just pointing to the ProActive home.

The resource manager and scheduler graphical user interfaces are written as Eclipse RCP plugins and can be embedded into Eclipse or launched as standalone applications. Download appropriate version for your platform, unzip and run it. It already has ProActive inside, so nothing has to be installed on your system (except Java of course).

¹ <http://java.sun.com/javase/downloads/index.jsp>

² <http://www.activeeon.com/community-downloads>

Chapter 3. User guide

This chapter describes how to connect to the resource manager and request nodes to run parallel computations. If you use ProActive Scheduler you do not have to worry about it as scheduler does it for you. If you would like to run the computations directly on nodes, you have to do this to connect to the existing resource manager.

All the code snippets described in this section can be found in the *org.ow2.proactive.resourcemanager.examples.documentation* package in the Resource Manager sources.

3.1. Connecting to the resource manager

Classes which are required to connect to the resource manager are located in `$RM_HOME/dist/lib/ProActive_ResourceManager-client.jar`. The ProActive library and all its dependencies are needed as well in the class path in order to communicate with the resource manager and nodes.

```

RMAuthentication auth = null;
try {
    // rmAddress is a String of the form: "rmi://hostname:port"
    // e.g. rmi://localhost:1099/
    auth = RMConnection.join(rmAddress);
} catch (RMException e) {
    // The connection to the resource manager cannot be established.
    e.printStackTrace();
}
ResourceManager resourceManager = null;
try {
    // (1) preferred authentication method (getting credentials from the disk)
    resourceManager = auth.login(Credentials.getCredentials());
} catch (KeyException e) {
    try {
        // (2) valid authentication method (creating credentials on the fly)
        PublicKey pubKey = auth.getPublicKey();
        if (pubKey == null) {
            pubKey = Credentials.getPublicKey(Credentials.getPubKeyPath());
        }

        if (pubKey != null) {
            // user and password have to be valid user and password names for the
            // resource manager
            // e.g. by default, "user" and "pwd" are valid
            // refer to ${rm.home}/config/authentication/login.cfg to see all valid
            // pairs and to add some others.
            Credentials cred = Credentials.createCredentials(new CredData(CredData.parseLogin(user),
                CredData.parseDomain(user), password), pubKey);
            resourceManager = auth.login(cred);
        }
    } catch (KeyException ex) {
        // cannot retrieve the public key
        ex.printStackTrace();
    } catch (LoginException ex) {
        // incorrect user name or password
        ex.printStackTrace();
    }
}

```

```

} catch (LoginException e) {
    // incorrect user name or password
    e.printStackTrace();
}

```



Warning

This piece of code will not work unless you have started a resource manager with at least one computing node beforehand. If not, go to the bin/[unix|windows]/ directory and launch the `rm-start[.bat]` script with the `-ln` argument.

```
$ rm-start -ln
```

To learn more about the command line interface, please refer to [Section 4.2, “The command line interface”](#).

Connecting to the resource manager implicates using a keypair infrastructure to establish a secure channel on which the credentials (username and password) will be sent securely. There are three ways to connect to the Resource Manager, as described in the example above:

1. Retrieve the Credentials from disk: supposes the script `create-cred[.bat]` was previously used to generate those credentials. It requires knowing the location of the public key corresponding to the private key that will be used for decryption on server side. To obtain the key, you need to contact the administrator of the Scheduler.

```

# use local public key to generate credentials for user 'demo'
/bin/unix $ ./create-cred -F $HOME/.proactive/rm_pubkey -o $HOME/.proactive/my_encrypted_credentials -l demo
# use rmi://example.com/RMAUTHENTICATION's public key to generate credentials for user 'admin' with pass
'admin' in non-interactive mode
/bin/unix $ ./create-cred -R rmi://example.com/RMAUTHENTICATION -o $HOME/.proactive/
my_encrypted_credentials -l admin -p admin

```

Credentials can now be retrieved when properly designated by the `pa.common.auth.credentials` property; i.e. using `java -Dpa.common.auth.credentials=$HOME/.proactive/my_encrypted_credentials`. Please type `create-cred -h` to have a complete list of possible options.

2. Create the encrypted Credentials on client side: as safe as the previous method, but requires user input, which prevents automation, or storing clear credentials on the disk, which can result to security breaches. This method also requires knowing the public key, which should be offered by the Resource Manager through the Authentication object with the `getPubKey()` method. If the Resource Manager does not know the public key, it can be stored locally on client side and designated by the `pa.common.auth.pubkey` Java property; i.e. using `java -Dpa.common.auth.pubkey=$HOME/.proactive/pub.key`.
3. If you cannot retrieve the public key, you can use the old deprecated API, which sends clear credentials on the network, and requires user input.

With the `ResourceManager` interface, you are now able to obtain the `RMMonitoring` interface which can be used to track resources and their states. Once you have a reference to the `ResourceManager` active object, you can perform all actions according to your role (permissions) such as shutdown of the Resource Manager if you are an administrator.

3.2. Getting and releasing nodes

Now that we have a reference to the resource manager, we can get computing nodes:

```

// For running your computations, you have to get one or more nodes from the
// resource manager. The resource manager will return you as many nodes verifying
// the selection script as it can within the limits of the given number of nodes.
NodeSet nodeSet = resourceManager.getAtMostNodes(nbOfNodes, selectionScript);

```

In the example above, we have requested `nbOfNodes` nodes providing `selectionScript` as criteria. The resource manager does not guaranty that it will find `nbOfNodes` nodes, but it will provide as many nodes as it can. `selectionScript` be null and in this case, no

particular selection will be performed (all nodes will be accepted). Let us now request nodes with some criteria. Criteria are described in selection scripts which are executed on nodes before they are selected. All the information which is possible to get from JVMs could be used for selection decisions. Some examples are available in the resource manager distribution in the *samples/scripts/selection* folder. Here is an example of script which selects only linux machines:

```
String script = "" + /* Check if OS name is Linux and their is wireless connection available */
+ "if (java.lang.System.getProperty('os.name').toLowerCase().equals('linux')){"
+ "    selected = true;" + "} else {" + "    selected = false;" + "}";
```

```
SelectionScript selectionScript = null;
try {
    selectionScript = new SelectionScript(script, "JavaScript");
} catch (InvalidScriptException e) {
    e.printStackTrace();
}
```

In the previous example, we have written the script directly in a String object but it is fortunatly possible to write it in a file and use this file for creating a SelectionScript object.

```
SelectionScript selectionScript = null;
try {
    selectionScript = new SelectionScript(new File(PAResourceManagerProperties.RM_HOME +
"samples/scripts/selection/checkWindows.js"), new String[] {}, false);
} catch (InvalidScriptException e) {
    e.printStackTrace();
}
```

After computations are completed, user can release nodes to make them available for other users.

```
// Once your computations done, you can release nodes.
BooleanWrapper released = resourceManager.releaseNodes(nodeSet);
```

Finally, a user can disconnect from the resource manager as follows:

```
// Once the resource manager is not needed anymore, you can disconnect.
BooleanWrapper disconnected = resourceManager.disconnect();
```



Static and dynamic scripts

There are 2 types of scripts: static and dynamic. Static scripts are executed on node only once and the resource manager remembers the result of execution. Next time it will see the same criteria in a user request, it will use this knowledge to select nodes without contacting them. One should prefer using static scripts for performance reasons. On the contrary, dynamic scripts are executed each time a request came. Dynamic scripts could be those which detect free memory, current system loading etc.

This was just a short overview of Resource Manager user API. For more details, see [the Resource Manager API](#)¹.

3.3. Nodes topology

If the resource manager is configured to discover nodes topology (option **pa.rm.topology.enabled=true** is specified in the config file) the user may request nodes not only with selection scripts but also with topology parameters. The following topology parameters are available:

- **ArbitraryTopologyDescriptor** - no constraint on node location. This descriptor is used by default and provides the quickest way of finding nodes.

¹ http://proactive.inria.fr/release-doc/ResourceManager/api_published/

- **BestProximityDescriptor** - the set of closest nodes. Could be useful for parallel communicative applications which suppose to be executed on several nodes simultaneously. The resource manager has a matrix of distances (latencies) between all nodes it handles. When a new selection request comes the RM starts to group nodes into clusters using [agglomerative hierarchical clustering](#)². The process of clustering can be driven by the function used in distances recalculation between clusters while merging. For now it is possible to use only BestProximityDescriptor.MAX function that basically means that the similarity of two clusters is the similarity of their most dissimilar members. It is also called complete linkage clustering in which the merge criterion is non-local; the entire structure of the clustering can influence merge decisions.
- **ThresholdProximityDescriptor** - the set of nodes within a threshold proximity. This descriptor guarantees that the resource manager returns a set of nodes which are interconnected within specified proximity. Note that previous descriptor does not guarantee this.
- **SingleHostDescriptor** - the set of nodes on a single host.
- **SingleHostExclusiveDescriptor** - the set of nodes of a single host exclusively. The host with selected nodes will be reserved for the user.

By specifying this descriptor in **ResourceManager.getAtMostNodes** user may get more nodes than he asked when host capacity exceeds the requested number of nodes. For instance if a user requests 1 node but there is only a free host with 5 nodes on it, all these 5 nodes will be provided to the user. To be more precise in the target node set 1 node will be presented in the main list and 4 nodes in the extra list (see **NodeSet.getExtraNodes()**). The resource manager tries to find the optimal host minimizing the waist of resources, namely if user requested k nodes

- The machine with exact capacity will be selected if exists.
 - The machine with bigger capacity will be selected if exists. The capacity of the selected machine will be the closest to k.
 - The machine with smaller capacity than k will be selected. In this case the capacity of selected host will be the biggest among all other.
- **MultipleHostsExclusiveDescriptor** - the set of nodes filled in multiple hosts. Hosts with selected nodes will be reserved for the user.

By specifying this descriptor in **ResourceManager.getAtMostNodes** user may get more nodes than he asked if the total capacity of found hosts exceeds the requested number of nodes. For example a user requests 6 nodes from the RM that handles 3 hosts with 5 nodes on each he gets the reference to 6 nodes in the node set plus 4 extra nodes (see **NodeSet.getExtraNodes()**). The resource manager tries to find the optimal host set minimizing the waist of resources, namely if user requested k nodes

- If one machine exists with the capacity k it will be selected.
 - If not the resource manager finds the machine with closest capacity to k and repeats the procedure for (k - "found host capacity") nodes number.
 - If it not possible to find exact number of nodes but it's possible to find more than they will be selected. The number of wasted resources and number of machines will be minimized.
 - Otherwise less nodes will be provided but as the closest as possible to k.
- **DifferentHostsExclusiveDescriptor** - the set of nodes each from a different host. All hosts with selected nodes will be reserved for the user. For example when user asks for 5 nodes with such descriptor he will get 5 nodes from 5 different hosts and all other nodes will be marked as busy and put into the extra list of the target node set.

The resource manager tries to find the optimal host minimizing the waist of resources, namely if user requested k nodes

- It will try to select k hosts with 1 node per host
- If there are not enough of such hosts it will look for machines with 2 nodes and so on.

Lets take a look how we apply it in practice

```
try {
    // For running your you parallel application, you need a set of closest nodes from the
    // resource manager.
    NodeSet closestNodes = resourceManager.getAtMostNodes(nbOfNodes,
```

² http://en.wikipedia.org/wiki/Cluster_analysis#Agglomerative_hierarchical_clustering

```

TopologyDescriptor.BEST_PROXIMITY, selectionScripts, exclusion);

// The previous request does not guarantee than nodes are interconnected with some reasonable threshold.
// If you want this kind of guarantee use the following request.
NodeSet thresholdNodes = resourceManager.getAtMostNodes(nbOfNodes,
    new ThresholdProximityDescriptor(threshold), selectionScripts, exclusion);

// In order to get nodes on the single host, just run
NodeSet singleHostNodes = resourceManager.getAtMostNodes(nbOfNodes,
    TopologyDescriptor.SINGLE_HOST, selectionScripts, exclusion);

// In order to get nodes on the single host exclusively, meaning that no other
// clients use this host, run
NodeSet singleHostExclusiveNodes = resourceManager.getAtMostNodes(nbOfNodes,
    TopologyDescriptor.SINGLE_HOST, selectionScripts, exclusion);
// here we can get more nodes
System.out.println("The number of nodes " + singleHostExclusiveNodes.size());
if (singleHostExclusiveNodes.getExtraNodes() != null) {
    // we have got the host with bigger capacity
    System.out.println("The host capacity is " + singleHostExclusiveNodes.size() +
        singleHostExclusiveNodes.getExtraNodes().size());
} else if (singleHostExclusiveNodes.size() < nbOfNodes) {
    // we have got the host with smaller capacity
    System.out.println("The host capacity is " + singleHostExclusiveNodes.size());
}

// In order to get nodes on multiple hosts exclusively user may also get
// more nodes (reflected hosts capacity) than requested.
NodeSet multipleHostsExclusiveNodes = resourceManager.getAtMostNodes(nbOfNodes,
    TopologyDescriptor.MULTIPLE_HOSTS_EXCLUSIVE, selectionScripts, exclusion);
} catch (TopologyException ex) {
    // topology is not enabled in the resource manager
    ex.printStackTrace();
}

```

3.4. Limitations

When the resource manager is used in standalone mode (without ProActive Scheduler) there are two limitations:

- **Remote class loading** will work only with rmi protocol. So if you would like to run your application on remote nodes using another communication protocol, you have to take care about availability of your classes on each node and probably restart the node adding your classes to the class path.
- **Application logs** are forwarded to the resource manager server log file, not to the client side. In order to see them, you have to have an access to the resource manager logs directory.

Chapter 4. Administration guide

4.1. Configuration guide

4.1.1. Main configuration file

The resource manager has a set of properties that can be changed regarding to your computing infrastructure and your needs. These properties are set in **settings.ini** in the **config/rm** directory:

You can define these variables either in **settings.ini** in the **config/rm** directory or in the the java command line using the -D option (ex: `java -Dpa.rm.node.source.ping.frequency=45000 ...`). Such a property can be added in the provided scripts (`rm-start[.bat]` for example).

- **pa.rm.node.name**: defines a name of the ProActive node that contains the resource manager active objects. Default is *RM_NODE*.
- **pa.rm.node.source.ping.frequency**: the resource manager checks whether computing nodes are still alive. This property sets this verification frequency in milliseconds. Default is *45000*.
- **pa.rm.client.ping.frequency**: the resource manager checks whether clients running computations on nodes are connected. When they disconnect, nodes taken by this client are released. This property sets this verification frequency in milliseconds. Default is *45000*.
- **pa.rm.aliveevent.frequency**: the period of sending "alive" event to resource manager's listeners (in ms). Default is *300000*.
- **pa.rm.select.script.timeout**: max execution time of a selection script, in milliseconds. If execution time is longer than this value, the resource manager assumes that computing node is not suitable for this selection script. Default is *45000*.
- **pa.rm.nodelookup.timeout**: when the resource manager lookups a node there could be a delay due to the network latency (or sometimes firewall configuration). Default is *10000*.
- **pa.rm.gcm.template.application.file**: path of default GCM Application descriptor, used for nodes deployments in GCM based node sources. If this file path is a relative, path is evaluated from the resource manager directory (variable *pa.rm.home*), otherwise path is directly interpreted. See [Section 4.3.3, "GCM customized infrastructure"](#) part. Default is *config/rm/deployment/GCMNodeSourceApplication.xml*.
- **pa.rm.gcmd.path.property.name**: string in the default GCM application that will be replaced by a path to a GCM deployment descriptor. See [Section 4.3.3, "GCM customized infrastructure"](#) part. Default is *gcmd.file*.
- **pa.rm.home**: The resource Manager home directory. Default is *.*, but this property is overridden in Resource Manager launching scripts, `rm-start[.bat]`, in order to build an absolute path of Resource Manager installation directory.
- **pa.rm.nodesource.infrastructures**: path to a file containing the list of supported infrastructures in the resource manager. Default is *config/rm/nodesource/infrastructures*.
- **pa.rm.nodesource.policies**: path to a file containing the list of supported node acquisition policies in the resource manager. Default is *config/rm/nodesource/policies*.
- **pa.rm.nodesource.maxthreadnumber**: max number of threads in node source for parallel execution of network activities Default is *10*.
- **pa.rm.selection.maxthreadnumber**: max number of threads in selection manager for parallel script execution of nodes Default is *10*.
- **pa.rm.monitoring.maxthreadnumber**: max number of threads in monitoring system to notify clients waiting for events. Default is *5*.
- **pa.rm.jmx.connectorname**: name of the JMX Connector for the RM (default is 'JMXRMAgent')
- **pa.rm.jmx.port**: Port number used by JMX. the port used for JMX service and the RMI protocol. It will create a RMI registry if needed.
- **pa.rm.account.refreshrate**: The statistics cache layer refresh rate. It is requested periodically by clients and in order to avoid data base contacting all the time, it is stored in dedicated structures. Default is *10 secs*.
- **pa.rm.ec2.properties**: path to the Amazon EC2 account credentials properties file, mandatory when using the EC2 Infrastructure. Default is *config/rm/deployment/ec2.properties*.

- **pa.rm.auth.jaas.path**: path to the Jaas configuration file which defines what modules are available for internal authentication. Default is *config/authentication/jaas.config*.
- **pa.rm.auth.privkey.path**: path to the Jaas configuration file which defines what modules are available for internal authentication. Default is *config/authentication/keys/priv.key*.
- **pa.rm.auth.pubkey.path**: path to the public key file which is used to encrypt credentials for authentication. Default is *config/authentication/keys/pub.key*.
- **pa.rm.ldap.config.path**: LDAP Authentication configuration file path, used to set LDAP configuration properties. Default is *config/authentication/ldap.cfg*.
- **pa.rm.defaultloginfilename**: login file name for file authentication method. Default is *config/authentication/login.cfg*.
- **pa.rm.defaultgroupfilename**: group file name for file authentication method. Default is *config/authentication/group.cfg*.
- **pa.rm.authentication.loginMethod**: property that defines the method that has to be used for logging users to the resource manager. Default is *RMFileLoginMethod*.
- **pa.rm.db.hibernate.configuration**: hibernate configuration file. Default is *config/rm/database/hibernate/hibernate.cfg.xml*.
- **pa.rm.db.hibernate.dropdb**: drop database before creating a new one. Default is *false*.

4.1.2. Users authentication

As presented before, clients of the resource manager are authenticated at connection time by providing their credentials encapsulating encrypted login and a password.

4.1.2.1. KeyPair authentication

Regardless of which method is actually used to perform the authentication, credentials need to be passed from the client to the Resource Manager, through the network. The data will be encrypted with an AES symmetric secret key to allow unlimited credentials size, and the AES key itself will be encrypted with an RSA keypair.

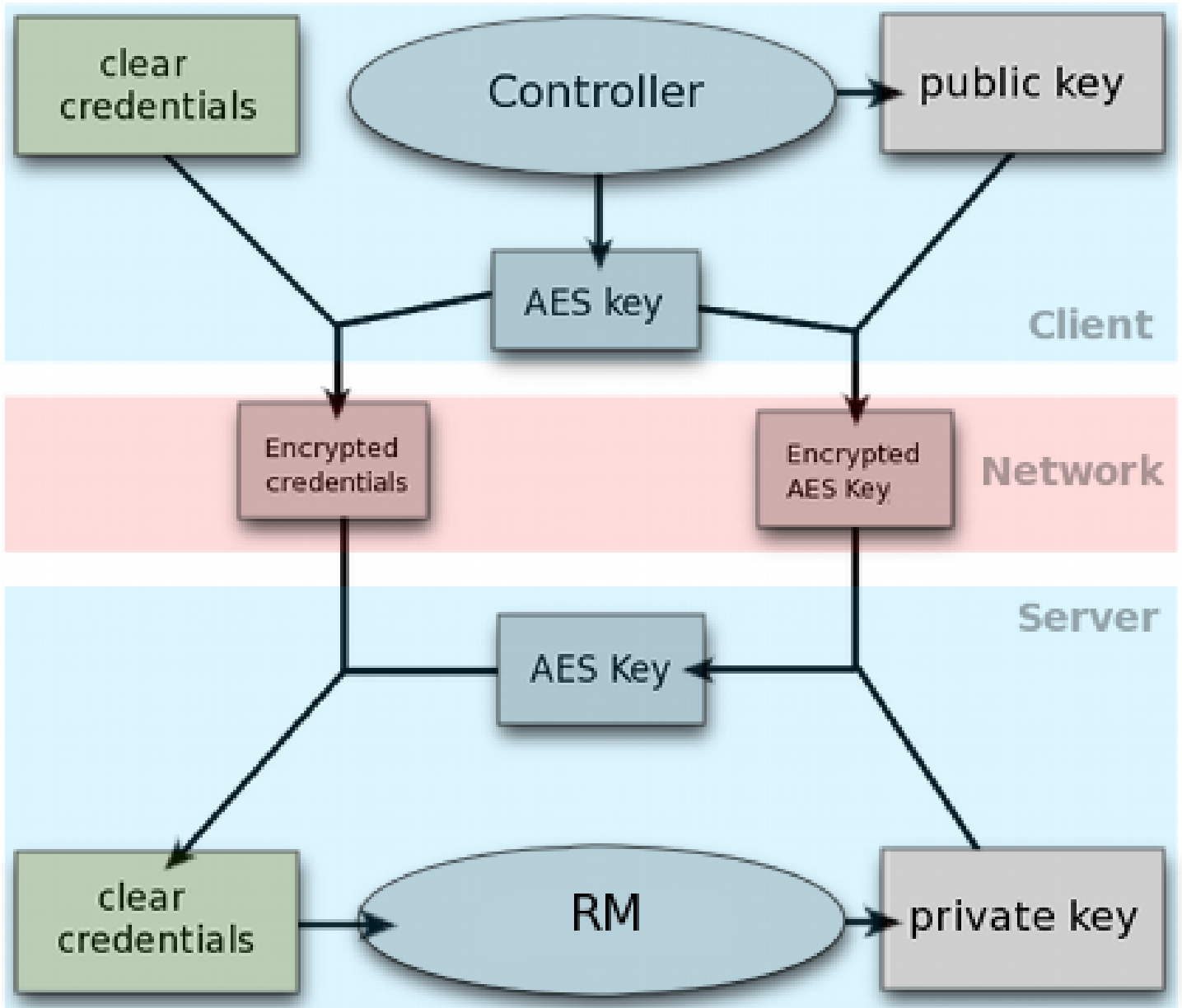


Figure 4.1. Credentials encryption

The Keypair can be generated with the **key-gen[.bat]** script:

```
bin/unix $ ./key-gen -p $HOME/.proactive/priv.key -P $HOME/.proactive/pub.key
```

Accordingly, the Resource Manager configuration must be set so that, when started:

- **pa.rm.auth.privkey.path**=\$HOME/.proactive/priv.key
- **pa.rm.auth.pubkey.path**=\$HOME/.proactive/pub.key

Although no encryption should be performed on server side, the public key should be known from the Resource Manager: indeed, a client can request the public key to the Resource Manager so that it may encrypt its credentials to perform authentication. This method does

not require the Resource Manager's administrator to manually propagate public keys to all its users. Users can encrypt their credentials with the **create-cred[.bat]** script. See [Section 4.4.1, “Start and stop the resource manager”](#) for client-side configuration.

4.1.2.2. Select authentication method

The resource manager manages users authentication and authorization, it has to store users account/password, and check login and password at connection. This storage of users accounts can be managed in two ways: by files, or by LDAP. A Resource Manager property (in `config/rm/settings.ini`) specifies which kind of authentication will be used:

```
#Property that defines the method that has to be used for logging users to the Resource Manager
#It can be one of the following values:
# - "RMFileLoginMethod" to use file login and group management
# - "RMLDAPLoginMethod" to use LDAP login management
pa.rm.authentication.loginMethod=RMFileLoginMethod
```

By default, authentication method is by file (`RMFileLoginMethod`). If you want to use the LDAP-based authentication, replace the `"RMFileLoginMethod"` value by `"RMLDAPLoginMethod"`.

4.1.2.3. Configure file-based authentication

By default, the resource manager stores users accounts, passwords, and group memberships (user or admin), in two files:

- **config/authentication/login.cfg** stores users and passwords accounts. Each line has to look like **user:passwd**. The default `login.cfg` file is given hereafter:

```
admin:admin
user:pwd
demo:demo
```

- **config/authentication/group.cfg** stores users membership. For each user registered in `login.cfg`, a group membership has to be defined in this file. Each line has to look like **user:group**. Group has to be **user** to have user rights, or **admin** to have administrator rights. Default `group.cfg` is like this:

```
admin:admin
demo:admin
user:user
```

You can change the default paths of these two files. Edit file **config/rm/settings.ini** and change the two properties:

- **pa.rm.defaultloginfilename** - To define a user/password file, change this line as follows: **pa.rm.defaultloginfilename=/etc/rm/mylogins.cfg**
- **pa.rm.defaultgroupfilename** - To define a group membership file, change the line as follows: **pa.rm.defaultgroupfilename=/etc/rm/mygroups.cfg**

4.1.2.4. Configure LDAP-based authentication

The resource manager is able to connect to an existing LDAP, to check users login/password and verify users group membership. This authentication method can be used with existing LDAP server which is already configured. In order to use it, few parameters have to be configured, such as path in LDAP tree users, LDAP groups that define user and admin group membership, URL of the LDAP server, LDAP binding method used by connection and configuration of SSL/TLS if you want a secured connection between the resource manager and LDAP.

We assume that LDAP server is configured in the way that:

- all existing users and groups are located under single domain

- users have object class specified in parameter "pa.ldap.user.objectclass"
- groups have object class specified in parameter "pa.ldap.group.objectclass"
- user and group name is defined in cn (Common Name) attribute

```
# EXAMPLE of user entry
#
# dn: cn=jdoe,dc=example,dc=com
# cn: jdoe
# firstName: John
# lastName: Doe
# objectClass: inetOrgPerson

# EXAMPLE of group entry
#
# dn: cn=mygroup,dc=example,dc=com
# cn: mygroup
# firstName: John
# lastName: Doe
# uniqueMember: cn=djoe,dc=example,dc=com
# objectClass: groupOfUniqueNames
```

settings.ini in **config/rm** directory, defines a path to a configuration file that contains all LDAP connection and authentication properties. Default value for this property defines a default configuration file: **config/authentication/ldap.cfg**. Specify your LDAP properties in this file. Properties are explained below.

4.1.2.4.1. Set LDAP url

First, you have to define the LDAP's URL of your organisation. This address corresponds to the property: **pa.ldap.url**. You have to put a standard LDAP-like URL, for example **ldap://myLdap**. You can also set an URL with secure access: **ldaps://myLdap:636**. See [Section 4.1.2.4.4, "Set SSL/TLS parameters"](#) for SSL/TLS configuration.

4.1.2.4.2. Define object class of user and group entities

Then you need to define how to differ user and group entities in LDAP tree. The users object class is defined by property **pa.ldap.user.objectclass** and by default is "*inetOrgPerson*". For groups, the property **pa.ldap.group.objectclass** has a default value "*groupOfUniqueNames*" which could be changed.

4.1.2.4.3. Configure LDAP authentication parameters

By default, the resource manager binds to LDAP in anonymous mode. You can change this authentication method by modifying the property **pa.ldap.authentication.method**. This property can have several values:

- **none** (default value) - the resource manager performs connection to LDAP in anonymous mode.
- **simple** - the resource manager performs connection to LDAP with a specified login/password (see below for user password setting).
- You can also specify a SASL mechanism for LDAPv3. There are many SASL available mechanisms: **cram-md5**, **digest-md5**, **kerberos4**... Just put **sasl** to this property to let the resource manager JVM choose SASL authentication mechanism.

If you specify an authentication method different from 'none' (anonymous connection to LDAP), you must specify a login/password for authentication. There are two properties to set in LDAP configuration file:

- **pa.ldap.bind.login** - sets user name for authentication.
- **pa.ldap.bind.pwd** - sets password for authentication.

4.1.2.4.4. Set SSL/TLS parameters

The ProActive Resource Manager is able to communicate with LDAP with a secured SSL/TLS layer. It can be useful if your network is not trusted, and critical information are transmitted between the rm server and LDAP, such as user passwords. First, set the LDAP URL

property **pa.ldap.url** to a URL of type *ldaps://myLdap*. Then set **pa.ldap.authentication.method** to *none* so as to delegate authentication to SSL.

For using SSL properly, you have to specify your certificate and public keys for SSL handshake. Java stores certificates in a keyStore and public keys in a trustStore. In most of the cases, you just have to define a trustStore with public key part of LDAP's certificate. Put certificate in a keyStore, and public keys in a trustStore with the keytool command (keytool command is distributed with standard java platforms):

```
keytool -import -alias myAlias -file myCertificate -keystore myKeyStore
```

myAlias is the alias name of your certificate, **myCertificate** is your private certificate file and **myKeyStore** is the new keyStore file produced in output. This command asks you to enter a password for your keyStore.

Put LDAP certificate's public key in a trustStore, with the keytool command:

```
keytool -import -alias myAlias -file myPublicKey -keystore myTrustStore
```

myAlias is the alias name of your certificate's public key, **myPublicKey** is your certificate's public key file and **myTrustStore** is the new trustStore file produced in output. This command asks you to enter a password for your trustStore.

Finally, in **config/authentication/ldap.cfg**, set keyStore and trustStore created before to their respective passwords:

- Set **pa.ldap.keystore.path** to the path of your keyStore.
- Set **pa.ldap.keystore.passwd** to the password defined previously for keyStore.
- Set **pa.ldap.truststore.path** to the path of your trustStore.
- Set **pa.ldap.truststore.passwd** to the password defined previously for trustStore.

4.1.2.4.5. Use fall back to file authentication

You can use simultaneously file-based authentication and LDAP-based authentication. Then Resource Manager can check user password and group membership in login and group files, as performed in FileLogin method, if user or group is not found in LDAP. It uses **pa.rm.defaultloginfilename** and **pa.rm.defaultgroupfilename** files to authenticate user and check group membership. There are two rules:

- If LDAP group membership checking fails, fall back to group membership checking with group file. To activate this behavior set **pa.ldap.group.membership.fallback** to **true**, in LDAP configuration file.
- If a user is not found in LDAP, fall back to authentication and group membership checking with login and group files. To activate this behavior, set **pa.ldap.authentication.fallback** to **true**, in LDAP configuration file.

4.1.3. User authorization

All users authenticated in the resource manager have they own role according to granted permissions. In the resource manager, we use standard Java Authentication and Authorization Service (JAAS) to address these needs. Security support on the method call level is provided by ProActive programming. This mechanism will not be discussed here and is described in details in the [ProActive Programming documentation](#)¹.

On the resource manager, level permissions allow to:

- perform user actions, like get/release nodes, add/remove node, etc.
- access node sources and limit nodes utilization to particular user/group

Users are organized in groups and after authentication each of them has a single UserPrincipal and some GroupPrincipals (may not have them). Using this principals, the permissions are granted in the security policy file, like the following:

¹ http://proactive.inria.fr/trunk/Programming/AdvancedFeatures/multiple_html/Security.html

```
grant principal org.ow2.proactive.authentication.principals.UserNamePrincipal "john" {
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.getAtMostNodes";
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.releaseNodes";
};
```

This means that user "john" can request nodes for computations and release them. It cannot perform any administrative actions (they have to be listed explicitly).

Permissions could be granted to groups and in this case will be applicable to all group members. For example, we may define a group of users who provides computing resources. We allow them to call add/remove nodes methods, so that they will be able to add their nodes to the resource manager.

```
grant principal org.ow2.proactive.authentication.principals.GroupNamePrincipal "providers" {
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.getAtMostNodes";
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.getExactlyNodes";
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.releaseNode";
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.releaseNodes";

  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.addNode";
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.removeNode";

  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.getNodesList";
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.getNodeSourcesList";
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.getFreeNodesNumber";
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.getTotalNodesNumber";
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.getTotalAliveNodesNumber";
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.getRMState";
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.getState";
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.isActive";
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.isAlive";
  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.getMonitoring";

  permission
  org.ow2.proactive.permissions.MethodCallPermission "org.ow2.proactive.resourcemanager.core.RMCore.disconnect";
};
```

But having the permission of add/remove method call is not enough to actually add nodes. As all the nodes in the resource manager are organized in node sources user has to have the node source administration permission. This permission cannot be granted in java security policy file and defines at node source creation time by specifying which group or user will use nodes from this source and administrate them.

There is a permission which authorized to perform any actions. For this needs, we have implemented custom *org.ow2.proactive.permissions.AllPermission* to separate an absolute freedom inside the resource manager from JVM. The following is the usage of this permission in group of administrators:

```
grant principal org.ow2.proactive.authentication.principals.GroupNamePrincipal "admin" {
    permission org.ow2.proactive.permissions.AllPermission;
};
```

The longest security check is at the moment of removing node. Here we verify that the user

- can call *removeNode* method
- can administrate the node source where the node is
- is the one who added the node (or has AllPermission)

In order to use JMX monitoring interface, the standart *javax.management.MBeanPermission* has to be granted. Through JMX, we expose an account data as well and for non-admin users we show them only their accounts

```
grant principal org.ow2.proactive.authentication.principals.GroupNamePrincipal "user" {
    ...
    // AuthPermission is requires for those who would like to access any mbean
    permission javax.security.auth.AuthPermission "getSubject";
    permission javax.management.MBeanPermission "-#-[-]", "queryNames";
    permission javax.management.MBeanPermission "javax.management.MBeanServerDelegate#-
[JMIImplementation:type=MBeanServerDelegate]", "addNotificationListener";
    permission
    javax.management.MBeanPermission "org.ow2.proactive.scheduler.core.jmx.mbean.MyAccountMBeanImpl#*[*:*]", "***";
    permission
    javax.management.MBeanPermission "org.ow2.proactive.resourcemanager.core.jmx.mbean.MyAccountMBeanImpl#*[*:*]",
    permission
    javax.management.MBeanPermission "org.ow2.proactive.resourcemanager.core.jmx.mbean.RuntimeDataMBeanImpl#*[*:*]",
    ...
};
```

The default java security file used with the resource manager is located in config directory **\$RM_HOME/config/security.java.policy-server**.

4.2. The command line interface

This part explains how to launch the ProActive Resource Manager and interact with it using command line interface (CLI). In order to use this interface the *JAVA_HOME* environment variable has to be defined.

4.2.1. Launching the resource manager

To start the resource manager, run the **rm-start[.bat]** or **rm-start-clean[.bat]** script in bin/[OS]/ directory. The second script drops all the data from data base used to store the history of RM. When launching the resource manager, it is possible to specify few arguments:

- **-ln** or **--localNodes**, start the resource manager with 4 local nodes
- **-d** or **--deploy**, followed by a list of GCM deployment files (GCMD, see below for GCM utilisation) deploys ProActive nodes described there.
- **-h** or **--help** - prints the command line help.

4.2.2. Interacting with the resource manager

rm-client[.bat], in `bin/[os]` directory, provides a way to connect to the resource manager and manipulate it in command line mode. This scripts supports a bunch of options:

```
$RM_HOME/bin/unix$ rm-client -h

usage: rm-client [-a <node URLs> | -cn <names> | -d <node URLs> | -ln | -lns | -ma | -ni <nodeURL> | -r <names> | -rp | -s | -stats | -t | -ua <username>] [-c <arg>] [-env <filePath>] [-f] [-g] [-h] [-i <params>] [-l <login>] [-ns <nodes URLs>] [-p <params>] [-sf <filePath arg1=val1 arg2=val2 ...>] [-u <rmURL>]
-a,--addnodes <node URLs>          <ctl> Add nodes by their URLs
-c,--credentials <arg>              Path to the credentials (/home/jlscheef/.proactive/security/creds.enc).
-cn,--createns <names>              <ctl> Create new node sources
-d,--removenodes <node URLs>        <ctl> Remove nodes by their URLs
-env,--environment <filePath>      Execute the given script and go into interactive mode
-f,--force                          <ctl> Do not wait for busy nodes to be freed before nodes removal, node source removal and shutdown actions
-g,--gui                             (-d, -r and -s) Start the console in a graphical view
-h,--help                            Display this help
-i,--infrastructure <params>        Specify an infrastructure when node source is created
-l,--login <login>                  The username to join the Resource Manager
-ln,--listnodes                       <ctl> List nodes handled by Resource Manager. Display is : NODESOURCE HOSTNAME STATE NODE_URL
-lns,--listns                         <ctl> List node sources on Resource Manager. Display is : NODESOURCE TYPE
-ma,--myaccount                       <ctl> Display current user account informations
-ni,--nodeinfo <nodeURL>             <ctl> Display node information
-ns,--nodesource <nodes URLs>       <ctl> Specify an existing node source name for adding nodes
-p,--policy <params>                 Specify a policy when node source is created
-r,--removens <names>                <ctl> Remove given node sources
-rp,--reloadpermissions               <ctl> Reload the permission file
-s,--shutdown                         <ctl> Shutdown Resource Manager
-sf,--script <filePath arg1=val1 arg2=val2 ...> <ctl> Execute the given javascript file with optional arguments.
-stats,--statistics                   <ctl> Display some statistics about the Resource Manager
-t,--topology                         <ctl> Displays nodes topology.
-u,--rmURL <rmURL>                  The Resource manager URL (default rmi://localhost:1099/)
-ua,--useraccount <username>        <ctl> Display account information by username
```

NOTE : if no `<ctl>` command is specified, the controller will start in interactive mode.

When new node source is created custom infrastructure or policy can be specified:

- *-infrastructure* parameter allows to list or specify an infrastructure. In order to see the list of supported infrastructures run

```
$RM_HOME/bin/unix$ rm-client -l admin -cn myns -infrastructure
```

Available node source infrastructures:
Name: Local Infrastructure

Description: Deploys nodes on Resource Manager's machine

Class name: org.ow2.proactive.resourcemanager.nodesource.infrastructure.LocalInfrastructure

Parameters: <class name> rmUrl[rmi://slave1:2007/] credentials maxNodes[4] nodeTimeout[5000] paProperties

Name: CLI Infrastructure

Description: Creates remote runtimes using custom scripts

Class name: org.ow2.proactive.resourcemanager.nodesource.infrastructure.CLIInfrastructure

Parameters: <class name> rmUrl[rmi://slave1:2007/] interpreter[bash] deploymentScript removalScript hostsList nodeTimeOut[60000] maxDeploymentFailure[5]

Name: Default Infrastructure Manager

Description: Default infrastructure

Class name: org.ow2.proactive.resourcemanager.nodesource.infrastructure.DefaultInfrastructureManager

Parameters: <class name> rmUrl[rmi://slave1:2007/]

Name: SSH Infrastructure

Description: Creates remote runtimes using SSH

Class name: org.ow2.proactive.resourcemanager.nodesource.infrastructure.SSHInfrastructure

Parameters: <class name> rmUrl[rmi://slave1:2007/] sshOptions javaPath[jdk/bin/java] schedulingPath[SCHEDULER_HOME] nodeTimeOut[60000] maxDeploymentFailure[5] targetOs[Linux] javaOptions rmCredentialsPath hostsList

Name: Generic Batch Job Infrastructure

Description: Acquires nodes from a GENERIC resource manager.

Class name: org.ow2.proactive.resourcemanager.nodesource.infrastructure.GenericBatchJobInfrastructure

Parameters: <class name> rmUrl[rmi://slave1:2007/] javaPath[jdk/bin/java] sshOptions schedulingPath[SCHEDULER_HOME] javaOptions maxNodes[1] nodeTimeOut[300000] serverName rmCredentialsPath submitJobOpt implementationClassname implementationFile

Name: GCM Infrastructure

Description: [DEPRECATED] Infrastructure described in GCM deployment descriptor

Class name: org.ow2.proactive.resourcemanager.nodesource.infrastructure.GCMInfrastructure

Parameters: <class name> rmUrl[rmi://slave1:2007/] descriptor

Name: GCM Customised Infrastructure

Description: [DEPRECATED] Handles hosts from the list using specified gcm deployment descriptor template with HOST java variable contract (see proactive documentation)

Class name: org.ow2.proactive.resourcemanager.nodesource.infrastructure.GCMCustomisedInfrastructure

Parameters: <class name> rmUrl[rmi://slave1:2007/] descriptor hostsList timeout[60000]

Name: LSF Infrastructure

Description: Acquires nodes from a LSF resource manager.

Class name: org.ow2.proactive.resourcemanager.nodesource.infrastructure.LSFInfrastructure

Parameters: <class name> rmUrl[rmi://slave1:2007/] javaPath[jdk/bin/java] sshOptions schedulingPath[SCHEDULER_HOME] javaOptions maxNodes[1] nodeTimeOut[300000] serverName rmCredentialsPath submitJobOpt

Name: Virtual Infrastructure

Description: Virtualized Infrastructure node acquisition

Class name: org.ow2.proactive.resourcemanager.nodesource.infrastructure.VirtualInfrastructure

Parameters: <class name> rmUrl[rmi://slave1:2007/] infrastructure VMMUrl VMMUser VMMPwd VMTTemplate VMMMax[0] hostCapacity[0] PAConfig RMCredentials

Name: PBS Infrastructure

Description: Acquires nodes from a PBS resource manager.

Class name: org.ow2.proactive.resourcemanager.nodesource.infrastructure.PBSInfrastructure

Parameters: <class name> rmUrl[rmi://slave1:2007/] javaPath[jdk/bin/java] sshOptions schedulingPath[SCHEDULER_HOME] javaOptions maxNodes[1] nodeTimeOut[300000] serverName rmCredentialsPath submitJobOpt[-l "nodes=1:ppn=1"]

Name: EC2 Infrastructure

Description: Handles nodes from the Amazon Elastic Compute Cloud Service.

Class name: org.ow2.proactive.resourcemanager.nodesource.infrastructure.EC2Infrastructure

Parameters: <class name> rmUrl[rmi://slave1:2007/] configurationFile RMCredentialsPath nodeHttpPort[80]

Name: WinHPC Infrastructure

Description: Windows HPC infrastrucure

Class name: org.ow2.proactive.resourcemanager.nodesource.infrastructure.WinHPCInfrastructure

Parameters: <class name> rmUrl[rmi://slave1:2007/] maxNodes[1] serviceUrl[https://<computerName>/HPCBasicProfile] userName password trustStore trustStorePassword javaPath[jre/bin/java] rmPath[SCHEDULER_HOME] RMCredentialsPath javaOptions extraClassPath timeout[60000]

Choose the one you need and add the corresponding parameters to the command line.

- *-policy* parameter allows to list or specify a policy. First see the list of available policies:

```
$RM_HOME/bin/unix$ rm-client -l admin -cn myns -infrastructure
org.ow2.proactive.resourcemanager.nodesource.infrastructure.DefaultInfrastructureManager -policy
```

Available node source policies:

Name: Release Resources When Scheduler Idle

Description: Releases all resources when scheduler is idle for specified time. Acquires them back on job submission.

Class name: org.ow2.proactive.scheduler.resourcemanager.nodesource.policy.ReleaseResourcesWhenSchedulerIdle

Parameters: <class name> nodeUsers[ALL] nodeProviders[ME] schedulerUrl schedulerCredentialsPath idleTime[60000]

Name: Cron Load Based Policy

Description: Triggers new nodes acquisition when scheduler is overloaded within a time slot defined in crontab syntax.

Class name: org.ow2.proactive.scheduler.resourcemanager.nodesource.policy.CronLoadBasedPolicy

Parameters: <class name> nodeUsers[ALL] nodeProviders[ME] schedulerUrl schedulerCredentialsPath refreshTime[1000] minNodes[0] maxNodes[10] loadFactor[10] nodeDeploymentTimeout[10000] acquisitionAllowed[* * * * *] acquisitionForbidden[* * * * *] preemptive[false] allowed[false]

Name: EC2 Policy

Description: Allocates as many resources as scheduler required according to loading factor. Releases resources smoothly.

Class name: org.ow2.proactive.scheduler.resourcemanager.nodesource.policy.EC2Policy

Parameters: <class name> nodeUsers[ALL] nodeProviders[ME] schedulerUrl schedulerCredentialsPath refreshTime[1000] minNodes[0] maxNodes[10] loadFactor[10] nodeDeploymentTimeout[2400000]

Name: Time Slot Policy

Description: Acquires and releases nodes at specified time.

Class name: org.ow2.proactive.resourcemanager.nodesource.policy.TimeSlotPolicy

Parameters: <class name> nodeUsers[ALL] nodeProviders[ME] acquireTime[7/18/11 10:31:13 AM CEST] releaseTime[7/18/11 11:31:13 AM CEST] period[86400000] preemptive[true]

Name: Cron Slot Load Based Policy

Description: Keeps all nodes up and running within specified time slot and acquires node on demand when scheduler is overloaded at another time.

Class name: org.ow2.proactive.scheduler.resourcemanager.nodesource.policy.CronSlotLoadBasedPolicy
 Parameters: <class name> nodeUsers[ALL] nodeProviders[ME] schedulerUrl schedulerCredentialsPath
 refreshTime[1000] minNodes[0] maxNodes[10] loadFactor[10] nodeDeploymentTimeout[10000] deployAllAt[* * * * *]
 undeployAllAt[* * * * *] preemptive[false] acquireNow[false]

Name: Scheduler Loading Policy

Description: Allocates as many resources as scheduler required according to loading factor. Releases resources smoothly.

Class name: org.ow2.proactive.scheduler.resourcemanager.nodesource.policy.SchedulerLoadingPolicy
 Parameters: <class name> nodeUsers[ALL] nodeProviders[ME] schedulerUrl schedulerCredentialsPath
 refreshTime[1000] minNodes[0] maxNodes[10] loadFactor[10] nodeDeploymentTimeout[10000]

Name: Static Policy

Description: Static nodes acquisition.

Class name: org.ow2.proactive.resourcemanager.nodesource.policy.StaticPolicy
 Parameters: <class name> nodeUsers[ALL] nodeProviders[ME]

Name: Cron Policy

Description: Acquires and releases nodes at specified time.

Class name: org.ow2.proactive.resourcemanager.nodesource.policy.CronPolicy
 Parameters: <class name> nodeUsers[ALL] nodeProviders[ME] nodeAcquisition[* * * * *] nodeRemoval[* * * * *]
 preemptive[false] forceDeployment[false]

Then select the appropriate one and specify it in the command line. Example:

```
>rm-client -l admin -cn myns -infrastructure
org.ow2.proactive.resourcemanager.nodesource.infrastructure.DefaultInfrastructureManager rmURL -policy
org.ow2.proactive.resourcemanager.nodesource.policy.StaticPolicy ALL ME
```

4.2.2.1. Interactive mode

Another way to interact with the resource manager is using the interactive mode of the controller. Interactive mode correspond to a JavaScript console, which is able to manage the resource manager. To start it in this mode, just launch the **bin/[os]/rm-client[.bat]** command without any parameter. Once connected, type ? or help() to get the list of provided functions:

```
exMode(display,onDemand)    Change the way exceptions are displayed (if display is true, stacks are displayed - if
onDemand is true, prompt before displaying stacks)
addnode(nodeURL, nsName)    Add node to the given node source (parameters is a string representing the node URL
to add and an optional string representing the node source in which to add the node)
removenode(nodeURL,preempt) Remove the given node (parameter is a string representing the node URL, node is
removed immediately if second parameter is true)
createns(nsName,infr,pol)   Create a new node source with specified name, infrastructure and policy (e.g.
createns('myname', ['infrastrucure', 'param1', ...], ['policy', 'param1', ...]))
removens(nsName,preempt)   Remove the given node source (parameter is a string representing the node source
name to remove, node source is removed immediately if second parameter is true)
listnodes()                List every handled nodes
listns()                   List every handled node sources
listInfrastructures()      List supported infrastructures
listPolicies()             List available node sources policies
shutdown(preempt)         Shutdown the Resource Manager (RM shutdown immediately if parameter is true)
jmxinfo()                 Display some statistics provided by MBean
exec(scriptFilePath)      Execute the content of the given script file (parameter is a string representing a script-file
path)
setLogsDir(logsDir)       Set the directory where the log are located, (default is RM_HOME/.logs
viewlogs(nbLines)         View the last nbLines lines of the logs file, (default nbLines is 20)
```

refreshPermissions()	Reload permissions policy file
exit()	Exits RM controller

4.2.2.1.1. execute actions in a Java script file

You can script a sequence of commands to send to Resource Manager in a javascript file:

```
/* set exec mode in order to not display stack trace, and not ask for either. */
exMode(false,false);

/* remove a node source, preemptively */
removens("my_node_source",true);

/* create a new node source with GCMD and static nodes acquisition*/
createns("my_node_source",
[org.ow2.proactive.resourcemanager.nodesource.infrastructure.GCMinfrastructure', 'rmURL', 'path_to_descriptor'],
[org.ow2.proactive.resourcemanager.nodesource.policy.StaticPolicy', 'ALL', 'ME']);

/* wait for a while */
java.lang.Thread.sleep(10000);

/* check that deployment has succeed */
listnodes();
```

Then run this script by typing in interactive mode:

```
> exec("myScript.js");
```

4.3. Organizing your nodes

The ProActive Resource Manager supports nodes aggregation from heterogeneous environments. As a node is just a JVM running somewhere, the process of communication to such nodes is unified and defined by ProActive library. The only part which has to be defined is the procedure of nodes deployment which could be quite different depending on infrastructures and their limitations. After [installation](#) of the server and node parts it is possible to configure an automatic nodes deployment. Basically, you can say to the resource manager how to launch JVMs with ProActive nodes and when.



Note

Lets give an example here. You have a cluster of linux-x64 machines permanently at your disposal with ssh access to all hosts and a cluster running Windows HPC where you would like to run computations periodically (let us say from 12 to 14 every day). In order to describe such a kind of behavior, we create 2 node sources using GUI or command line interface. The first one is the node source with "ssh infrastructure" and static deployment policy. The command would be the following:

```
$RM_HOME>./bin/unix/rm-client --createns "MyCluster" -infrastructure
org.ow2.proactive.resourcemanager.nodesource.infrastructure.SSHInfrastructure
rmURL sshOptions javaPath schedulingPath nodeTimeout attempt targetOS javaOptions rmCreds hostsList
```

The semantic of this command is quite straight forward, namely the resource manager tries immediately to launch the command of JVM creation through ssh using a special class which creates ProActive nodes inside JVMs and register them in the resource manager. Static node acquisition is default and you do not have to explicitly specify it.

The procedure of deployment on Windows HPC is more tricky because it involves the native scheduler and in order to take into account other users of the cluster, we have to go through it. Basically, what the resource manager does is contact

the windows scheduler exposed as a web service (it requires additional configuration of windows cluster) and submit a job launching the JVM process with ProActive. Here we also describe when the deployment has to be triggered using time slot policy.

```
$RM_HOME>./bin/unix/rm-client --createns "WindowsCluster" -infrastructure
org.ow2.proactive.resourcemanager.nodesource.infrastructure.WinHPCInfrastructure
rmURL maxNodes serviceUrl userName password trustStore trustStorePassword javaPath rmPath
RMCredentialsPath javaOptions extraClassPath timeout
-policy org.ow2.proactive.resourcemanager.nodesource.policy.TimeSlotPolicy nodesAvailableTo
administrator acquireTime releaseTime period preemptive
```

As you can see in the example above, in order to create a node source, you have to define two entities **infrastructure manager** and **node source policy**.

Infrastructure manager is responsible for communicating with an infrastructure. When a new node has to be deployed, an infrastructure manager will launch new JVM or just request an already existing nodes running somewhere. All these details are specific to the infrastructure manager implementation.



Warning

ProActive Scheduling supports only one node per JVM. So that if you're configuring the resource manager to be used with scheduler take it into account. In the following, the term "node" always refers to a **single node deployed in a dedicated JVM**.

Node source policy is a set of rules and conditions which describes when and how many nodes have to be acquired or released. Policies use node source API to manage the node acquisition.

Node sources were designed in a way that:

- All logic related to node acquisition is encapsulated in the infrastructure manager.
- Conditions and rules of node acquisition is described in the node source policy.
- Permissions to the node source. Each policy has two parameters:
 - **nodeUsers** - utilization permission defined who can get nodes for computations from this node source. It has to take one of the following values:
 - "ME" - Only the node source creator
 - "MY_GROUPS" - Only groups in which the node source creator belongs to
 - "PROVIDER" - Only the node provider who added the node. Provider can use only its nodes
 - "PROVIDER_GROUPS" - Only groups in which the node provider belongs to. Users from this groups can user only nodes of this provider
 - "ALL" - Everybody
 - **nodeProviders** - Provider permission defines who can add nodes to this node source. It should take one of the following values:
 - "ME" - Only the node source creator
 - "MY_GROUPS" - Only the group in which the node source creator belongs to
 - "ALL" - Everybody

The user created the node source is the administrator of this node source. It can add and removed nodes to it, remove the node source itself, but cannot use nodes if usage policy is set to PROVIDER or PROVIDER_GROUPS (unless it's granted AllPermissions).

- New infrastructure manager or node source policy can be dynamically plugged into the Resource Manager. In order to do that, it is just required to add new implemented classes in the class path and update corresponding list in the configuration file (`{RM_HOME}/config/rm/nodesource`).

In the resource manager, there is always a default node source consisted of DefaultInfrastructureManager and Static policy. It is not able to deploy nodes anywhere but makes it possible to add existing nodes to the RM.

4.3.1. Default infrastructure

Default infrastructure manager is designed to be used with ProActive agent. It cannot perform an automatic deployment but any users (including an agent) can add already existing nodes into it. In order to create a node source with this infrastructure, run the following command:

```
$RM_HOME>./bin/unix/rm-client --createns defaultns -infrastructure
org.ow2.proactive.resourcemanager.nodesource.infrastructure.DefaultInfrastructureManager rmURL
```

The only parameter to provide is the following one:

- **rmURL** - the URL of the resource manager.

4.3.2. Local infrastructure

Local Infrastructure Manager can be used to start nodes locally, i.e, on the host running the Resource Manager. In order to create a node source with this infrastructure, run the following command:

```
$RM_HOME>./bin/unix/rm-client --createns localns -infrastructure
org.ow2.proactive.resourcemanager.nodesource.infrastructure.LocalInfrastructure RmUrl credentialsPath
numberOfNodes timeout javaProperties
```

- **RMUrl** - Url of the resource manager server (can leave it empty to use default value, i.e, the url of the resource manager you connect to)
- **credentialsPath** - The absolute path of the credentials file used to set the provider of the nodes.
- **numberOfNodes** - The number of nodes to deploy.
- **timeout** - The length in ms after which one a node is not expected anymore.
- **javaProperties** - The java properties to setup the ProActive environment for the nodes

4.3.3. GCM customized infrastructure

GCM customized infrastructure can deploy/remove a single node to/from the infrastructure described in GCM descriptor. It makes possible to use this infrastructure manager together with more precise policies such as "scheduler aware policy" (see below). In order to create such node source user has to specify GCMD and hosts list. GCMD has to define a deployment of a single node and does not have to have an explicit host names. Instead of this, `{HOST}` variable must be used and at the moment of deployment the resource manager associates host to the gcmd (see `$RM_HOME/config/rm/deployment/deployment_ssh_hosts_list_template.xml`).

```
$RM_HOME>./bin/unix/rm-client --createns gcmns -infrastructure
org.ow2.proactive.resourcemanager.nodesource.infrastructure.GCMCustomisedInfrastructure rmURL config/rm/
deployment/deployment_ssh_hosts_list_template.xml config/rm/deployment/hostlist nodeTimeout
```

This infrastructure needs 4 arguments, described hereafter:

- **RMUrl** - Url of the resource manager server
- **descriptor** - path to a template GCM Deployment descriptor file where the `{HOST}` variable is used.
- **hostsList** - path to a file containing a list of hosts (one host per line).
- **timeout** - The length in ms after which one a node is not expected anymore.

Even if GCM and GCMCustomized infrastructures used the same GCM Application descriptor file, only GCM Customized makes an optimal usage of it. The main difference comes from the fact that the GCM Customized infrastructure manager predicts the deployment of the nodes by giving them a name. This name can directly be set by the infrastructure manager filling in a variable contract (name `jvmargDefinedByIM`) of the xml descriptor file.

4.3.4. SSH infrastructure

This infrastructure allows to deploy nodes over ssh. Having a list of hosts the resource manager construct the ssh command to launch remote JVMs. Java path and rm distribution path on remote hosts have to be specified together with other parameters. Example:

This infrastructure needs 10 arguments, described hereafter:

- **Rm Url** - The resource manager's url that will be used by the deployed nodes to register by themselves.
- **SSH Options** - Options you can pass to the SSHClient executable (-l inria to specify the user for instance)
- **Java Path** - Path to the java executable on the remote hosts.
- **Scheduling Path** - Path to the Scheduling/RM installation directory on the remote hosts.
- **Node Time Out** - A duration after which one the remote nodes are considered to be lost.
- **Attempt** - The number of time the resource manager tries to acquire a node for which one the deployment fails before discarding it forever.
- **Target OS** - One of 'LINUX', 'CYGWIN' or 'WINDOWS' depending on the machines' (in Hosts List file) operating system.
- **Java Options** - Java options appended to the command used to start the node on the remote host.
- **Rm Credentials Path** - The absolute path of the 'rm.cred' file to make the deployed nodes able to register to the resource manager (config/authentication/rm.cred).
- **Hosts List** - Path to a file containing the hosts on which resources should be acquired. This file should contain one host per line, described as a host name or a public IP address, optionally followed by a positive integer describing the number of runtimes to start on the related host (default to 1 if not specified). Example:

```
rm.example.com
test.example.net 5
192.168.9.10 2
```

4.3.5. Command Line infrastructure

This generic infrastructure allows to deploy nodes using deployment script written in arbitrary language. The infrastructure just launches this script and waits until the ProActive node is registered in the resource manager. Command line infrastructure could be used when you prefer to describe the deployment process using shell scripts instead of Java. Script examples could be found in RM_HOME/samples/scripts/deployment. The deployment script has 4 parameters: HOST_NAME, NODE_NAME, NODE_SOURCE_NAME, RM_URL. The removal script has 2 parameters: HOST_NAME and NODE_NAME.

This infrastructure needs 7 arguments, described hereafter:

- **Rm Url** - The resource manager's url that will be used by the deployed nodes to register by themselves.
- **Interpreter** - Path to the script interpreter (bash by default).
- **Deployment Script** - A script that launches a ProActive node and register it to the RM.
- **Removal Script** - A script that removes the node from the resource manager.
- **Hosts List** - Path to a file containing the hosts on which resources should be acquired. This file should contain one host per line, described as a host name or a public IP address, optionally followed by a positive integer describing the number of runtimes to start on the related host (default to 1 if not specified). Example:

```
rm.example.com
test.example.net 5
192.168.9.10 2
```

- **Node Time Out** - The length in ms after which one a node is not expected anymore.
- **Max Deployment Failure** - the number of times the resource manager tries to relaunch the deployment script in case of failure.

4.3.6. Windows HPC infrastructure

The deployment through Windows HPC scheduler. In order to make it functional, the Windows HPC has to be configured according to [this guide](http://technet.microsoft.com/en-us/library/cc972837(WS.10).aspx)². After exposing windows native scheduler as a web service it is possible to contact it from Java and submit any command. In our case, it is a command launching JVM on one of the cluster nodes.

```
$RM_HOME>./bin/unix/rm-client --createns winhpc -infrastructure
org.ow2.proactive.resourcemanager.nodesource.infrastructure.WinHPCInfrastructure rmURL 8 https://cluster_name/
```

² [http://technet.microsoft.com/en-us/library/cc972837\(WS.10\).aspx](http://technet.microsoft.com/en-us/library/cc972837(WS.10).aspx)

HPCBasicProfile username password trustStore trustStorePassword javaPath rmPath RMUrl RMCredentialsPath
javaOptions extraClassPath nodeTimeout

This infrastructure needs 12 arguments, described hereafter:

- **RMUrl** - Url of the resource manager server
- **maxNodes** - Maximum number of nodes to deploy.
- **serviceUrl** - Url of the WinHPC web service.
- **userName** - username for the web service connection
- **password** - password for the web service connection
- **trustStore** - name of the trustStore
- **trustStorePassword** - password of the trustStore
- **javaPath** - Path to the java executable on the WinHPC server.
- **rmPath** - Path to the Resource Manager directory on the WinHPC server.
- **RMCredentialsPath** - Path to the RM Credentials
- **javaOptions** - Java options appended to the command used to start the node on WinHPC server.
- **extraClassPath** - Extra class path to be added.
- **timeout** - The length in ms after which one a node is not expected anymore.

4.3.7. Amazon EC2 Infrastructure

4.3.7.1. Overview

The Elastic Compute Cloud, aka **EC2**, is an Amazon Web Service, that allows its users to use machines (instances) on demand on the cloud. An EC2 instance is a Xen virtual machine, running on different kinds of hardware, at different prices, but always paid by the hour, allowing lots of flexibility. Being virtual machines, instances can be launched using custom operating system images, called **AMI** (Amazon Machine Image). For the Resource Manager to use EC2 instances as computing nodes, a specific EC2 Infrastructure as well as AMI creation utilities are provided.

In order to use EC2 instances in the Resource Manager, it is recommended to take the following steps:

- [Section 4.3.7.2, “AWS Account”](#) describes the minimum configuration needed to deploy nodes on EC2
- [Section 4.3.7.3, “Deploying Nodes”](#) gives a simple example with the default provided configuration, once your AWS account and local configuration are setup.

4.3.7.2. AWS Account

To use the EC2 Infrastructure in the Resource Manager, proper Amazon credentials are needed. This section describes briefly how to obtain them, and how to use them to configure your environment.

1. First, you need to create an AWS account at <http://aws.amazon.com/>.
2. With your new AWS account, sign up for EC2 at <http://aws.amazon.com/ec2/>.
3. Now, you need to obtain the credentials. [On the AWS website](#)³, point your browser to the **Your Web Services Account** button, a drop down list displays. Click **View Access Key Identifiers**.
4. Use this information to fill in the properties **AWS_AKEY** (Access Key), **AWS_SKEY** (Secret Key) and **AWS_USER** (numerical EC2 user ID) in the file located in **/config/rm/deployment/ec2.properties**. Those three parameters should never change, except if you need for some reason to handle multiple EC2 accounts. Other properties in the configuration file are:
 - **AMI**: Defines which AMI will be deployed on instances. The value to provide is the unique AMI Id, ami-XXX. If no default value is provided, you need to find a public ProActive AMI corresponding to the release you are using.
 - **INSTANCE_TYPE**: One of m1.small, m1.large, m1.xlarge, c1.medium, c1.xlarge, which defines what kind of hardware the instance will be running on. m1 instances are memory focused, whereas c1 instances are for more CPU intensive tasks. This property defaults is m1.small, the cheapest and slowest of all.

³ <http://aws.amazon.com/>



Warning

Only m1.xlarge and c1.xlarge instances are able to launch x86_64 AMI. When trying to run such AMI, if INSTANCE_TYPE is not one of the xlarge, it will be forced to m1.xlarge. If you want to control the cost of your instances, do not run x86_64 AMIs.

- **MAX_INST:** Sets a maximum number of instances that can be deployed simultaneously on this infrastructure, which is useful to control the cost of instances.

4.3.7.3. Deploying Nodes

Using the Scheduler and the Resource Manager with an EC2 NodeSource requires specific configuration. EC2 instances, indeed, are located on a private network, behind a NAT device: this allows fast communications between two EC2 instances, but this is an issue when using the RMI protocol. The workaround is to use a PAMR router, or the HTTP protocol.

For that reason, the HTTP protocol is forced for communications going from the RM to the EC2 nodes. On the other way, from EC2 to the RM, using HTTP communications is also the simplest, also RMISsh is possible but requires manual configuration for deploying RSA keys. To configure your RM with HTTP, all you need to specify is a configuration file with the following properties, for both Resource Manager **and** Scheduler.

```
<prop key="proactive.net.nolocal" value="true"/>
<prop key="proactive.communication.protocol" value="http" />
<prop key="proactive.http.port" value="PORT" />
<prop key="proactive.net.noprivate" value="true" />
<prop key="proactive.useIpAddress" value="true" />
```

Note that you need to specify a different port for both configurations. Sample configurations can be found in `/samples/ec2`.

Using this configuration, you can start a Resource Manager and a Scheduler using the `/bin/unix/rm-start` script. An EC2 NodeSource can now be added using the command line interface of the Resource Manager:

```
$RM_HOME>./bin/unix/rm-client --createn ec2 -infrastructure
org.ow2.proactive.resourcemanager.nodesource.infrastructure.EC2Infrastructure ./config/rm/deployment/ec2.properties
rmUrl RMCredentialsPath nodeHttpPort
```

You will need to specify:

- **configurationFile** - EC2 configuration file such as `/config/rm/deployment/ec2.properties`, which you should have filled with your AWS credentials beforehand.
- **rmUrl** - a fully qualified URL conforming to `protocol://IP:port/`, where IP is the public IP of the machine where the Resource Manager is launched. Protocol and port are mandatory, as these values will be used to configure the ProActive runtime of the EC2 instances at boot. It describes the protocol and port used for communications from the instance to the Resource Manager.
- **RMCredentialsPath** - to register on the Resource Manager, nodes need these credentials to log in. That can be found in `/config/authentication/login.cfg` when using default configuration.
- **nodeHttpPort** - As stated before, the EC2 node will expose itself with the HTTP protocol. You can configure which port it will use. This is useful if the machine on which the RM is started has restrictive firewall rules for outgoing connections.

Once the Resource Manager is up and running with its EC2 NodeSource, new jobs can be added by using `/bin/unix/scheduler-admin`, or the Scheduler GUI.

4.3.8. Load Sharing Facility (LSF) infrastructure

This infrastructure knows how to acquire nodes from LSF by submitting a corresponding job. It will be submitted through SSH from the RM to the LSF server. As an alternative, you may consider using [the GCM Customized infrastructure](#) instead, which provides more control over the deployment, but requires more configuration.

```
$RM_HOME>./bin/unix/rm-client --createns lsf -infrastructure
org.ow2.proactive.resourcemanager.nodsource.infrastructure.LSFInfrastructure rmURL javaPath sshOptions
schedulingPath javaOptions maxNodes nodeTimeout LSFServer RMCredentialsPath bsubOptions
```

where:

- **RMUrl** - URL of the Resource Manager from the LSF nodes point of view - this is the URL the nodes will try to lookup when attempting to register to the RM after their creation.
- **javaPath** - path to the java executable on the remote hosts (ie the LSF slaves).
- **SSH Options** - Options you can pass to the SSHClient executable (-l inria to specify the user for instance)
- **schedulingPath** - path to the Scheduling/RM installation directory on the remote hosts.
- **javaOptions** - Java options appended to the command used to start the node on the remote host.
- **maxNodes** - maximum number of nodes this infrastructure can simultaneously hold from the LSF server. That is useful considering that LSF does not provide a mechanism to evaluate the number of currently available or idle cores on the cluster. This can result to asking more resources than physically available, and waiting for the resources to come up for a very long time as the request would be queued until satisfiable.
- **Node Time Out** - The length in ms after which one a node is not expected anymore.
- **Server Name** - URL of the LSF server, which is responsible for acquiring LSF nodes. This server will be contacted by the Resource Manager through an SSH connection.
- **RM Credentials Path** - Encrypted credentials file, as created by the create-cred[.bat] utility. These credentials will be used by the nodes to authenticate on the Resource Manager.
- **Submit Job Opt** - Options for the bsub command client when acquiring nodes on the LSF master. Default value should be enough in most cases, if not, refer to the documentation of the LSF cluster.

4.3.9. Portable Batch System (PBS) infrastructure

This infrastructure knows how to acquire nodes from PBS (i.e. Torque) by submitting a corresponding job. It will be submitted through SSH from the RM to the PBS server. As an alternative, you may consider using [the GCM Customized infrastructure](#) instead, which provides more control over the deployment, but requires more configuration.

```
$RM_HOME>./bin/unix/rm-client --createns pbs -infrastructure
org.ow2.proactive.resourcemanager.nodsource.infrastructure.PBSInfrastructure rmURL javaPath sshOptions
schedulingPath javaOptions maxNodes nodeTimeout PBSServer RMCredentialsPath qsubOptions
```

where:

- **RMUrl** - URL of the Resource Manager from the PBS nodes point of view - this is the URL the nodes will try to lookup when attempting to register to the RM after their creation.
- **javaPath** - path to the java executable on the remote hosts (ie the PBS slaves).
- **SSH Options** - Options you can pass to the SSHClient executable (-l inria to specify the user for instance)
- **schedulingPath** - path to the Scheduling/RM installation directory on the remote hosts.
- **javaOptions** - Java options appended to the command used to start the node on the remote host.
- **maxNodes** - maximum number of nodes this infrastructure can simultaneously hold from the PBS server. That is useful considering that PBS does not provide a mechanism to evaluate the number of currently available or idle cores on the cluster. This can result to asking more resources than physically available, and waiting for the resources to come up for a very long time as the request would be queued until satisfiable.
- **Node Time Out** - The length in ms after which one a node is not expected anymore.
- **Server Name** - URL of the PBS server, which is responsible for acquiring PBS nodes. This server will be contacted by the Resource Manager through an SSH connection.
- **RM Credentials Path** - Encrypted credentials file, as created by the create-cred[.bat] utility. These credentials will be used by the nodes to authenticate on the Resource Manager.
- **Submit Job Opt** - Options for the qsub command client when acquiring nodes on the PBS master. Default value should be enough in most cases, if not, refer to the documentation of the PBS cluster.

4.3.10. Generic Batch Job infrastructure

Generic Batch Job infrastructure provides users with the capability to add the support of new batch job scheduler by providing a class extending `org.ow2.proactive.resourcemanager.nodesource.infrastructure.BatchJobInfrastructure`. Once you have written that implementation, you can create a node source which makes usage of this infrastructure by running the following command:

```
$RM_HOME>./bin/unix/rm-client --createns pbs -infrastructure
org.ow2.proactive.resourcemanager.nodesource.infrastructure.GenericBatchJobInfrastructure rmURL javaPath
sshOptions schedulingPath javaOptions maxNodes nodeTimeout BatchJobServer RMCredentialsPath subOptions
implementationName implementationPath
```

where:

- **RMUrl** - URL of the Resource Manager from the batch job scheduler nodes point of view - this is the URL the nodes will try to lookup when attempting to register to the RM after their creation.
- **javaPath** - path to the java executable on the remote hosts (ie the slaves of the batch job scheduler).
- **SSH Options** - Options you can pass to the SSHClient executable (-l inria to specify the user for instance)
- **schedulingPath** - path to the Scheduling/RM installation directory on the remote hosts.
- **javaOptions** - Java options appended to the command used to start the node on the remote host.
- **maxNodes** - maximum number of nodes this infrastructure can simultaneously hold from the batch job scheduler server.
- **Node Time Out** - The length in ms after which one a node is not expected anymore.
- **Server Name** - URL of the batch job scheduler server, which is responsible for acquiring nodes. This server will be contacted by the Resource Manager through an SSH connection.
- **RM Credentials Path** - Encrypted credentials file, as created by the create-cred[.bat] utility. These credentials will be used by the nodes to authenticate on the Resource Manager.
- **Submit Job Opt** - Options for the submit command client when acquiring nodes on the batch job scheduler master.
- **implementationName** - Fully qualified name of the implementation of `org.ow2.proactive.resourcemanager.nodesource.infrastructure.BatchJobInfrastructure` provided by the end user.
- **implementationPath** - The absolute path of the implementation of `org.ow2.proactive.resourcemanager.nodesource.infrastructure.BatchJobInfrastructure`.

4.3.11. Virtualized infrastructure

4.3.11.1. Hardware Virtualization Overview

Virtualized Infrastructure here means a part of an Infrastructure that runs a virtualization software. This virtualized infrastructure can then be used as a resource pool for Resource Manager execution. A brief introduction to the installation and the use of the virtualization software handled by the Resource Manager is given below. The following section is dedicated to explain the use of the Resource Manager for such an infrastructure.

Hardware virtualization allows to run several operating systems on a unique machine. This is done thanks to a specific software called “Virtual Machine Monitor” (VMM). Every virtualization solution needs a particular operating system to work (“Dom0” in case of bare metal, VMM and “host OS” in case of hosted virtualization, we will dig deeper with this later on). The VMM can either emulates specific hardware devices or grants access to real hardware to the virtual machine (or guest operating system i.e. VM). One thus benefits from many features for different purposes. We identify 2 kinds of VMM:

- **Type 1 hypervisor** (or bare metal).

This type of hypervisor is named "bare metal" because it does not need any operating system beneath to work. Depending on what kind of product you use, you will benefit from different drivers for different hardware and, maybe, will not be able to get the software functional because of unsupported hardware (see a description at: <http://en.wikipedia.org/wiki/Hypervisor>). Most often, you will need a “DOM 0” virtual machine, which is in fact the operating system that stands for your workspace to manage your virtual infrastructure. The virtual machine monitor (VMM) itself is a small footprint software (about 50Mo) which will only check and schedule underlying hardware access. To set up your virtual environment you need an extra software (xm for XenOss, xe for XenServer, vmx for ESX ...) which is usable directly from your DOM 0. This type of hypervisor is, in general, more efficient

and faster than other hypervisors as it implements its own hardware access policy at the lowest possible level and since you avoid overhead induced by an underlying operating system. Here are some well-known type 1 hypervisors: [XenServer](#)⁴, [Hyper-V](#)⁵, [xVM Server](#)⁶, [VMware ESX/ESXi](#)⁷, [Xen OSS](#)⁸

- **Type 2 hypervisor** (or hosted).

This type of hypervisor is running on top of an operating system and is seen as a common process like every virtual machine containers. We can thus measure the host operating system overhead that treat the guest operating system as a child process whereas it was more a "brother" in the case of type one hypervisor. Furthermore, every kind of virtualization process cannot be used in hosted virtualization for some reasons (browse http://en.wikipedia.org/wiki/Platform_virtualization for more details). Here are the main type 2 hypervisors: [Virtualbox](#)⁹, [VMware Server](#)¹⁰, [KVM](#)¹¹

The most important point a user has to be informed of is the network part. We can essentially distinguish three sorts of network provisions:

- **Bridge Networking** - Here, the network routing is made at the 3rd level of the OSI/ISO stack. Your computer's NIC binded to your company/internet network is setup in "PROMISCUOUS MODE" to intercept not only packet intended for the host/dom0 IP but also for newly created Virtual interfaces designed to provide network to virtual machines. Depending on what kind of virtualization product you use, you may have to create the virtual interfaces by yourself (brctl/openvpn on linux systems & Network Manager on Windows). With this solution, your guest operating systems are part of the company network like your host computer. That means a fully point-to-point connectivity between both hardware/virtual machine on your network.
- **Nat/Route** - The network routing is also made at the 3rd level of the OSI/ISO stack. This time, a newly created virtual interface will ensure POST/PRE-ROUTING & MASQUERADE for your virtual machine to have network access. It is really easy to find such configuration example on the internet ([Here for linux for instance](#)¹²).
- **NAT user** - This time network routing is made at the virtualization software layer. For the outer world (the host machine included), the virtual machines are unreachable. Most often, the associated subnet has the netmask 10.0.0.1/24, 10.0.0.1 & 10.0.0.2 are the IPs for the host machine (only seen by the virtual machines) and every virtual machines belongs to its own subnet (virtual machines cannot speak to each other).

4.3.11.2. Virtualized Infrastructure Management

The way RM Nodes belonging to a virtualized infrastructure are acquired is divided in three steps:

- **Contact the virtual machine manager** and ask it to clone and power on the virtual machines that will be used to run RM Nodes. To get that done, you must provide the pieces of information required: the url of the virtual machine monitor (this one depends on the underlying virtual infrastructure, we will dig deeper into this subject later), a user of the virtual machine monitor as well as his password, the template virtual machine you want to use (most of the cases, the name of the virtual machine as it appears in your virtual infrastructure management tool), the maximum number of this instance the virtual machine monitor can handle. Some other information are needed but are related to the virtual infrastructure such as the host capacity (the number of RM Node that will be launched per virtual machine), the Resource Manager's url and credentials to be able to register the created RM Nodes.
- **Start the RM Nodes** once the virtual machine starts. This step requires the retrieval of the information provided in the previous step from the inside of the virtual machine. To get that done we provide template daemon files located in scripts/virtualization (rm-node-starter.bat & rm-node-starter.sh). These scripts needs additional tools to work. To get more information about how it works, refer to the section of the corresponding virtualization tool [VMware](#), [XenServer](#), [Virtualbox](#) or [Hyper-V](#).
- **Register RM Node** in the Resource Manager. This step can be done either by remote node registration or by local node registration. In the first case, the newly started nodes remotely connect to the Resource Manager and register themselves as available (this method is applied if you provide the Resource Manager's url). The latter ensures a local node registration, this means that once the RM Nodes are started, they communicate their url to the virtual machine monitor. The Resource Manager on its side "polls" the virtual machine monitor to know when the newly created RM Nodes are available and registers them. This second method is more

⁴ <http://community.citrix.com/cdn/xs>

⁵ <http://www.microsoft.com/hyper-v-server/en/us/default.aspx>

⁶ <http://www.sun.com/software/products/xvmserver/index.xml>

⁷ <http://www.vmware.com/products/vi/esx/>

⁸ <http://www.xen.org/>

⁹ <http://www.virtualbox.org/>

¹⁰ <http://www.vmware.com/products/server/>

¹¹ <http://www.qumranet.com/>

¹² <http://www.revsys.com/writings/quicktips/nat.html>

attractive if you want to use Scheduler aware policies for example because the Resource Manager will cache available nodes for futur reuse. In case of remote registration, you must ensure that the Resource Manager is able to contact RM Nodes (that means that the communication protocol used for communication is well suited for the kind of network connection used. For example, in case of NAT User, you will have to use ProActive Message Routing Protocol i.e. PAMR).

Here is a snapshot of the panel for the information keyboarding that you can find in the [Resource Manager GUI](#):

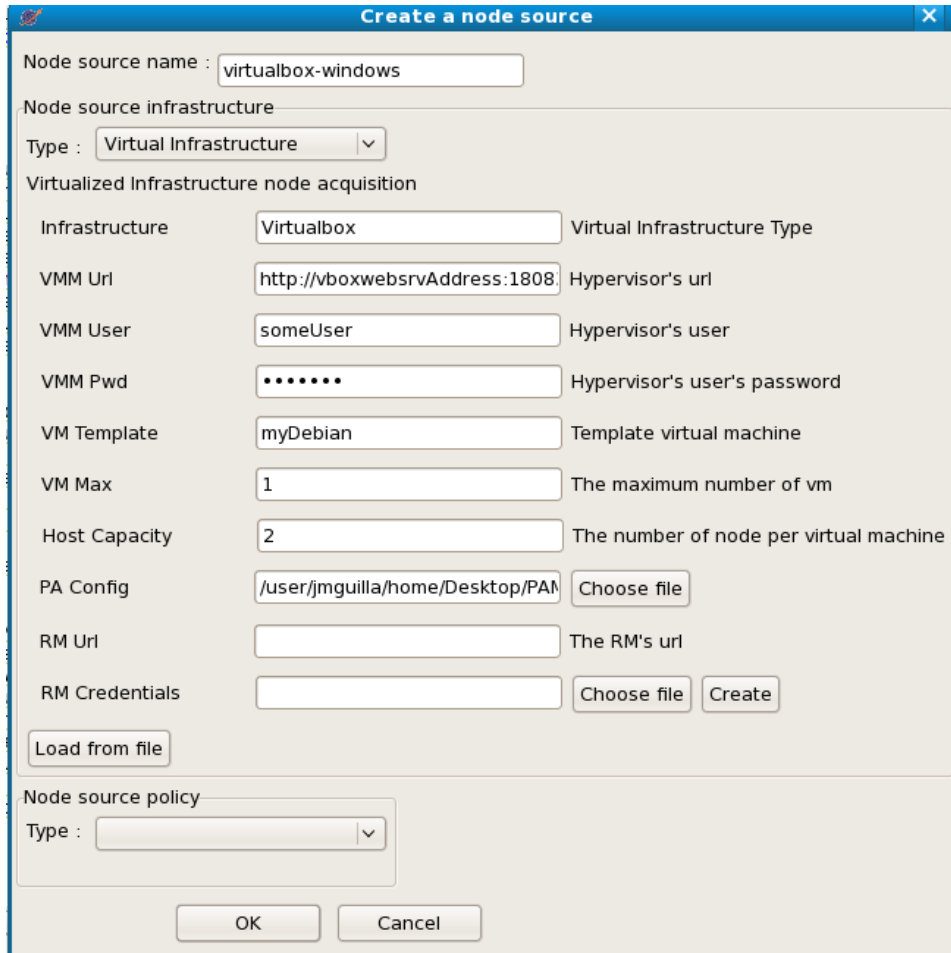


Figure 4.2. Virtual Infrastructure NodeSource creation dialog



Note

The "Load from file" button allows you to fill fields providing a file of "key = value" entries. The key must be the field names without whitespaces (case sensitive, an example can be found in `config/rm/deployment/vmm.properties`).

or

```
$RM_HOME>./bin/unix/rm-client -uc --createns virtual -infrastructure
org.ow2.proactive.resourcemanager.nodesource.infrastructure.VirtualInfrastructure RMUrl infrastructure VMMUrl
VMMUser VMMPwd VMTemplate VMMMax hostCapacity PAConfig RMCredentials
```

where:

- **RMUrl** field holds the Resource Manager's url that will be used to connect remotely, from the virtual machine, by the RM Nodes to register themselves as available. If you do not fill this field, local node registration will be used.

- **infrastructure** - Kind of virtualized infrastructure to manage. For the time being, only [vmware](#), [xenserver](#) and [virtualbox](#) are supported.
- **VMMUrl** - Url of the Hypervisor.
- **VMMUser** - Hypervisor's user name.
- **VMMPwd** - Hypervisor's password name.
- **VMMTemplate** - Template virtual machine.
- **VMMMax** - The maximum number of virtual machine.
- **hostCapacity** - The number of nodes per virtual machine.
- **PAConfig** - File describing ProActive properties used to update RM Nodes environment. You can thus dynamically precise the node's environment like a precise network communication protocol.
- **RMCredentials** - Encrypted credentials file, as created by the create-cred[.bat] utility. These credentials will be used by the nodes to authenticate on the Resource Manager.

4.3.11.2.1. VMware Products

This section deals with VMware products ([Virtual Infrastructure compliant](#)¹³). To use this kind of products for Resource Manager Node acquisition, you must first create your virtual machine, then, copy your Resource Manager distribution inside the virtual machine. Next, use one of the scripts that we provide (`scripts/virtualization/rm-node-starter.[sh|bat]`) to register a new service on your virtual machine. Note that for both case, you have to get python installed and available in the *PATH* variable, and especially for windows, you must use the [AutoEXnt software](#)¹⁴. You also need to install VMware Guest Tools (from vi web management interface, right panel, install guest tools), and have them available in the *PATH* variable. You must modify the `rm-node-starter` scripts in the virtual machine to match your current Resource Manager setup.

Once you have your virtual machine template correctly installed, you can use it from the Resource Manager as a RM Node pool. The **Infrastructure** field appearing on the snapshot above must contain the value "vmware". The **VMM Url** field holds the address of your VMware virtual machine monitor. For VMware Virtual Infrastructure compliant products, most often, it is either `http://yourAddress:8222/sdk` or `https://yourAddress:8333/sdk` by default. If you decide to provide a **VM Max** number greater than 1, the Resource Manager is allowed to clone the template virtual machine for provisioning. The clone feature for VMware product is provided by the product itself. The clone virtual machine shares the entire set of device of the cloned virtual machine. The templates virtual disk(s) is(are) attached to the clone in independent-nonpersistent mode, that means that every power off on the clone will erase every data modification made since last startup. Note that during the clone up time, the template virtual machine cannot be used anymore as the virtual disk(s) is(are) locked by the clone. When the clone virtual machines are not used anymore, they are destroyed. If a problem occurs during destruction time, the best way to come back to a clean environment state is to detach every virtual disk from the clone virtual machines (not from the template) and to remove every clone (check the "Delete this virtual machine's files from the disk" when confirmation is asked. This way, all disks but the template one will be removed).

4.3.11.2.2. XenServer

This section deals with Citrix's XenServer product. To use this kind of products for Resource Manager Node acquisition, you must first create your virtual machine, then, copy your Resource Manager distribution inside the virtual machine. Next, use one of the scripts that we provide (`scripts/virtualization/rm-node-starter.[sh|bat]`) to register a new service on your virtual machine. Note that for both cases, you have to get python installed and available in the *PATH* variable, and especially for windows, you must use the [AutoEXnt software](#)¹⁵. You must modify the `rm-node-starter` scripts in the virtual machine to match your current Resource Manager setup.

Because Citrix does not provide any equivalent of VMware guest tools, we use the embeded mysql server to hold deployment information. To be able to access these data from the inside of the virtual machine, we use the python XenAPI.py lib to connect to the running XenServer and get the pieces of information. They are stored in the XenServer DataStore at the VirtualMachine object level as a Hashtable. All the data holded by this table are retrieved thanks to one of the virtual machine MAC address (XenServer ensures that every allocated MAC address is different). Thus, to be sure that the Node acquisition will be effective, you must ensure the uniqueness of MAC address for your entire virtual infrastructure. **You also need to fill** the associated server information located in the

¹³ <http://www.vmware.com/products/vi/>

¹⁴ <http://www.microsoft.com/Downloads/details.aspx?FamilyID=9d467a69-57ff-4ae7-96ee-b18c4790cffd&displaylang=en>

¹⁵ <http://www.microsoft.com/Downloads/details.aspx?FamilyID=9d467a69-57ff-4ae7-96ee-b18c4790cffd&displaylang=en>

file `scripts/virtualization/xenserver/xenserver.py` in the virtual machine, **xenServerAddress**: the url of the XenServer running your virtual machine, **xenServerUserID**: the username you use to connect to your XenServer and his password: **xenServerUserPWD**.

Once you have your virtual machine template correctly installed, you can use it from the Resource Manager as a RM Node pool. The **Infrastructure** field appearing on the snapshot above must contain the value "xenserver". The **VMM Url** field holds the address of your XenServer. By default this is the http url of the server. If you decide to provide a **VM Max** number greater than 1, the Resource Manager is allowed to clone the template virtual machine for provisioning. The clone feature is provided by the virtual machine monitor. The way XenServer clone a virtual machine relies on the underlying disk store file system type. If you want to enable efficient clone feature (using differential disks) you must ensure that the virtual disks used by your template virtual machine are stored on a EXT3 base store (the default store is LVM base). To change this characteristic refer to this [good tutorial](#)¹⁶.

4.3.11.2.3. xVM Virtualbox

This section deals with Sun's xVM Virtualbox product. To use this kind of products for Resource Manager Node acquisition, you must first create your virtual machine, then, copy your Resource Manager distribution inside the virtual machine. Next, use one of the scripts that we provide (`scripts/virtualization/rm-node-starter.[sh|bat]`) to register a new service on your virtual machine. Note that for both case, you have to get python installed and available in the *PATH variable*, and especially for windows, you must use the [AutoEXnt software](#)¹⁷. You also need to install Virtualbox Guest Tools (from Virtual Machine window devices, install guest additions), and have them available in the PATH variable. You also must modify the `rm-node-starter` scripts in the virtual machine to match your current Resource Manager setup.

Once you have your virtual machine template correctly installed, you can use it from the Resource Manager as a RM Node pool. The **Infrastructure** field appearing on the snapshot above must contain the value "virtualbox". The **VMM Url** field holds the address of the `vboxwebsrv` program shipped with xVM Virtualbox. We recommend to run it with a **-t** parameter set to **20**. Note that the default value for the **-p** parameter is **18083**, that means that the default VMM Url would be `http://vboxwebsrvAddress:18083`. We also notices that some `vboxwebsrv` releases had some troubles with authentication. If it is your case, just submit the following command:

```
VBoxManage setproperty websrvauthlibrary null
```

and restart `vboxwebsrv`. If you decide to provide a **VM Max** number greater than 1, the Resource Manager is allowed to clone the template virtual machine for provisioning. The clone feature is not provided by xVM Virtualbox itself. To emulate it, you must snapshot the template virtual machine. The snapshot for Virtualbox is in fact the creation of a new differential disk hierarchy. That means that every newly created virtual machine attached to this snapshot will create a new differential disk. When the clone virtual machines are not used anymore, they are destroyed. If a problem occurs during destruction time, the best way to come back to a clean environment state is to remove every clone virtual machine and from the virtual disk management window, remove every virtual disk belonging to the snapshot hierarchy of the template virtual machine that is not attached to a virtual machine.

4.3.11.2.4. Microsoft Hyper-V

This section deals with Microsoft's Hyper-V product. To use this kind of products for Resource Manager Node acquisition, you must first create your virtual machine, then, copy your Resource Manager distribution inside the virtual machine. Next, register one of the scripts that we provide (`scripts/virtualization/rm-node-starter.[sh|bat]`) as a new service on your virtual machine. Note that for both case, you have to get python installed and available in the *PATH variable*, and especially for windows, you must use the [AutoEXnt software](#)¹⁸. You also need to install Hyper-V guest drivers (if not available, fill `scripts/virtualization/hyperv/hyperv.py` properties to be able to connect to the Hyper-V host). You also must modify the `rm-node-starter` scripts in the virtual machine to match your current Resource Manager setup.

Once you have your virtual machine template correctly installed, you can use it from the Resource Manager as a RM Node pool. The **Infrastructure** field appearing on the snapshot above must contain the value "hyperv-wmi" or "hyperv-winrm" depending on what kind of management tool you want to use. See one of the following instructions to set the appropriated tool:

- **WMI**: it is based on DCOM, so you have to allow DCOM communication through your firewall
- **WinRM**: uses http, you must set up the host configuration like explained in [this tutorial](#)¹⁹

¹⁶ http://www.tokeshi.com/index.php?option=com_content&task=view&id=5025

¹⁷ <http://www.microsoft.com/Downloads/details.aspx?FamilyID=9d467a69-57ff-4ae7-96ee-b18c4790cffd&displaylang=en>

¹⁸ <http://www.microsoft.com/Downloads/details.aspx?FamilyID=9d467a69-57ff-4ae7-96ee-b18c4790cffd&displaylang=en>

¹⁹ <http://blogs.dirteam.com/blogs/sanderberkouwer/archive/2008/02/23/remotely-managing-your-server-core-using-winrm-and-winsr.aspx>

The **VMM Url** field holds the address of the Hyper-V host and depends on the protocol you want to use. For WinRM, use a url like: **http(s)://your.host.com/wsman**, for WMI, use the FQDN of your server. If you decide to provide a **VM Max** number greater than 1, the Resource Manager is allowed to clone the template virtual machine for provisioning. The clone feature is not provided by Hyper-V itself. To emulate it, we create a new virtual machine with a differencing hard disk attached to the template virtual machine's hard disk. When the clone virtual machines are not used anymore, they are destroyed, and so do the attached differencing hard disks. If a problem occurs during destruction time, the best way to come back to a clean environment state is to remove every clone virtual machine by hand with their associated differencing hard disks. Note that when you clone a virtual machine, you cannot use the template virtual machine anymore without jeopardizing the integrity of the clone virtual machine.

4.3.12. Static policy

Static node source policy starts node acquisition when nodes are added to the node source and never removes them. Nevertheless, nodes can be removed by user request.

For using this policy, you have to precise the following parameters:

- **nodeUsers** - utilization permission defined who can get nodes for computations from this node source. It has to take one of the following values:
 - "ME" - Only the node source creator
 - "MY_GROUPS" - Only groups in which the node source creator belongs to
 - "PROVIDER" - Only the node provider who added the node. Provider can use only its nodes
 - "PROVIDER_GROUPS" - Only groups in which the node provider belongs to. Users from this groups can user only nodes of this provider
 - "ALL" - Everybody
- **nodeProviders** - Provider permission defines who can add nodes to this node source. It should take one of the following values:
 - "ME" - Only the node source creator
 - "MY_GROUPS" - Only the group in which the node source creator belongs to
 - "ALL" - Everybody

4.3.13. Time slot policy

Time slot policy is aimed to acquire nodes for particular time with an ability to do it periodically.

For using this policy, you have to precise the following parameters:

- **nodeUsers** - utilization permission defined who can get nodes for computations from this node source. It has to take one of the following values:
 - "ME" - Only the node source creator
 - "MY_GROUPS" - Only groups in which the node source creator belongs to
 - "PROVIDER" - Only the node provider who added the node. Provider can use only its nodes
 - "PROVIDER_GROUPS" - Only groups in which the node provider belongs to. Users from this groups can user only nodes of this provider
 - "ALL" - Everybody
- **nodeProviders** - Provider permission defines who can add nodes to this node source. It should take one of the following values:
 - "ME" - Only the node source creator
 - "MY_GROUPS" - Only the group in which the node source creator belongs to
 - "ALL" - Everybody
- **acquireTime** - Absolute acquire date (e.g. "6/3/10 1:18:45 PM CEST").
- **releaseTime** - Absolute releasing date (e.g. "6/3/10 2:18:45 PM CEST").
- **period** - period time in millisecond (default is 86400000).
- **preemptive** - Preemptive parameter indicates the way of releasing nodes. If it is true, nodes will be released without waiting the end of jobs running on (default is false).

4.3.14. Cron policy

Cron policy is aimed to acquire and remove nodes at specific time defined in the cron syntax.

For using this policy, you have to precise the following parameters:

- **nodeUsers** - utilization permission defined who can get nodes for computations from this node source. It has to take one of the following values:
 - "ME" - Only the node source creator
 - "MY_GROUPS" - Only groups in which the node source creator belongs to
 - "PROVIDER" - Only the node provider who added the node. Provider can use only its nodes
 - "PROVIDER_GROUPS" - Only groups in which the node provider belongs to. Users from this groups can user only nodes of this provider
 - "ALL" - Everybody
- **nodeProviders** - Provider permission defines who can add nodes to this node source. It should take one of the following values:
 - "ME" - Only the node source creator
 - "MY_GROUPS" - Only the group in which the node source creator belongs to
 - "ALL" - Everybody
- **nodeAcquisition** - The time policy will trigger the deployment of all nodes (e.g. "0 12 * * *" every day at 12.00).
- **nodeRemoval** - The time policy will trigger the removal of all nodes (e.g. "0 13 * * *" every day at 13.00).
- **preemptive** - Preemptive parameter indicates the way of releasing nodes. If it is true, nodes will be released without waiting the end of jobs running on (default is false).
- **forceDeployment** - If for the example above (the deployment starts every day at 12.00 and the removal starts at 13.00) you are creating the node source at 12.30 the next deployment will take place the next day. If you'd like to force the immediate deployment set this parameter to true.

4.3.15. "Remove nodes when scheduler is idle" policy

"Remove nodes when scheduler is idle" policy removes all nodes from the infrastructure when the scheduler is idle and acquires them when a new job is submitted. This policy may be useful if there is no need to keep nodes alive permanently. Nodes will be released after a specified "idle time". This policy will use a listener of the scheduler, that is why its URL, its user name and its password have to be specified.

For using this policy, you have to precise the following parameters:

- **nodeUsers** - utilization permission defined who can get nodes for computations from this node source. It has to take one of the following values:
 - "ME" - Only the node source creator
 - "MY_GROUPS" - Only groups in which the node source creator belongs to
 - "PROVIDER" - Only the node provider who added the node. Provider can use only its nodes
 - "PROVIDER_GROUPS" - Only groups in which the node provider belongs to. Users from this groups can user only nodes of this provider
 - "ALL" - Everybody
- **nodeProviders** - Provider permission defines who can add nodes to this node source. It should take one of the following values:
 - "ME" - Only the node source creator
 - "MY_GROUPS" - Only the group in which the node source creator belongs to
 - "ALL" - Everybody
- **schedulerUrl** - Url of the Scheduler
- **schedulerCredentialsPath** - Path to the credentials used for scheduler authentication.
- **idleTime** - idle time in millisecond to wait before removing all nodes (default is 60000).



Note

This policy is available only when using the **ProActive Scheduler**.

4.3.16. "Scheduler loading" policy

"Scheduler loading" policy acquires/releases nodes according to the scheduler loading factor. This policy allows to configure the number of resources which will be always enough for the scheduler. Nodes are acquired and released according to scheduler loading factor which is a number of tasks per node. In the same manner as the previous policy, this one also requires scheduler URL, user name and password. It is important to correctly configure maximum and minimum nodes that this policy will try to hold. Maximum number should not be greater than potential nodes number which is possible to deploy to underlying infrastructure. If there are more currently acquired nodes than necessary, policy will release them one by one after having waited for a "release period" delay. This smooth release procedure is implemented because deployment time is greater than the release one. Thus, this waiting time deters policy from spending all its time trying to deploy nodes.

For using this policy, you have to precise the following parameters:

- **nodeUsers** - utilization permission defined who can get nodes for computations from this node source. It has to take one of the following values:
 - **"ME"** - Only the node source creator
 - **"MY_GROUPS"** - Only groups in which the node source creator belongs to
 - **"PROVIDER"** - Only the node provider who added the node. Provider can use only its nodes
 - **"PROVIDER_GROUPS"** - Only groups in which the node provider belongs to. Users from this groups can user only nodes of this provider
 - **"ALL"** - Everybody
- **nodeProviders** - Provider permission defines who can add nodes to this node source. It should take one of the following values:
 - **"ME"** - Only the node source creator
 - **"MY_GROUPS"** - Only the group in which the node source creator belongs to
 - **"ALL"** - Everybody
- **schedulerUrl** - Url of the Scheduler
- **schedulerCredentialsPath** - Path to the credentials used for scheduler authentication.
- **refreshTime** - time between each calculation of the number of needed nodes.
- **minNodes** - Minimum number of nodes to deploy
- **maxNodes** - Maximum number of nodes to deploy
- **loadFactor** - number of tasks per node. Actually, if this number is N , it does not means that there will be exactly N tasks executed on each node. This factor is just used to compute the total number of nodes. For instance, let us assume that this factor is 3 and that we schedule 100 tasks. In that case, we will have 34 (= upper bound of $100/3$) started nodes. Once one task finished and the refresh time passed, one node will be removed since 99 divided by 3 is 33. When there will remain 96 tasks (asuming that no other tasks are scheduled meanwhile), an other node will be removed at the next calculation time, and so on and so forth...
- **nodeDeploymentTimeout** - The node deployment timeout.



Note

This policy is available only when using the **ProActive Scheduler**.

4.3.17. "Cron load based" policy

The **"Cron load based" policy** triggers new nodes acquisition when scheduler is overloaded (exactly like with **"Scheduler loading" policy**) only within a time slot defined using crontab syntax. All other time the nodes are removed from the resource manager.

For using this policy, you have to precise the following parameters:

- **nodeUsers** - utilization permission defined who can get nodes for computations from this node source. It has to take one of the following values:
 - "ME" - Only the node source creator
 - "MY_GROUPS" - Only groups in which the node source creator belongs to
 - "PROVIDER" - Only the node provider who added the node. Provider can use only its nodes
 - "PROVIDER_GROUPS" - Only groups in which the node provider belongs to. Users from this groups can user only nodes of this provider
 - "ALL" - Everybody
- **nodeProviders** - Provider permission defines who can add nodes to this node source. It should take one of the following values:
 - "ME" - Only the node source creator
 - "MY_GROUPS" - Only the group in which the node source creator belongs to
 - "ALL" - Everybody
- **schedulerUrl** - Url of the Scheduler
- **schedulerCredentialsPath** - Path to the credentials used for scheduler authentication.
- **refreshTime** - time between each calculation of the number of needed nodes.
- **minNodes** - Minimum number of nodes to deploy
- **maxNodes** - Maximum number of nodes to deploy
- **loadFactor** - number of tasks per node. Actually, if this number is N , it does not means that there will be exactly N tasks executed on each node. This factor is just used to compute the total number of nodes. For instance, let us assume that this factor is 3 and that we schedule 100 tasks. In that case, we will have 34 (= upper bound of $100/3$) started nodes. Once one task finished and the refresh time passed, one node will be removed since 99 divided by 3 is 33. When there will remain 96 tasks (asuming that no other tasks are scheduled meanwhile), an other node will be removed at the next calculation time, and so on and so forth...
- **nodeDeploymentTimeout** - The node deployment timeout.
- **acquisitionAllowed** - The time when the policy starts to work as the "**scheduler loading**" policy (e.g. "0 12 * * *" every day at 12.00).
- **acquisitionForbidden** - The time policy removes all the nodes from the resource manager (e.g. "0 13 * * *" every day at 13.00).
- **preemptive** - Preemptive parameter indicates the way of releasing nodes. If it is true, nodes will be released without waiting the end of jobs running on (default is false).
- **allowed** - If true acquisition will be immediately allowed.



Note

This policy is available only when using the **ProActive Scheduler**.

4.3.18. "Cron slot load based" policy

The "**Cron slot load based**" policy triggers new nodes acquisition when scheduler is overloaded (exactly like with "**Scheduler loading**" policy) only within a time slot defined using crontab syntax. The other time it holds all the available nodes.

For using this policy, you have to precise the following parameters:

- **nodeUsers** - utilization permission defined who can get nodes for computations from this node source. It has to take one of the following values:
 - "ME" - Only the node source creator
 - "MY_GROUPS" - Only groups in which the node source creator belongs to
 - "PROVIDER" - Only the node provider who added the node. Provider can use only its nodes
 - "PROVIDER_GROUPS" - Only groups in which the node provider belongs to. Users from this groups can user only nodes of this provider
 - "ALL" - Everybody
- **nodeProviders** - Provider permission defines who can add nodes to this node source. It should take one of the following values:

- **"ME"** - Only the node source creator
- **"MY_GROUPS"** - Only the group in which the node source creator belongs to
- **"ALL"** - Everybody
- **schedulerUrl** - Url of the Scheduler
- **schedulerCredentialsPath** - Path to the credentials used for scheduler authentication.
- **refreshTime** - time between each calculation of the number of needed nodes.
- **minNodes** - Minimum number of nodes to deploy
- **maxNodes** - Maximum number of nodes to deploy
- **loadFactor** - number of tasks per node. Actually, if this number is N , it does not mean that there will be exactly N tasks executed on each node. This factor is just used to compute the total number of nodes. For instance, let us assume that this factor is 3 and that we schedule 100 tasks. In that case, we will have 34 (= upper bound of $100/3$) started nodes. Once one task finished and the refresh time passed, one node will be removed since 99 divided by 3 is 33. When there will remain 96 tasks (assuming that no other tasks are scheduled meanwhile), another node will be removed at the next calculation time, and so on and so forth...
- **nodeDeploymentTimeout** - The node deployment timeout.
- **acquisitionAllowed** - The time when the policy starts to work as the **"scheduler loading" policy** (e.g. "0 12 * * *" every day at 12.00).
- **acquisitionForbidden** - The time policy removes all the nodes from the resource manager (e.g. "0 13 * * *" every day at 13.00).
- **preemptive** - Preemptive parameter indicates the way of releasing nodes. If it is true, nodes will be released without waiting the end of jobs running on (default is false).
- **allowed** - If true acquisition will be immediately allowed.



Note

This policy is available only when using the **ProActive Scheduler**.

4.3.19. Amazon EC2 policy

Allocates resources according to the Scheduler loading factor, releases resources considering that EC2 instances are paid by the hour.

For using this policy, you have to precise the following parameters:

- **nodeUsers** - utilization permission defined who can get nodes for computations from this node source. It has to take one of the following values:
 - **"ME"** - Only the node source creator
 - **"MY_GROUPS"** - Only groups in which the node source creator belongs to
 - **"PROVIDER"** - Only the node provider who added the node. Provider can use only its nodes
 - **"PROVIDER_GROUPS"** - Only groups in which the node provider belongs to. Users from this groups can use only nodes of this provider
 - **"ALL"** - Everybody
- **nodeProviders** - Provider permission defines who can add nodes to this node source. It should take one of the following values:
 - **"ME"** - Only the node source creator
 - **"MY_GROUPS"** - Only the group in which the node source creator belongs to
 - **"ALL"** - Everybody
- **schedulerUrl** - Url of the Scheduler
- **schedulerCredentialsPath** - Path to the credentials used for scheduler authentication.
- **preemptive** - Preemptive parameter indicates the way of releasing nodes. If it is true, nodes will be released without waiting the end of jobs running on (default is false).
- **refreshTime** - time between each calculation of the number of needed nodes.
- **loadFactor** - number of tasks per node. Actually, if this number is N , it does not mean that there will be exactly N tasks executed on each node. This factor is just used to compute the total number of nodes. For instance, let us assume that this factor is 3 and that

we schedule 100 tasks. In that case, we will have 34 (= upper bound of 100/3) started nodes. Once one task finished and the refresh time passed, one node will be removed since 99 divided by 3 is 33. When there will remain 96 tasks (assuming that no other tasks are scheduled meanwhile), another node will be removed at the next calculation time, and so on and so forth...

- **releaseDelay** - Delay between each node release. This time is useful since the deploying time is important. Let us assume that a node has to be removed. If this releaseDelay did not exist (or if it was set to 0), this node would be removed instantaneously. Let us assume that right after this removal, another task is scheduled, requiring a new node. In that case, we would lose a lot of time removing the previous node and deploying another one whereas the task could have been scheduled on the same node. This releaseDelay therefore represents the time to wait before effectively removing a node.



Note

This policy is available only when using the **ProActive Scheduler**.

4.3.20. Custom infrastructure/policy

The resource manager provides the way to create your own custom policy or infrastructure. In order to do it:

- First implement your policy extending from **org.ow2.proactive.resourcemanager.nodesource.policy.NodeSourcePolicy** or infrastructure extending from **org.ow2.proactive.resourcemanager.nodesource.infrastructure.InfrastructureManager**. To add the parameter which will be visible in GUI/CLI just put **@Configurable** annotation to the corresponding field.
- Second put your classes into **\$RMHOME/addons** directory so that the resource manager will find them.
- And finally update the configuration file. For newly created policy put the name into **\$RMHOME/config/rm/nodesource/policies**. For the infrastructure do the same in **\$RMHOME/config/rm/nodesource/infrastructures**.

When you start the resource manager your infrastructure/policy will be in the list of supported ones.

4.4. Administration with Java API

The ProActive Resource Manager can be used directly from the Java code through its Java API. This chapter explains how to do that, namely how to start and stop the resource manager, add existing nodes and deploy new nodes with GCM deployment descriptors.

4.4.1. Start and stop the resource manager

Resource Manager can be started locally using a static method of RMFactory class:

```
// Creates initializer for the resource manager.
// The 6 following lines are mandatory for starting RM in a clean JVM
// But each variable can be set as JVM argument (i.e. -Dvar=value) instead of as API.
RMInitializer init = new RMInitializer();

// PResourceManagerProperties.RM_HOME is empty. You have to start your
// application with -Dpa.rm.home=<YOUR_RM_HOME_DIR>
System.out.println("RM home directory = " + PResourceManagerProperties.RM_HOME);
init.setRMHomePath(PResourceManagerProperties.RM_HOME.toString());
init.setLog4jConfiguration(PResourceManagerProperties.RM_HOME + "config/log4j/rm-log4j-server");
init
    .setJavaSecurityPolicy(PResourceManagerProperties.RM_HOME +
        "config/security.java.policy-server");
init.setProActiveConfiguration(PResourceManagerProperties.RM_HOME +
    "config/proactive/ProActiveConfiguration.xml");
init.setResourceManagerPropertiesConfiguration(PResourceManagerProperties.RM_HOME +
    "config/authentication/login.cfg");

// Starts an empty RM...
System.out.println("Starting RM, please wait...");
```

```

RMAuthentication auth = null;
try {
    auth = RMFactory.startLocal(init);
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

// Retrieves connection URL for RM: auth.getHostURL()
String rmUrl = auth.getHostURL();
System.out.println("RM successfully started at : " + rmUrl);

```

In order to perform any operations with the Resource Manager, you have to authorize and acquire Resource Manager interface. This action is the same as the one we saw in [Section 3.1, “Connecting to the resource manager”](#). Refer to that section to know how to get a reference towards the resource manager in order to be able to perform actions authorized by your permissions. For example, as an administrator, you are able to shutdown the resource manager:

```
resourceManager.shutdown(true);
```

This method terminates all RM's active objects and stops JVM where RMCore is running on. The boolean parameter that it takes in argument defines whether the Resource Manager stops immediately, even if there are some busy nodes (i.e. some nodes have been provided to an application) or waits until applications give them back.

4.4.2. Register an existing node

Resource Manager provides a way to add ProActive nodes that are already deployed. Nodes are added using their URLs.

```

String nodeUrl = "rmi://kisscool.inria.fr:1099/MyNode";
BooleanWrapper isAdded = resourceManager.addNode(nodeUrl);

```



Note

The snippet here above has the same behavior as:

```
BooleanWrapper isAdded = resourceManager.addNode(nodeUrl, "Default");
```

The second parameter stands for the name of the nodesource to which one the node is added.

To start a node, you can use the `rm-start-node` script:

```
$RM_HOME/bin/[os]> rm-start-node -n MyNode
```

The script prints the url of the node that you must provide to the `addNode` method. You can also directly make the node register itself to the resource manager. In this case you need to provide the appropriate url of the resource manager and the credentials you want to use either specifying a file, an environment variable or the value:

```
$RM_HOME/bin/[os]> rm-start-node -n MyNode -r rmi://kisscool.inria.fr:1099/ -f $RM_HOME/config/
authentication/rm.cred
```

In the example above, `-r rmi://kisscool.inria.fr:1099/` stands for the url of the resource manager and `-f $RM_HOME/config/authentication/rm.cred` is the credentials file (use `-s` to provide the name of a nodesource).

4.4.3. Remove a node from the resource manager

You can remove a node from Resource Manager with the code:

```
BooleanWrapper isRemoved = resourceManager.removeNode(nodeUrl, true);
```

The first parameter is the URL of the node to remove and the second one is a boolean. If this boolean is set to true, the resource manager removes immediately the node even if it is busy. If this boolean is set to false and the node is busy, the resource manager waits until the user using the node releases it. Once the node is released the RM removes it.

4.4.4. Create a node source

You can create different node sources using the Resource Manager interface. Let us take a look at the following example to know how to create one of them:

```
// creating infrastructure manager parameters
String javaPath = System.getProperty("java.home") + "/bin/java";
String javaOptions = "";
Object[] infrastructureParameters = new Object[] { javaPath,
    PAResourceManagerProperties.RM_HOME.toString(), "rmi", "1099", javaOptions,
    "kisscool.inria.fr".getBytes() };

Object[] policyParameters = new Object[] { "MY_GROUPS", "ME" };

resourceManager.createNodeSource("MySshNodeSource", SSHInfrastructure.class.getName(),
    infrastructureParameters, StaticPolicy.class.getName(), policyParameters);

while (resourceManager.getState().getTotalNodesNumber() != 1) {
    System.out.println("waiting....");
    Thread.sleep(1000);
}
System.out.println("ok");
```

The node source is created with the ssh infrastructure manager and a static policy. The ssh infrastructure takes 6 parameters: the path to the java executable, the Resource Manager home directory, the protocol and the port that will be used to access created nodes, the list of Java options and finally, the list of hosts. For the static policy, 2 parameters has to be provided: the permission for node utilization and the permission for providing nodes. The provider permission specifies users who can add nodes to this node source and is one of the followings: "ME", "MY_GROUPS", "ALL". The utilization permission defines who can obtain nodes for computations from this node source and must be one of the followings: "ME" (node source creator), "MY_GROUPS", "PROVIDER", "PROVIDER_GROUPS", "ALL". In the previous example, only the user who has created the node source, that is "admin", can add nodes to it but every user belonging to his groups can use them.



Note

In order to know the needed parameters for every infrastructure type and policy, you can use the command line interface. Please read [Section 4.2.2, "Interacting with the resource manager"](#)

4.5. Accounting

The users (clients) of the Resource Manager may request and offer nodes for computation. To keep track on how much node time was consumed or contributed by a particular user, the Resource Manager associates a user to an account.

More precisely, the nodes can be manipulated by the following basic operations available to the users:

- The **ADD** operation is a registration of a node into the Resource Manager initiated by a user considered as a node provider.

A node can be added, through the API, as a result of a deployment process, through an agent or manually from the command line interface.

- The **REMOVE** operation is the unregistration of a node from the Resource Manager.

A node can be removed, through the API, by a user or automatically if it is unreachable by the Resource Manager.

- The **GET** operation is a node reservation, for an unknown amount of time, by a user considered as a node owner.

For example, the ProActive Scheduler can be considered as a user that reserves a node for a task computation.

- The **RELEASE** operation on a reserved node by any user.

The following accounting data is gathered by the Resource Manager:

- **The used node time:** The amount of time other users have spent in using the resources of a particular user.

More precisely, for a specific node owner, it is the sum of all time intervals from **GET** to **RELEASE**.

- **The provided node time:** The amount of time a user has offered resources to the Resource Manager.

More precisely, for a specific node provider, it is the sum of all time intervals from **ADD** to **REMOVE**.

- **The provided node count:** The number of provided nodes.

The accounting information can be accessed through a JMX client or through the Resource Manager GUI.

4.6. The JMX interface

The JMX interface for remote management and monitoring provides information about the running ProActive Resource Manager and allows the user to modify its configuration. For more details about JMX concepts, please refer to official documentation about the [JMX architecture](http://java.sun.com/j2se/1.5.0/docs/guide/jmx/overview/architecture.html)²⁰.

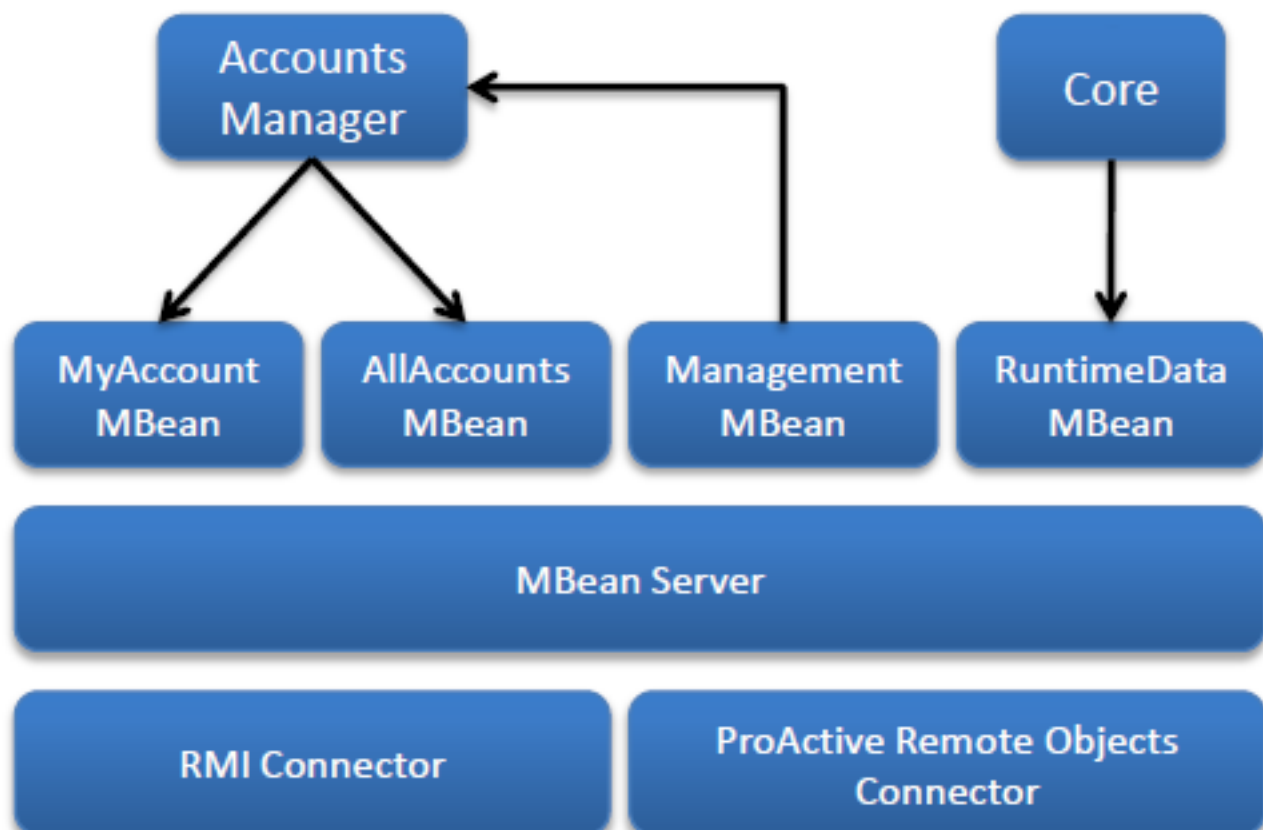


Figure 4.3. Structure of the Resource Manager JMX interface

²⁰ <http://java.sun.com/j2se/1.5.0/docs/guide/jmx/overview/architecture.html>

The following aspects (or services) of the Resource Manager are instrumented using MBeans that are managed through a JMX agent.

- The Resource Manager **Core** exposes various runtime information using the **RuntimeDataMBean** such as:
 - The Resource Manager status
 - Available/Free/Busy/Down nodes count
 - Average activity/inactivity percentage
- The **Accounts Manager** exposes accounting information using the **MyAccountMBean** and **AllAccountsMBean** such as:
 - The used node time
 - The provided node time
 - The provided node count
- Various management operations are exposed using the **ManagementMBean** such as:
 - Setting the accounts refresh rate
 - Refresh all accounts
 - Reload the permission policy file

As shown on [Figure 4.3, “Structure of the Resource Manager JMX interface”](#), the MBean server can be accessed by remote applications using one of the two available connectors:

- The standard solution based on Remote Method Invocation (RMI) protocol is the RMI Connector accessible at the following url:

```
service:jmx:rmi:///jndi/rmi://HOSTNAME:PORT/JMXRMAgent
```

where

- **HOSTNAME** is the hostname on which the RM is started
- **PORT** (5822 by default) is the port number on which the JMX RMI connector server has been started. It is defined by the property **pa.rm.jmx.port**.
- The ProActive Remote Objects Connector provides ProActive protocol aware connector accessible at the following url:

```
service:jmx:ro:///jndi/PA_PROTOCOL://HOSTNAME:PORT/JMXRMAgent
```

where

- **PA_PROTOCOL** is the protocol defined by the **proactive.communication.protocol** property
- **HOSTNAME** is the hostname on which the RM is started
- **PORT** is the protocol dependent port number usually defined by the property **proactive.PA_PROTOCOL.port**

The name of the connector (JMXRMAgent by default) is defined by the property **rm.jmx.connectorname**.

The JMX url on which to connect can be obtained from the Authentication API of the RM or by reading the log file located in `$RM_HOME/logs/RM.log`. In that log file, the address you have to retrieve is the one where the JMX RMI connector server has been started:

```
[INFO 2010-06-17 10:23:27,813] [RM.AbstractJMXHelper.boot] Started JMX RMI connector server at service:jmx:rmi:///jndi/rmi://kisscool.inria.fr:5822/JMXRMAgent
```

Once connected, you'll get an access to RM statistics and accounting.

For example, to connect to the Resource Manager JMX Agent with the popular JConsole tool, just enter the url of the standard RMI Connector as shown on the [Figure 4.4, “Connection using JConsole”](#), as well as the username and the password.

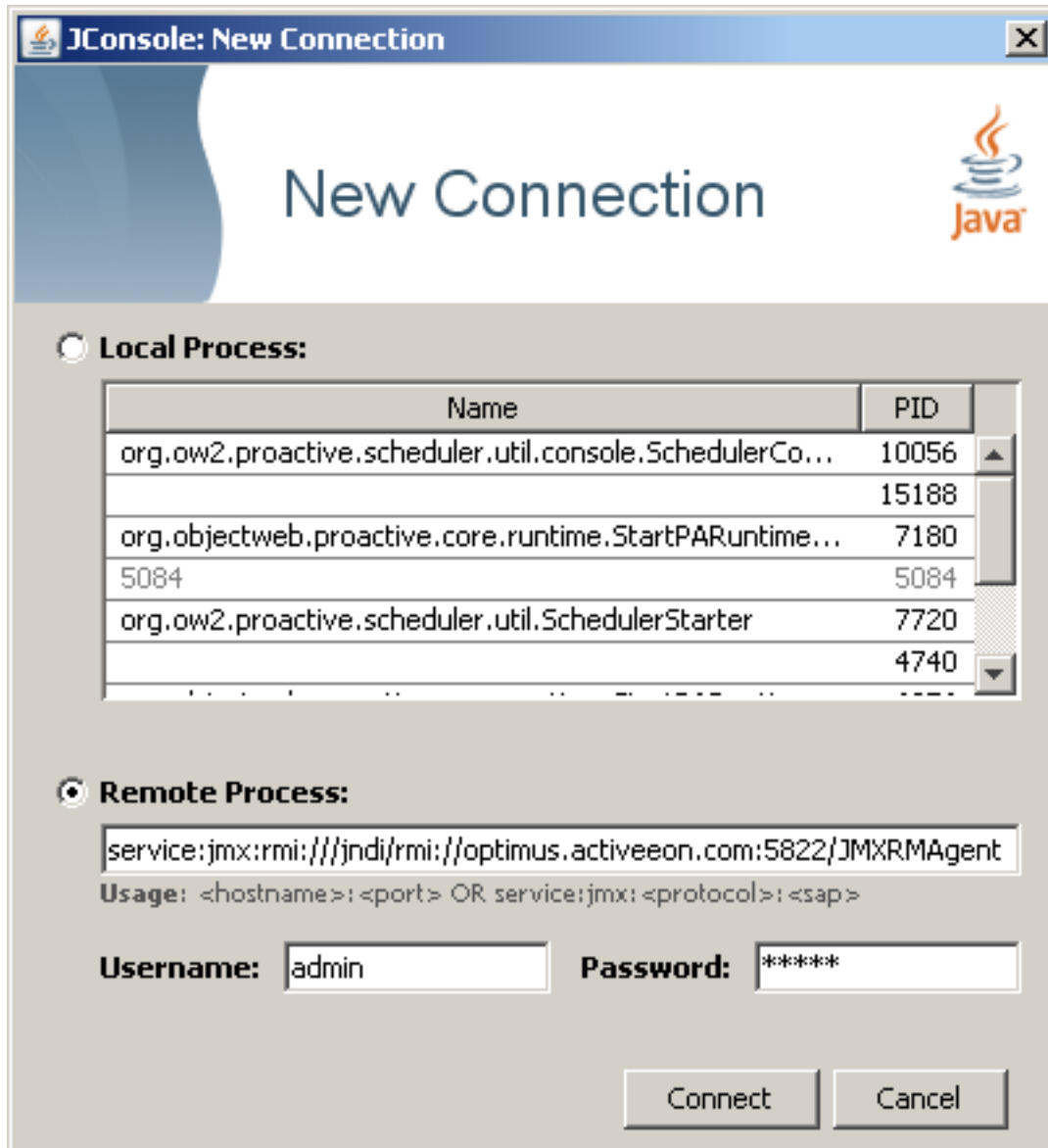


Figure 4.4. Connection using JConsole

Then depending on the allowed permissions browse the attributes of the MBeans.

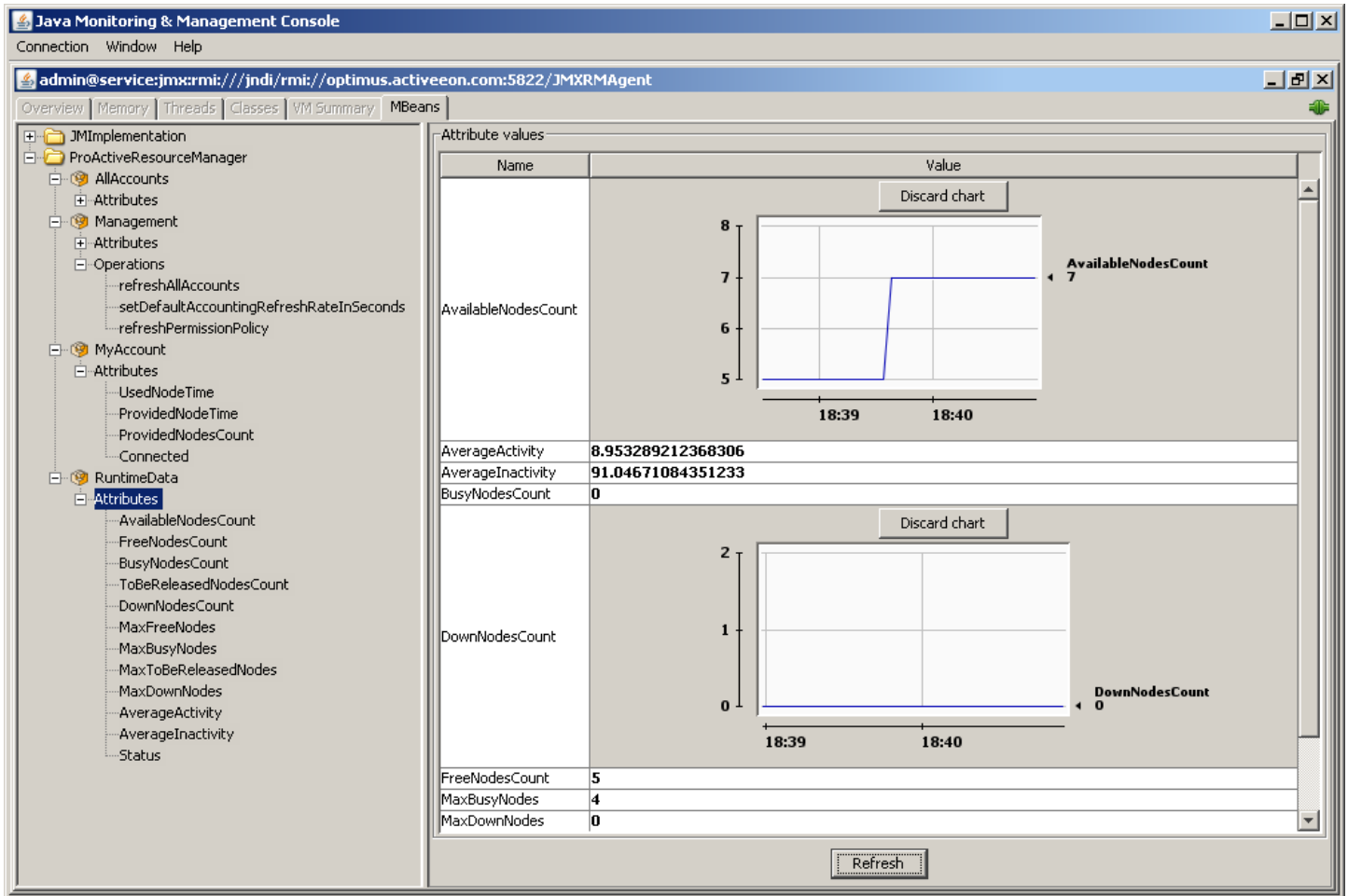


Figure 4.5. Browse MBean attributes

4.7. Resource Manager Graphical User Interface

Resource Manager Eclipse Plugin is a **graphical client** for remote monitoring and control of the ProActive Resource Manager. This stand alone application allows you to perform administrative actions on Resource Manager, deployment and monitoring of computing nodes, creation of node sources and removal of nodes.

The Resource Manager Eclipse Plugin is available in two forms:

- A **Java stand alone application** based on [Eclipse Rich Client Platform \(RCP\)](http://wiki.eclipse.org/index.php/Rich_Client_Platform)²¹, available for any platform (Windows, Linux, Mac OSX, Solaris...).
- A set of **Eclipse**²² **plugins**: with all the functionalities of the stand alone application.

4.7.1. Launching the Resource Manager GUI

Depending on the way you get the RM plugin, just follow one of these steps:

- **If you are using the stand alone application**, just start it using the provided executable. ("ResourceManager" on unix, "ResourceManager.exe" on Windows).

²¹ http://wiki.eclipse.org/index.php/Rich_Client_Platform

²² <http://www.eclipse.org>

- **If you are using the Eclipse plugins set**, install it as an eclipse plugin and restart Eclipse. To do this, copy all the jar files from the `plugins/` directory of your Resource Manager RCP distribution into the `plugins/` directory of your Eclipse distribution and restart Eclipse.

Then, go to **Window->Open Perspective->Other...->Resource Manager** (it could be already opened as it is the default perspective).

Once started, the first screen displayed is the following one:

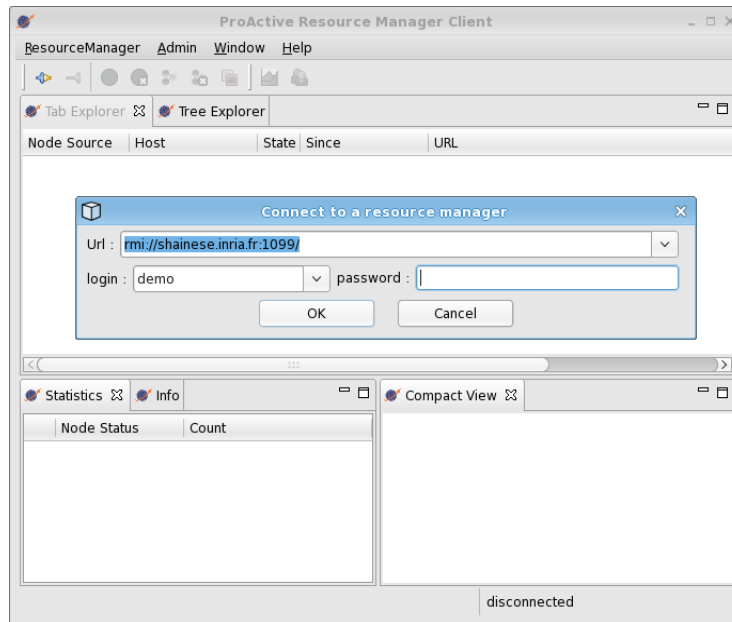


Figure 4.6. Resource Manager startup screen

4.7.2. Connect to an existing Resource Manager

A Resource Manager has to be already started before connecting the Resource Manager GUI. If no RM has been started, please refer to [Section 4.2.1, “Launching the resource manager”](#).

Normally, when starting the GUI, a connection dialog box should appear on top of the main screen. If not, right click on Resource Explorer area and select "Connect" on the pop-up menu. Clicking on this action, the following dialog box will show up:

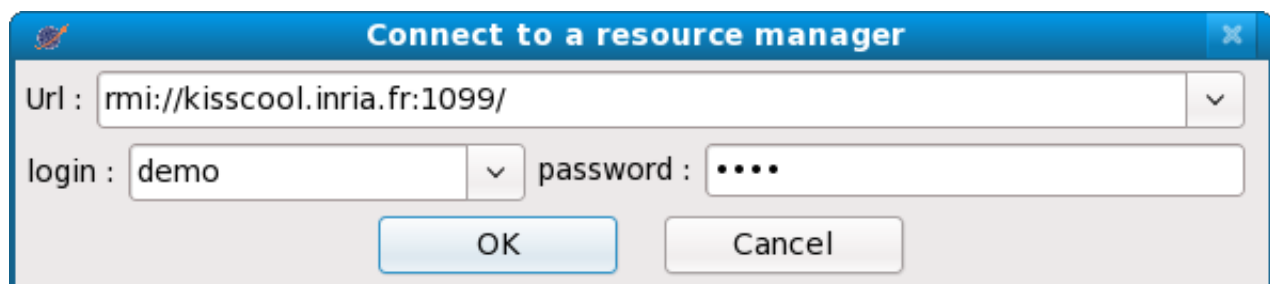


Figure 4.7. Connection screen

Enter the URL of an existing Resource Manager and login information. You can use for example "demo" as a login and "demo" again as a password. This user is an administrator so you will be able to perform any action such as management of nodes and node sources,

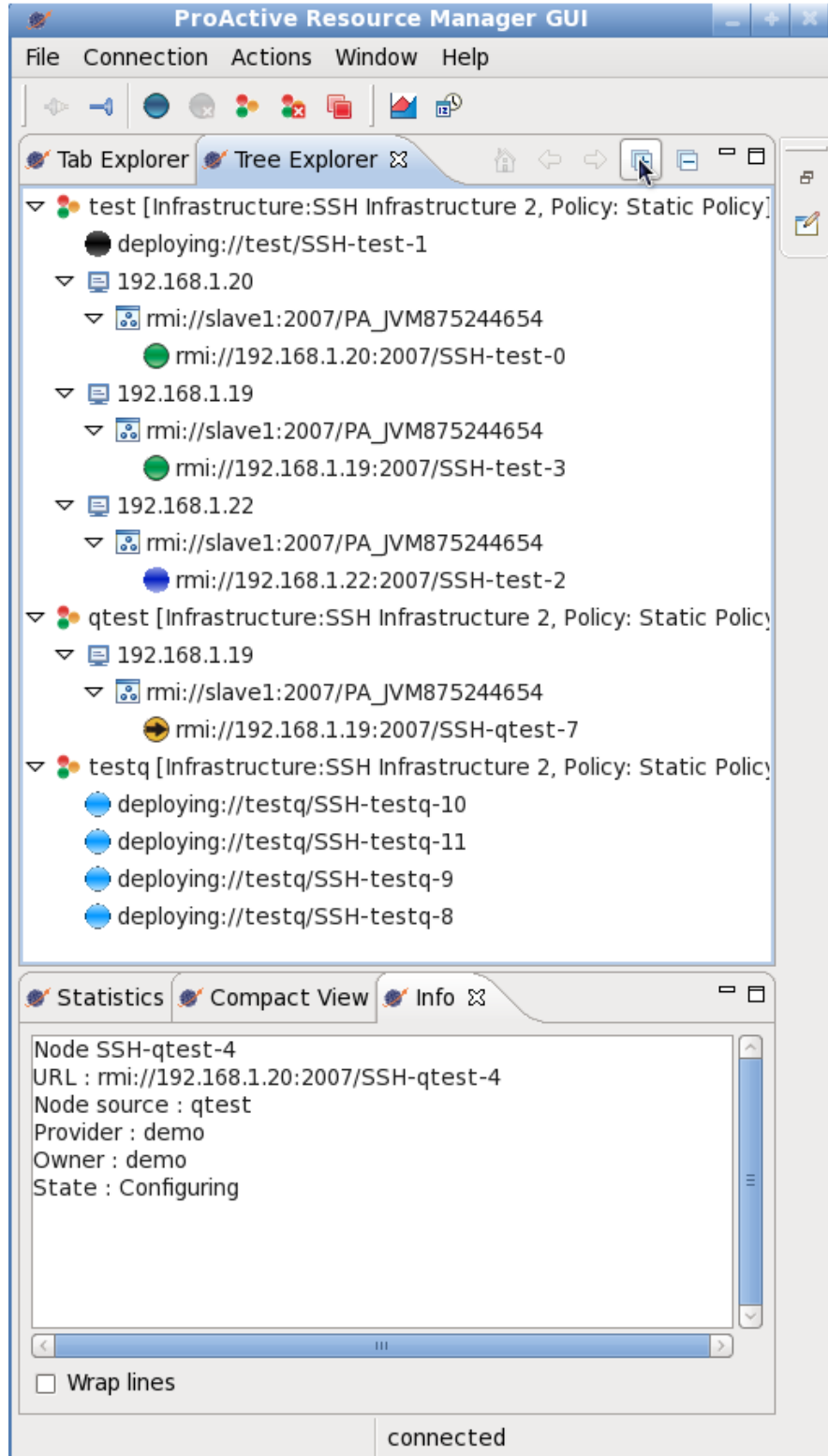
shutdown of resource manager and so on... without any permission problems. Regular users can only get information about resources and their states. The URL is composed of the protocol, the host name on which the RM is started and the port on which it is listening.

You are now connected and ready to perform administrative actions on the remote Resource Manager.

4.7.3. Resource Manager main views

4.7.3.1. Tree view

Once you are connected, you will see all the information about RM resources through this view. This view presents all nodes handled by Resource Manager, their URLs, and their states. It uses a tree representation: tree's roots represents node sources, the second level displays host names and, finally, leafs show nodes deployed on this host. For each node, you can see its URL, and its state represented by an [icon](#).

**Figure 4.8. Tree view**

4.7.3.2. Tab view

The **table view** is view where each line corresponds to one node.

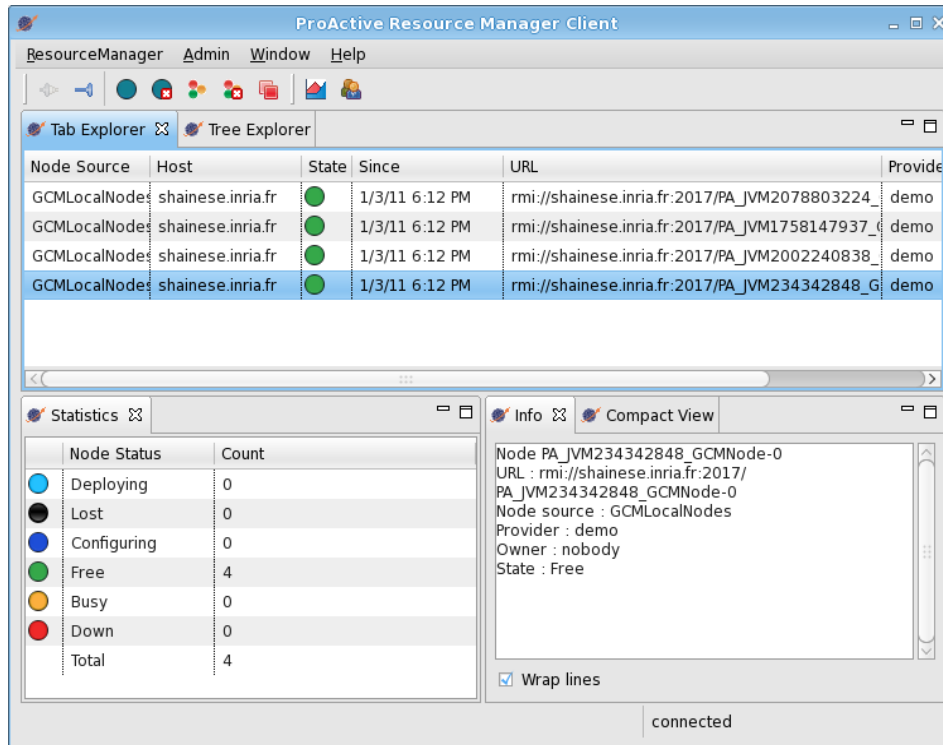


Figure 4.9. Tab view

4.7.3.3. Compact view

The **compact view** is a view where all nodes are displayed in a compact form. It is in particular very useful when you have a great number of nodes to monitor and you want a global view.

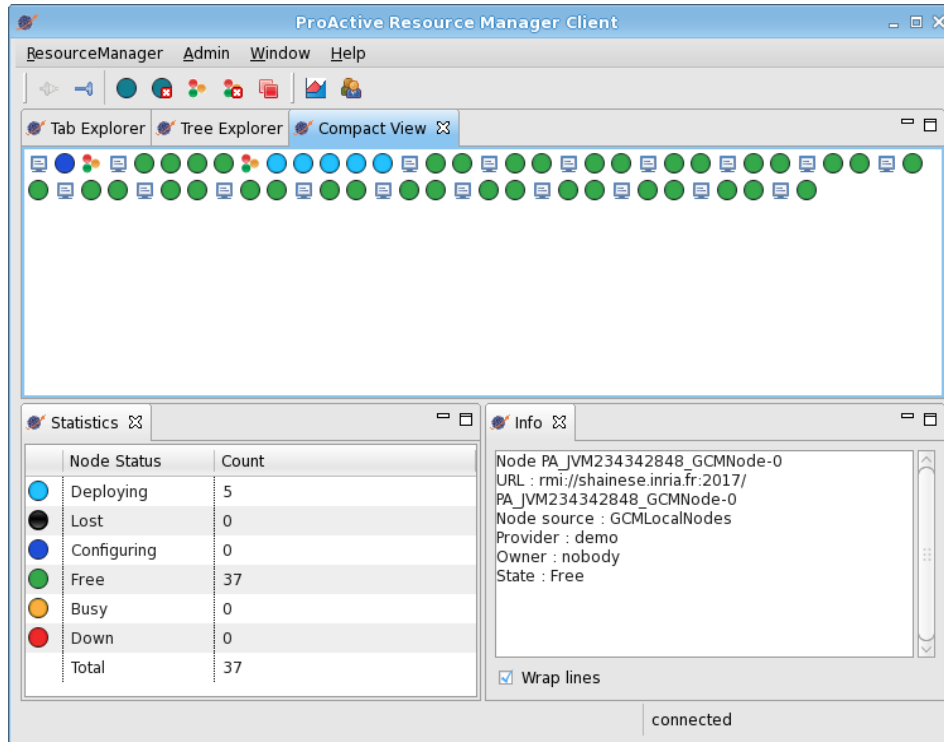


Figure 4.10. Compact view

4.7.3.4. Topology view

The **topology view** visualizes nodes according to network latencies between them. This view allows to see groups of nodes close to each other, latencies between all the nodes and of course the information about free/busy nodes.

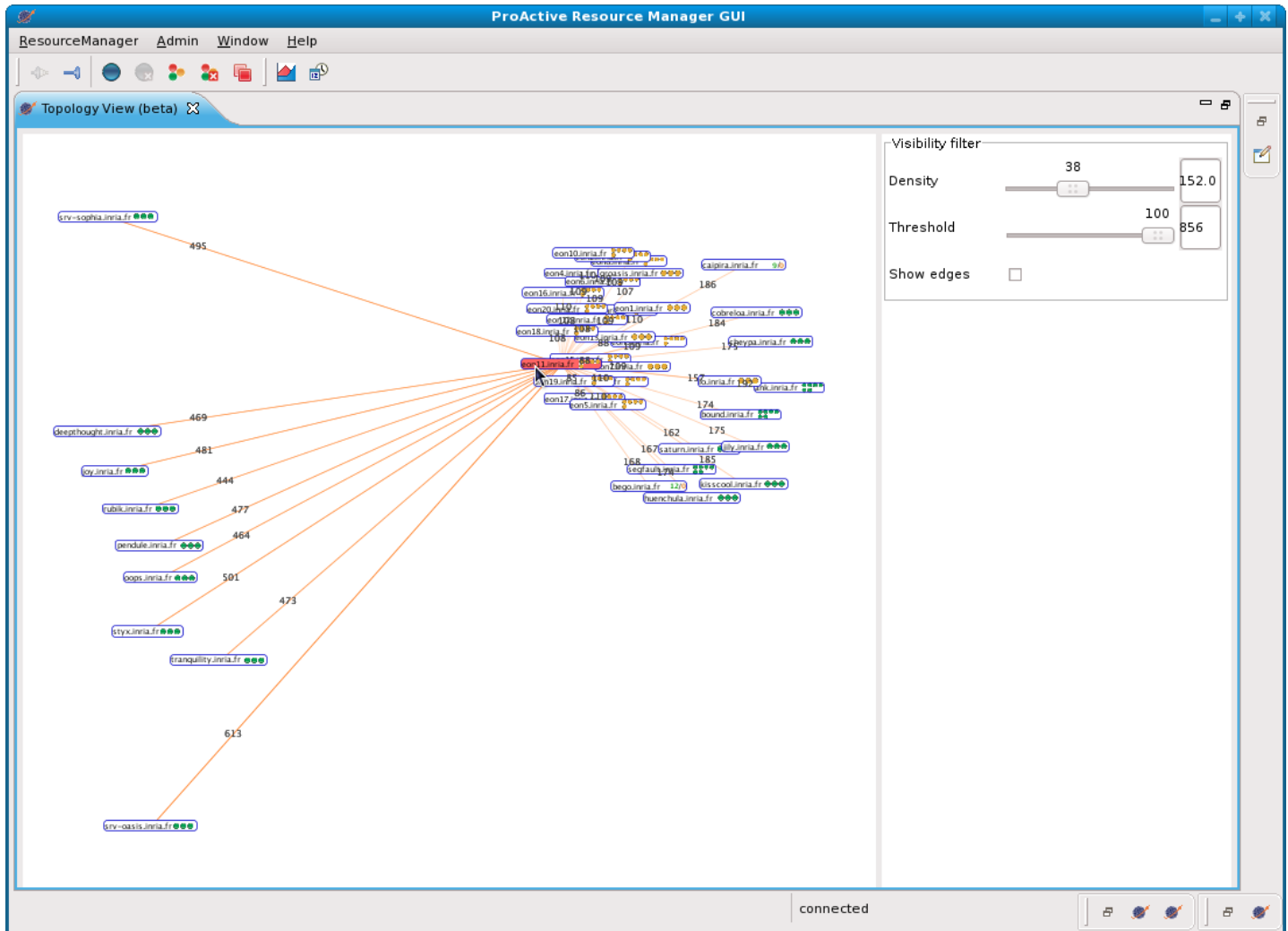


Figure 4.11. Compact view

4.7.3.5. Node icons

In every view, a node is represented by a round icon changing according to its state:



Node is **deploying**, its deployment has been triggered but the node didn't registered yet.



Node is **lost**. The deployment of the node failed. The node won't be added to the resource manager.



Node is **configuring**. The node registered to the resource manager and is under configuration. This step can be time consuming depending above all on the nodes already managed by the resource manager.



Node is **free**, ready to be provided for performing a task (to the Scheduler or another application).



Node is **busy**, it currently performs a task execution.



Node is **to be released**, it currently performs a task execution, and has to be released at the end of this execution.



Node is **down**, it is unreachable by Resource Manager or is broken. A failure has occurred on the node (network failed, computer got down, Java runtime that manages this node got down...).

4.7.3.6. Toolbar

Every view provides a tool bar with different available actions:

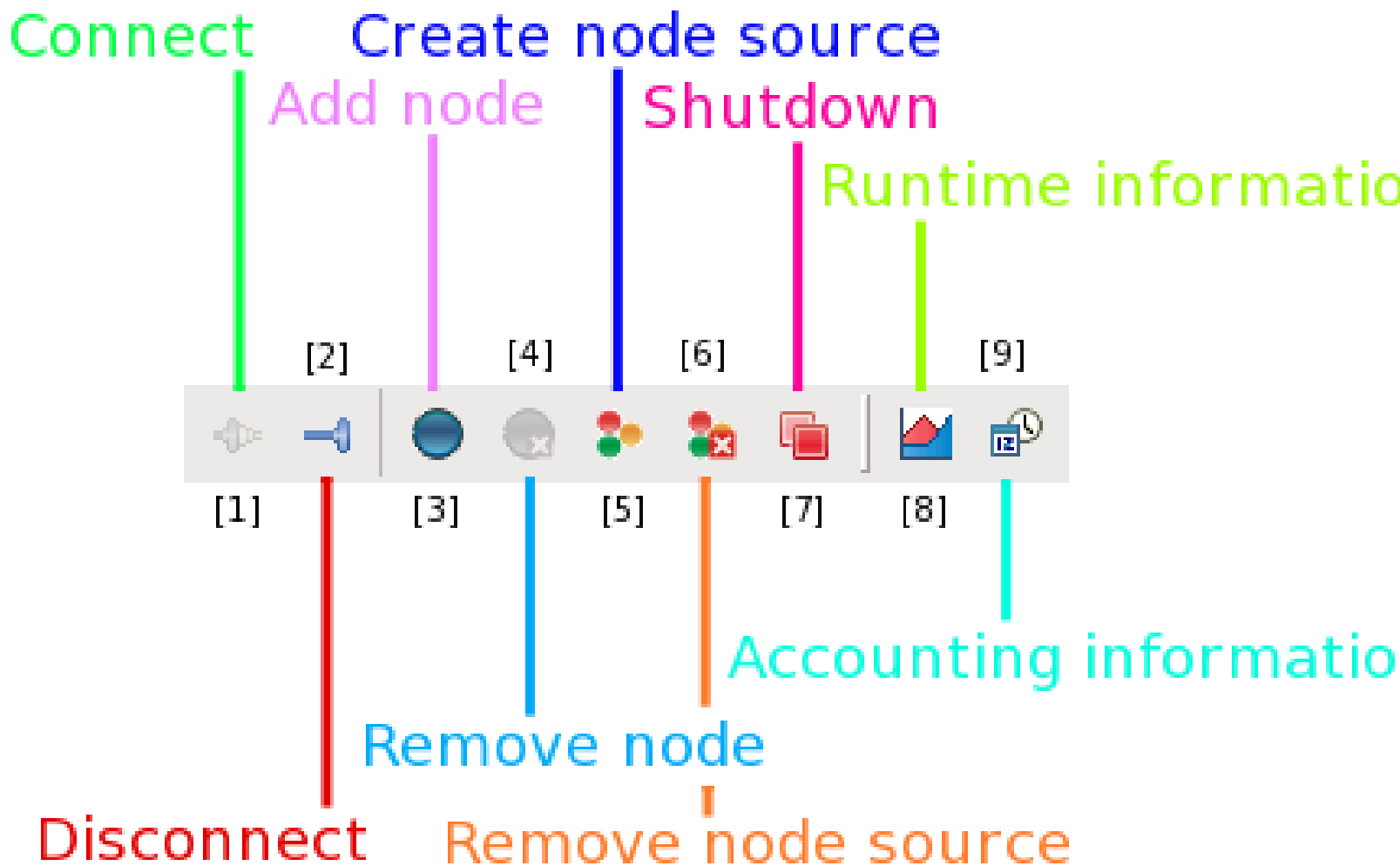


Figure 4.12. Resource Explorer view tool bar

- [1] **Connect** - Open the connection dialog box.
- [2] **Disconnect** - Leave the current Resource Manager.
- [3] **Add node** - Add an existing node to Resource Manager providing its URL.
- [4] **Remove node** - Remove selected nodes. Using the tree view, it is possible for example to remove all the nodes belonging to a JVM, a host or a node source.
- [5] **Create Node Source** - Create a new node source. See [Section 4.7.4.1, “Create a Node Source”](#).
- [6] **Remove Node Source** - Remove a node source from Resource Manager, and release all its nodes.
- [7] **Shutdown Resource Manager** - Stop Resource Manager application, remove all the nodes, and stop Resource Manager daemon.
- [8] **Runtime information** - Show runtime information in the form of graphics.
- [9] **Accounting information** - Show accounting information in the form of graphics.

All these actions are explained hereafter.

4.7.4. Main actions

4.7.4.1. Create a Node Source

This action allows to create a new Node Source in Resource Manager. A Node Source is a RM's component that is able to handle a set of nodes (See [Section 4.3, “Organizing your nodes”](#)). In the dialog, fill in the node source name, then select infrastructure type and policy.

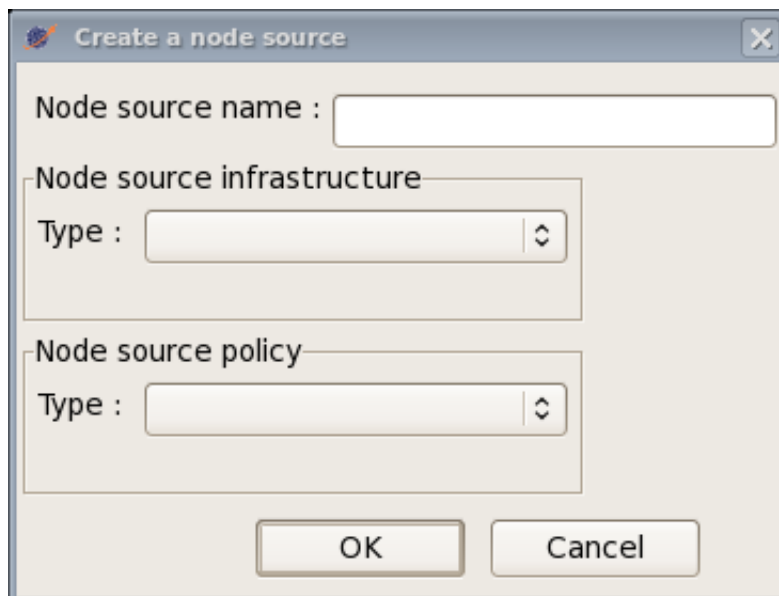


Figure 4.13. Node Source creation dialog box

In this section, we will not describe all infrastructures and policies since the descriptions of every parameter have already been given in [Section 4.3, “Organizing your nodes”](#). We will only give the example of the SSH infrastructure and the Scheduler loading policy.

4.7.4.1.1. SSH infrastructure example

SSH Infrastructure provides a basic but functional way to acquire resources through an SSH connection. Although such acquisition can be achieved using a GCM descriptor, this infrastructure can in some situations be easier to use for SSH-only use-cases.

To acquire nodes, this infrastructure needs to start runtimes on remote hosts. To do so, it needs to know some information, assumed to be true for all hosts:

- **Rm Url** - The resource manager's url that will be used by the deployed nodes to register by themselves.
- **SSH Options** - Options you can pass to the SSHClient executable (-l inria to specify the user for instance)
- **Java Path** - Path to the java executable on the remote hosts.
- **Scheduling Path** - Path to the Scheduling/RM installation directory on the remote hosts.
- **Node Time Out** - A duration after which one the remote nodes are considered to be lost.
- **Attempt** - The number of time the resource manager tries to acquire a node for which one the deployment fails before discarding it forever.
- **Target OS** - One of 'LINUX', 'CYGWIN' or 'WINDOWS' depending on the machines' (in Hosts List file) operating system.
- **Java Options** - Java options appended to the command used to start the node on the remote host.
- **Rm Credentials Path** - The absolute path of the 'rm.cred' file to make the deployed nodes able to register to the resource manager (config/authentication/rm.cred).
- **Hosts List**: The path to a file containing the hosts on which resources should be acquired. This file should contain one host per line, described as a host name or a public IP address, optionally followed by a positive integer describing the number of runtimes to start on the related host (default to 1 if not specified). Example:

```
rm.example.com
test.example.net 5
192.168.9.10 2
```

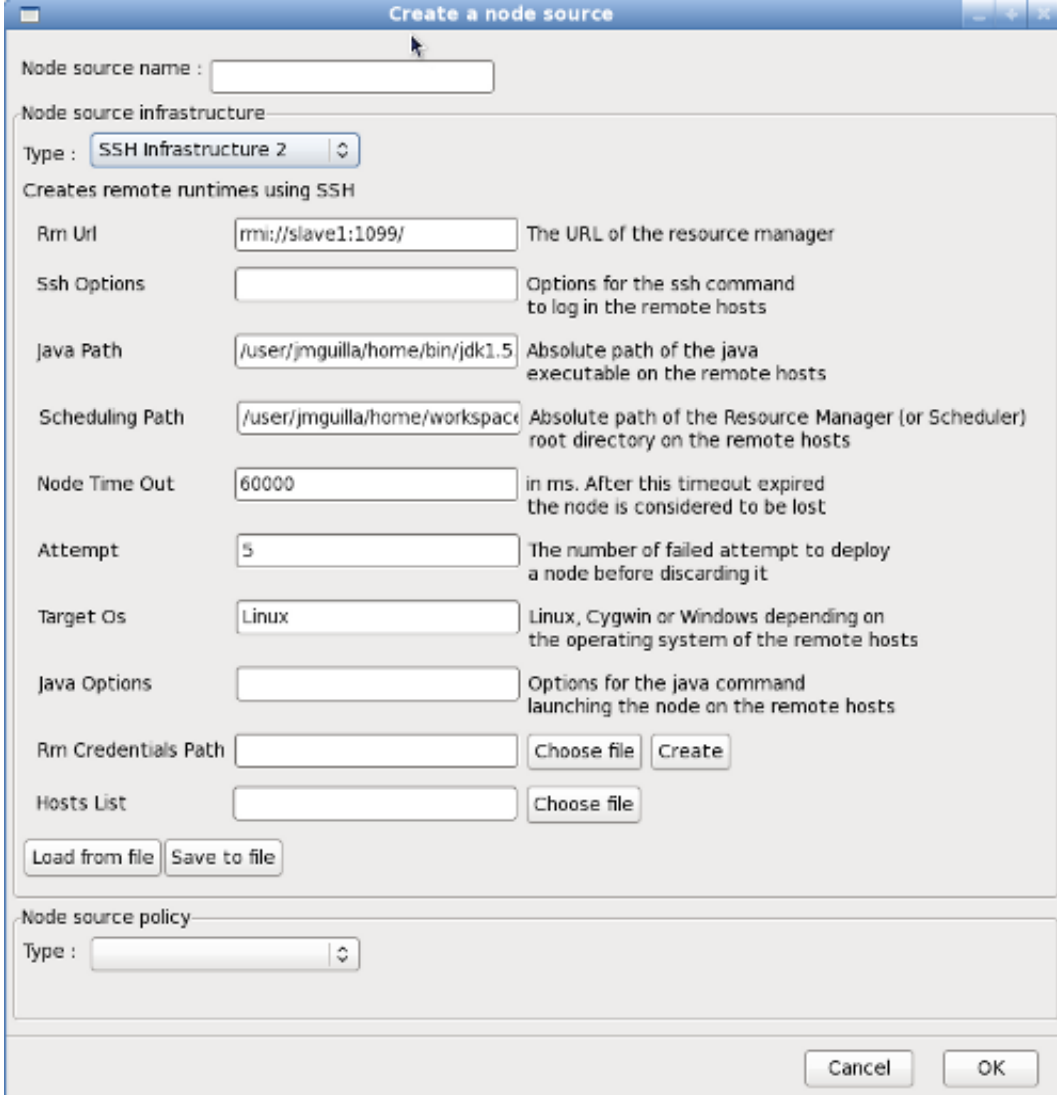


Figure 4.14. SSH infrastructure configuration panel



Tip

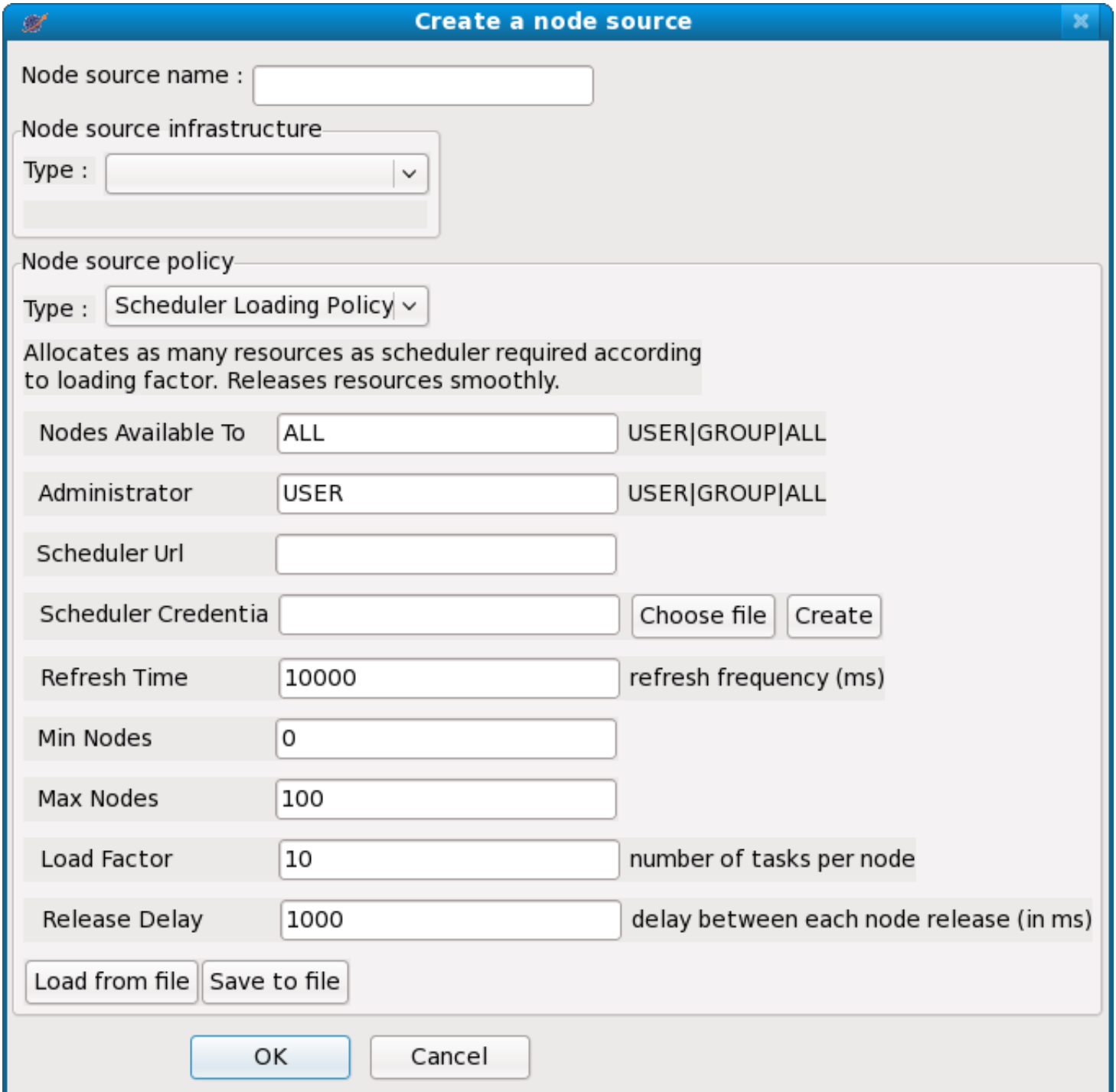
In order to fill in all the fields at once, it is possible to load or to save a file with the list of parameters and values. This file should be of the following form:

```
# My comments
param1=value1
param2=value2
# and so on...
```

4.7.4.1.2. Scheduler loading policy

This policy allows to configure the number of resources which will be always enough for the scheduler. Nodes are acquired and released according to scheduler loading factor which is a number of tasks per node. In the same manner as the previous policy, this one also requires scheduler URL, user name and password. It is important to correctly configure maximum and minimum nodes that this policy will try

to hold. Maximum number should not be greater than potential nodes number which is possible to deploy to underlying infrastructure. If there are more currently acquired nodes than necessary, policy will release them one by one after having waited for a "release period" delay. This smooth release procedure is implemented because deployment time is greater than release one. Thus, this waiting time deters policy from spending all its time trying to deploy nodes.



Create a node source

Node source name :

Node source infrastructure

Type :

Node source policy

Type :

Allocates as many resources as scheduler required according to loading factor. Releases resources smoothly.

Nodes Available To USER|GROUP|ALL

Administrator USER|GROUP|ALL

Scheduler Url

Scheduler Credential

Refresh Time refresh frequency (ms)

Min Nodes

Max Nodes

Load Factor number of tasks per node

Release Delay delay between each node release (in ms)

Figure 4.15. Scheduler loading policy configuration panel

**Note**

This policy is only available when using the Resource Manager with the ProActive Scheduler

**Tip**

In order to fill in all the fields at once, it is possible to load or to save a file with the list of parameters and values. This file should be of the following form:

```
# My comments  
param1=value1  
param2=value2  
# and so on...
```

4.7.4.2. Remove a Node Source

In order to remove a node source, select its name in removal dialog and press OK:

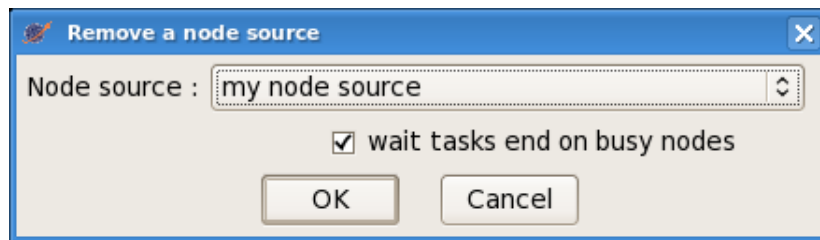


Figure 4.16. Node Source removal dialog box

4.7.4.3. Add node by URL

You can add existing node to any Node Source by providing its URL:

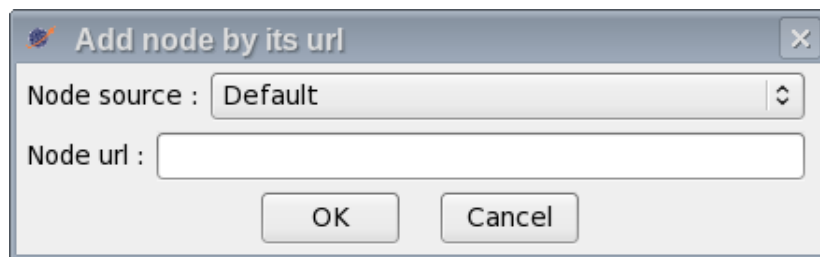


Figure 4.17. Add Node by URL

Choose a Node Source's name from your nodes sources list, and fill the node's URL to add. Node will be added and managed by the node source that you have chosen.

4.7.4.4. Remove a node

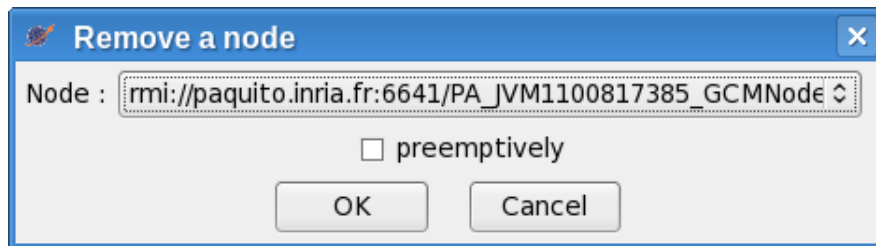


Figure 4.18. Node removal dialog

Choose the node to remove in the list. As explained before, node's removal can be preemptive or not. If you don't choose a preemptive removal and if the node is busy, then it will just put to the "to remove" state, and will be removed at tasks' end.

- If the node to remove is handled by a Static node Source and the node's Java Virtual Machine contains only the node to remove (RM does not handle other node from the same JVM), then the JVM is killed.
- If the node to remove is handled by a dynamic Node Source, then the node's JVM is killed only if the node's removal is preemptive.

4.7.4.5. Shutdown

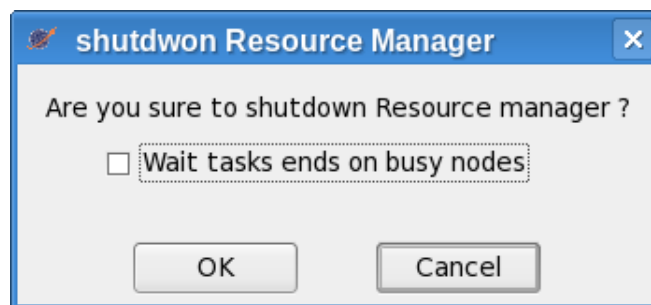


Figure 4.19. RM shutdown dialog box

A dialog asks you to confirm RM's shutdown. All nodes are removed from Resource Manager. Node handled by static nodes sources are destroyed. Nodes handled by dynamic node source are released to their infrastructures. You can tick the check-box **Wait tasks ends on busy nodes**. If checked, and RM has busy nodes, Resource Manager put these busy nodes to the "to remove" state, and wait tasks' end on these nodes, before shutting down. Otherwise, shutdown is performed immediately even if there are busy nodes, tasks are interrupted on these busy nodes.

Chapter 5. ProActive Windows Agent

5.1. Context

In distributed systems, desktop computers can be an important source of computational power. Moreover, one of the definitions of grid stands for a type of parallel and distributed system that enables the sharing, selection, and aggregation of resources distributed across multiple administrative domains based on their (resources) criteria: availability, capacity and performance. In such a context the main purpose of the ProActive Windows Agent is to make the configuration of these criteria achievable (schedule working plan and limit the shared amount of RAM and CPU).

5.2. Functionalities

The ProActive Windows Agent is a Windows Service: a long-running executable that performs specific functions and which is designed not to require user intervention. The agent is able to create a ProActive computational resource on the local machine. This resource will be provided to ProActive applications (such as Resource Manager) according to a user defined schedule. A tray icon shows the state of the agent and allows the user to start/stop it, or modify its schedule. The ProActive Windows Agent does not interfere with the day-to-day usage of the desktop Windows machine.

Understanding of ProActive basic concepts is required for the comprehension of the following description.

The core client is a process which:

- Loads the user's configuration.
- Creates schedules according to the working plan specified in the configuration.
- Spawns a JVM process that will run a specified java class depending on the selected connection type. 3 types of connections are available:
 - **Local Registration** - The specified java class will create a ProActive local node as a computational resource and register it locally.
 - **Resource Manager Registration** - The specified java class will create a ProActive local node as a computational resource and register it in the specified Resource Manager, thus being able to execute java or native tasks received from the Scheduler.

It is important to note that a JVM process running tasks can potentially spawn child processes.

- **Custom** - The user can specify its own java class.
- Watches the spawned JVM process in order to comply to the following limitations:
 - **RAM limitation** - The user can specify a maximum amount of memory allowed for a JVM process and its children. If the limit is reached, then all processes are automatically killed.
 - **CPU limitation** - The user can specify a maximum CPU usage allowed for a JVM process and its children. If the limit is exceeded by the sum of CPU usages of all processes, they are automatically throttled to reach the given limit.
- Restarts the spawned JVM process in case of failures with a timeout policy.



Warning: Do not confuse agent and node deployment!

The role of the agent is to create a node (locally) and to register it to the resource manager (in case of a resource manager connection) which can be distant. The deployment can be considered as the opposite process: the resource manager creates nodes on remote machine and register them to itself. So **when you use the ProActive Windows Agent, you do not need (and do not have) to use a node deployment.**

5.3. Example

This simple example illustrates how to create an active object on a ProActive node created with the ProActive Windows Agent. We suppose that the following conditions are verified:

- The agent is installed (see [Section 5.4, "Installation"](#) for installation) on a host with IP address equal to 192.168.1.62

- It is configured with the **RMI registration** as selected connection type and "toto" as node name.

Once the agent is started, the following java code gets a reference on a remote node and creates an active object on the remote node identified by the url "rmi://192.168.1.62:1099/toto"



Note

This piece of code should be executed on a separate JVM. Moreover, this test is more useful for the RMI Registration connection type since for the Resource Manager one, you can directly see whether the node has been started.

```
import org.objectweb.proactive.api.PAActiveObject;
import org.objectweb.proactive.core.node.Node;
import org.objectweb.proactive.core.node.NodeFactory;

public class Test {

    public static void main(String[] args) {

        try {

            // Note that the port is 1100 and not 1099 like those used by the
            // resource manager. The initial port is incremented to ensure that
            // each runtime uses a unique port.
            // Thus, this port corresponds to the port of the first runtime.
            // If a second runtime has been launched, it would use the port 1101,
            // and so on and so forth...

            final Node n = NodeFactory.getNode("rmi://192.168.1.62:1100/toto");

            System.out.println("Nb of active objects on the remote node: " + n.getNumberOfActiveObjects() +
                " local runtime hashCode " + Runtime.getRuntime().hashCode());

            final Test stubOnTest = (Test) PAActiveObject.newActive(Test.class.getName(), null, n);

            final String receivedMessage = stubOnTest.getMessage();

            System.out.println("Nb of active objects on the remote node: " + n.getNumberOfActiveObjects() +
                " received message: " + receivedMessage);
        } catch (Throwable t) {
            t.printStackTrace();
        }
    }

    public Test() {
    }

    public String getMessage() {
        return "A message from " + Runtime.getRuntime().hashCode();
    }
}
```

The output is:

```
--> This ClassFileServer is listening on port 2027
Detected an existing RMI Registry on port 1099
Nb of active objects on the remote node: 0 local runtime hashcode 6588476
Generating class : pa.stub.org.ow2.proactive.resourcemanager.utils._StubTest
Nb of active objects on the remote node: 1 received message: A message from 26670261
```

5.4. Installation

The ProActive Windows Agent installation pack is available on the official [ProActive website](http://www.proactive.inria.fr/)¹. Follow the links and get the latest version.



Note

- .Net framework v3.5 or later should be installed on your system. If not, the installer will ask you to install it. Go to <http://www.microsoft.com/downloads> and download Microsoft .NET Framework Version 3.5 (or later) Redistributable Package.
 - Visual C++ 2008 (or later) Redistributable Package is also needed. The installer will install it for you, if not found.
- Run the **setup.exe** file.
 - Accept the licence agreement.
 - Select the components you want to install.
 - Choose the installation folder of the ProActive agent.
 - Then, the following windows will appear:

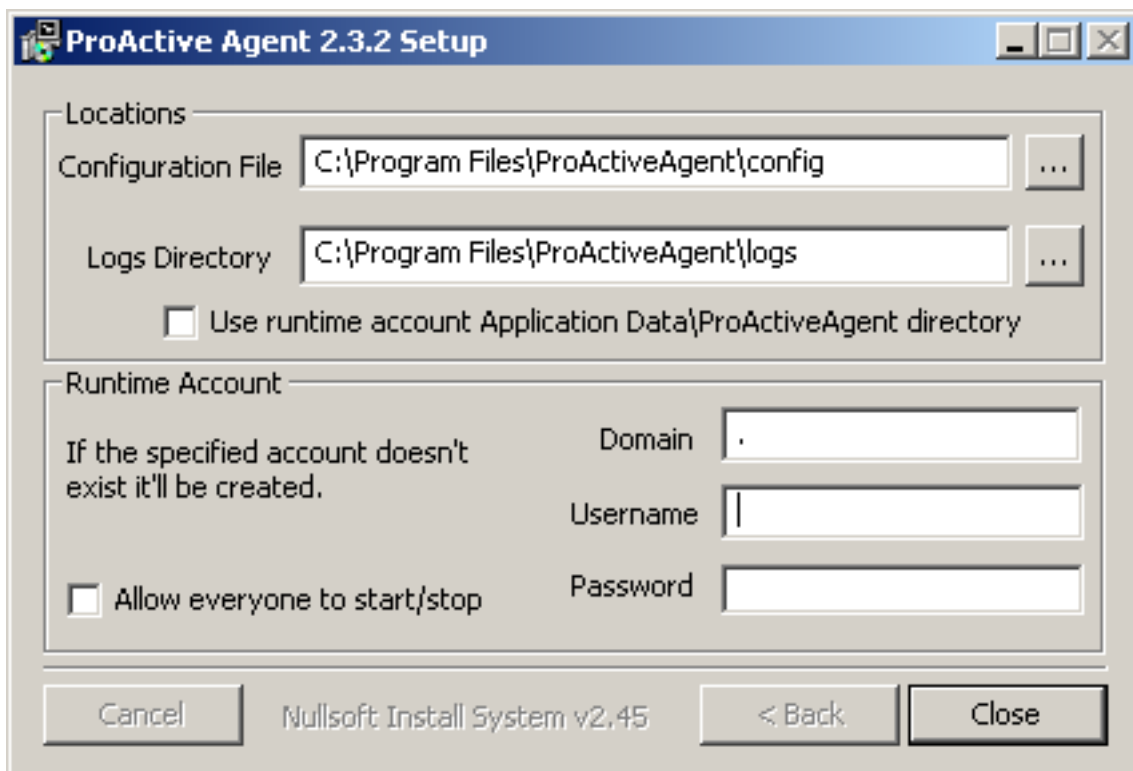


Figure 5.1. Configuration during installation

¹ <http://proactive.inria.fr/>

- Specify the directory that will contain the configuration file named PAAgent-config.xml, note that if this file already exists in the specified directory it will be re-used.
- Specify the directory that will contain the log files of the ProActive Agent and the spawned runtimes.
- Specify an existing, local account under which the ProActive Runtime(s) will be spawned. It is highly recommended to specify an account that is not part of the Administrators group to isolate the ProActive Runtime and reduce security risks.

The password is encrypted using [Microsoft AES Cryptographic Provider](#)² and only Administrators have access permissions to the keyfile (restrict.dat) this is done using the [SubInACL](#)³ tool.

If the specified account does not exist the installation program will prompt the user to create a non-admin account with the required privileges.

Note that the ProActive Agent service is installed under [LocalSystem](#)⁴ account, this should not be changed, however it can be using the "services.msc" utility. ("Control Panel->Administrative Tools->Services")

- If you want that any non-admin user (except guest accounts) to be able to start/stop the ProActive Agent service check the "Allow everyone to start/stop" box. If this option is checked the installer will use the [SubInACL](#)⁵ tool. If the tool is not installed in the "Program Files\Windows Resource Kits\Tools" directory the installer will try to download its installer from the official Microsoft page.
- The installer will check whether the selected user account has the required privileges. If not follow the steps to add these privileges:
 - In the 'Administrative Tools' of the 'Control Panel', open the 'Local Security Policy'.
 - In 'Security Settings', select 'Local Policies' then select 'User Rights Assignments'.
 - Finally, in the list of policies, open the properties of 'Replace a process-level token' policy and add the needed user. Do the same for 'Adjust memory quotas for a process'. For more information about these privileges refer to the official Microsoft [page](#)⁶.

At the end of the installation, the ProActive Agent Control utility should be started. This next section explains how to configure it.

To uninstall the ProActive Windows Agent, simply run "Start->Programs->ProActiveAgent->Uninstall ProActive Agent".

5.5. Configuration

Launch "Start/Programs/ProActiveAgent/AgentControl" program or click on the notify icon if the "Automatic launch" is activated. Double click on the tray icon to open the ProActive Agent Control window. The following window will appear:

² <http://msdn.microsoft.com/en-us/library/aa386979%28v=vs.85%29.aspx>

³ <http://www.microsoft.com/downloads/en/details.aspx?FamilyID=e8ba3e56-d8fe-4a91-93cf-ed6985e3927b>

⁴ [http://msdn.microsoft.com/en-us/library/ms684190\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms684190(VS.85).aspx)

⁵ <http://www.microsoft.com/downloads/en/details.aspx?FamilyID=e8ba3e56-d8fe-4a91-93cf-ed6985e3927b>

⁶ [http://msdn.microsoft.com/en-us/library/bb530716\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/bb530716(VS.85).aspx)

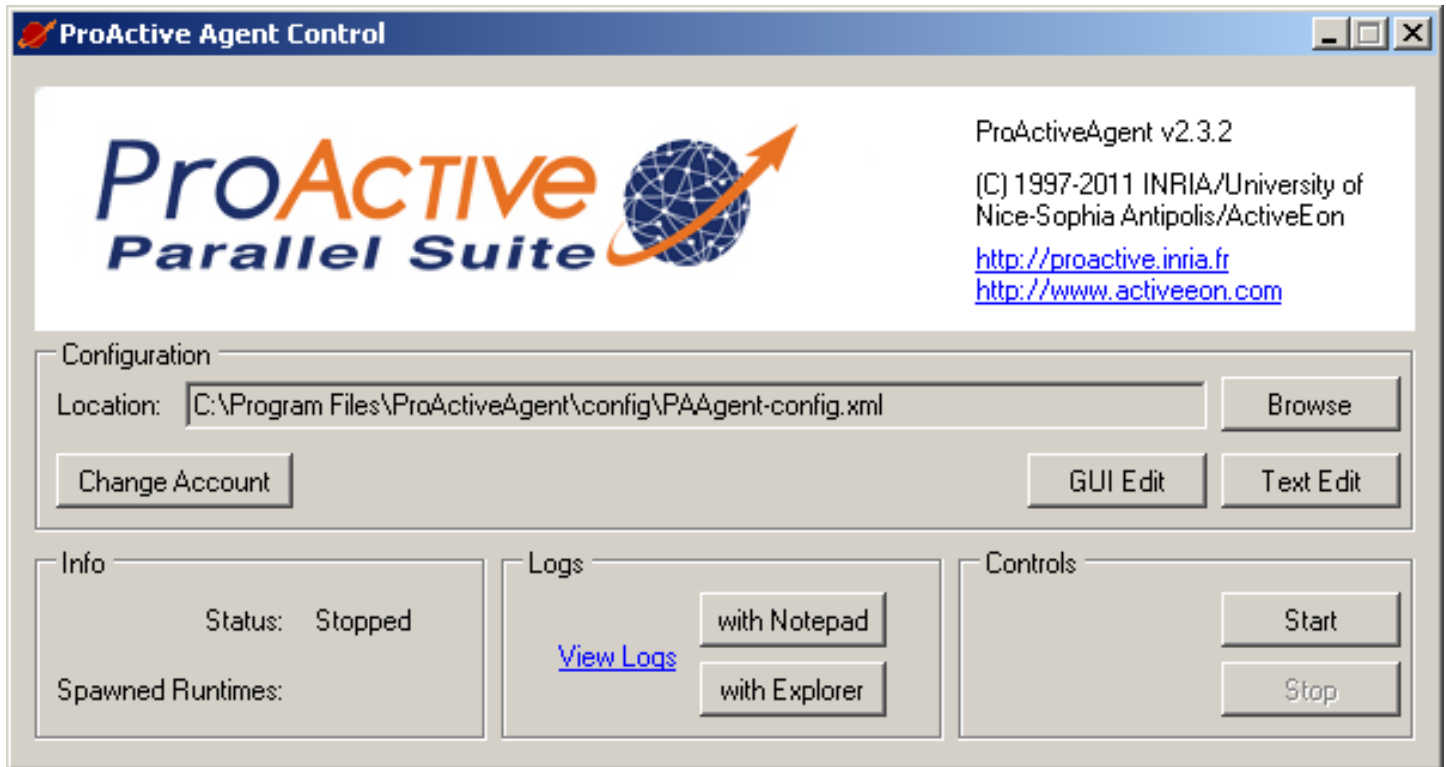


Figure 5.2. ProActive Agent Control window

From the ProActive Agent Control window, the user can load a configuration file, edit it, start/stop the service and view logs. A GUI for editing is provided (explained below). Even if it is not recommended, the user can edit the configuration file by yourself with your favorite text editor.

It is also possible to change the ProActive Runtime Account using the "Change Account" button.



Warning

The configuration file format has changed since version 2.2 which is backward compatible with previous version files. If the user edits such a configuration and saves the modifications the file will be overwritten with the new format.

Clicking on "GUI Edit", the following windows appears:

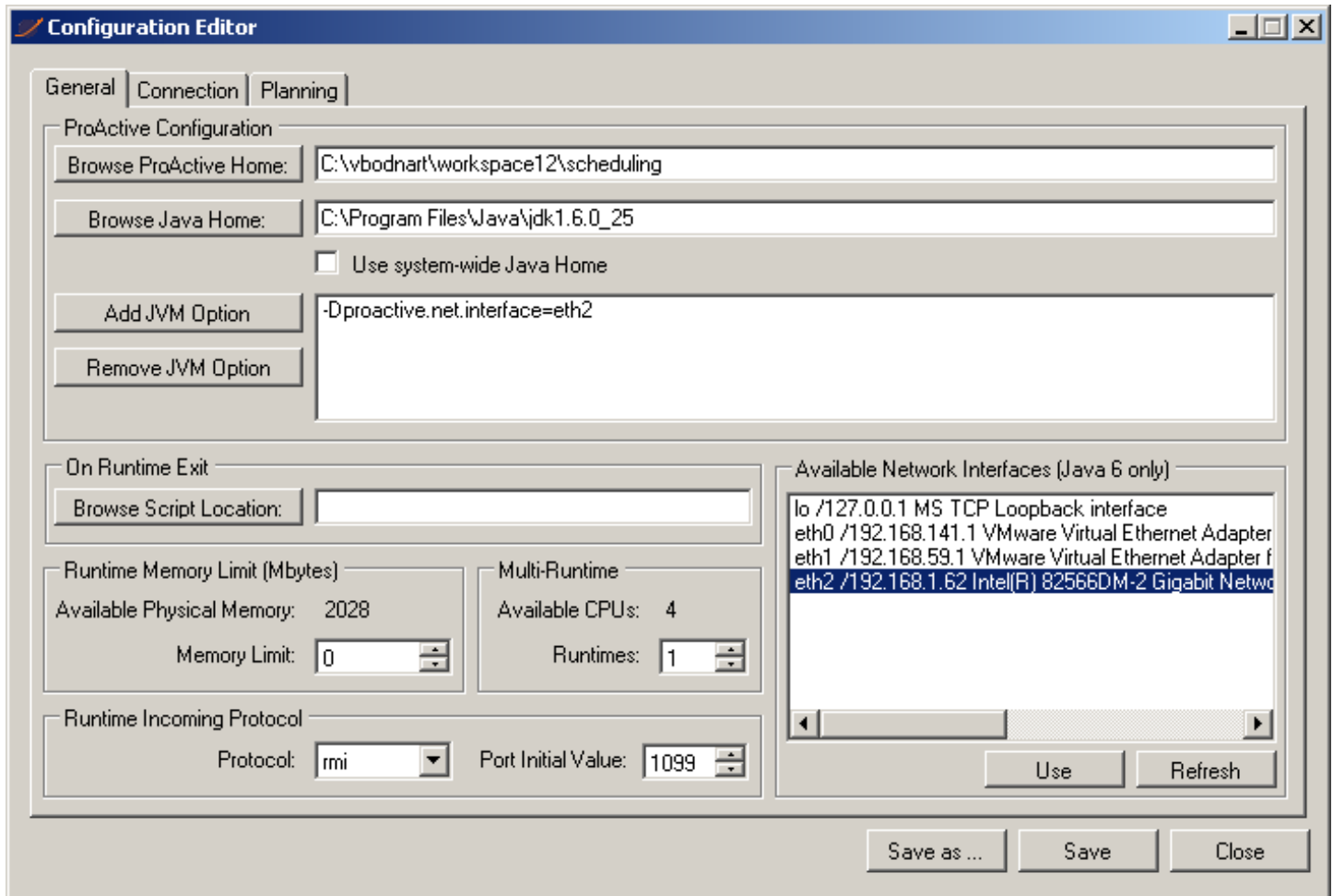


Figure 5.3. Configuration Editor window - General Tab

In the general tab, the user can specify:

- The ProActive (or Scheduler) location.
- The JVM location (usually something like C:\Program Files\Java\jdk1.6.0_12).
- The number of Runtimes (the number of spawned JVMs).
- The JVM options.

Note that if the parameter contains $\${rank}$, it will be dynamically replaced by the Runtime rank starting from 0.

- The "On Runtime Exit" script. A script executed after a Runtime exits. This can be useful to perform additional cleaning operation.

Note that the script receives as parameter the PID of the Runtime.

- The user can set a memory limit that will prevent the spawned processes to exceed a specified amount of RAM. If a spawned process or its child process requires more memory, it will be killed as well as its child processes.

Note that this limit is disabled by default (0 means no limit) and a ProActive Runtime will require at least 128 MBytes.

- It is possible to list all available network interfaces by clicking on the "Refresh" button and add the selected network interface name as a value of the "proactive.net.interface" property by clicking on "Use" button. See the ProActive documentation for further information.
- The user can specify the protocol (rmi or http) to be used by the Runtime for incoming communications.

To ensure that a unique port is used by a Runtime, the initial port value will be incremented for each Runtime and given as value of the "-Dproactive.SELECTED_PROTOCOL.port" JVM property. If the port chosen for a runtime is already used, it is incremented until an available port number is found.

Clicking on the "Connection" tab, the windows will look like this:

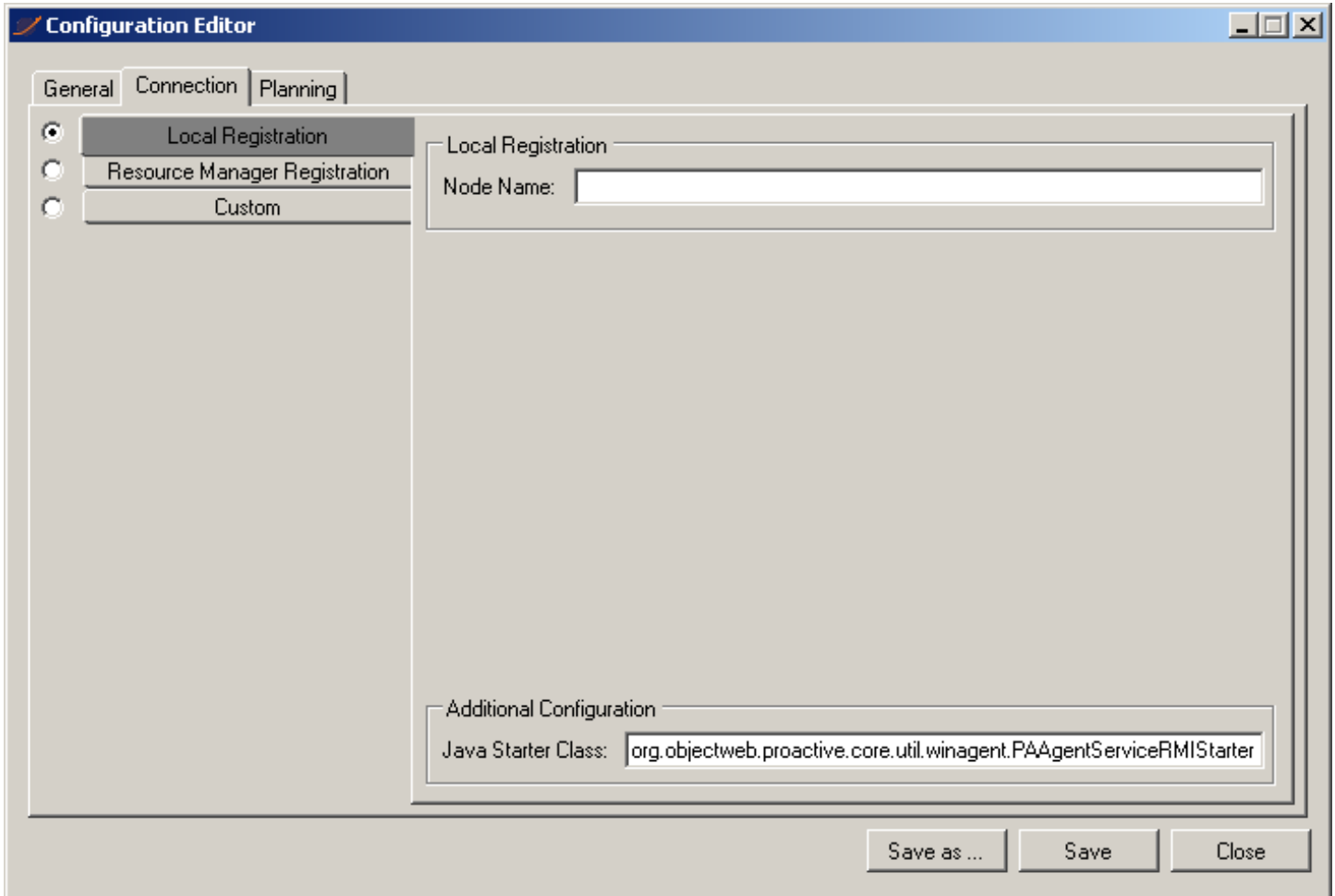


Figure 5.4. Configuration Editor window - Connection Tab (Resource Manager Registration)

In the "Connection" tab, the user can select between three types of connections:

- **Local Registration** - creates a local ProActive node and registers (advertises) it in a local RMI registry. The node name is optional.
- **Resource Manager Registration** - creates a local ProActive node and registers it in the specified Resource Manager. The mandatory Resource Manager's url must be like 'protocol://host:port/'. The node name and the node source name are optional. Since the Resource Manager requires an authentication, the user specifies the file that contains the credential. If no file is specified the default one usually located in "%USERPROFILE%\proactive\security" folder is used.
- **Custom** - the user specifies its own java starter class and the arguments to be given to the main method. The java starter class must be in the classpath when the JVM process is started.

Finally, clicking on the "Planning" tab, the windows will look like this:

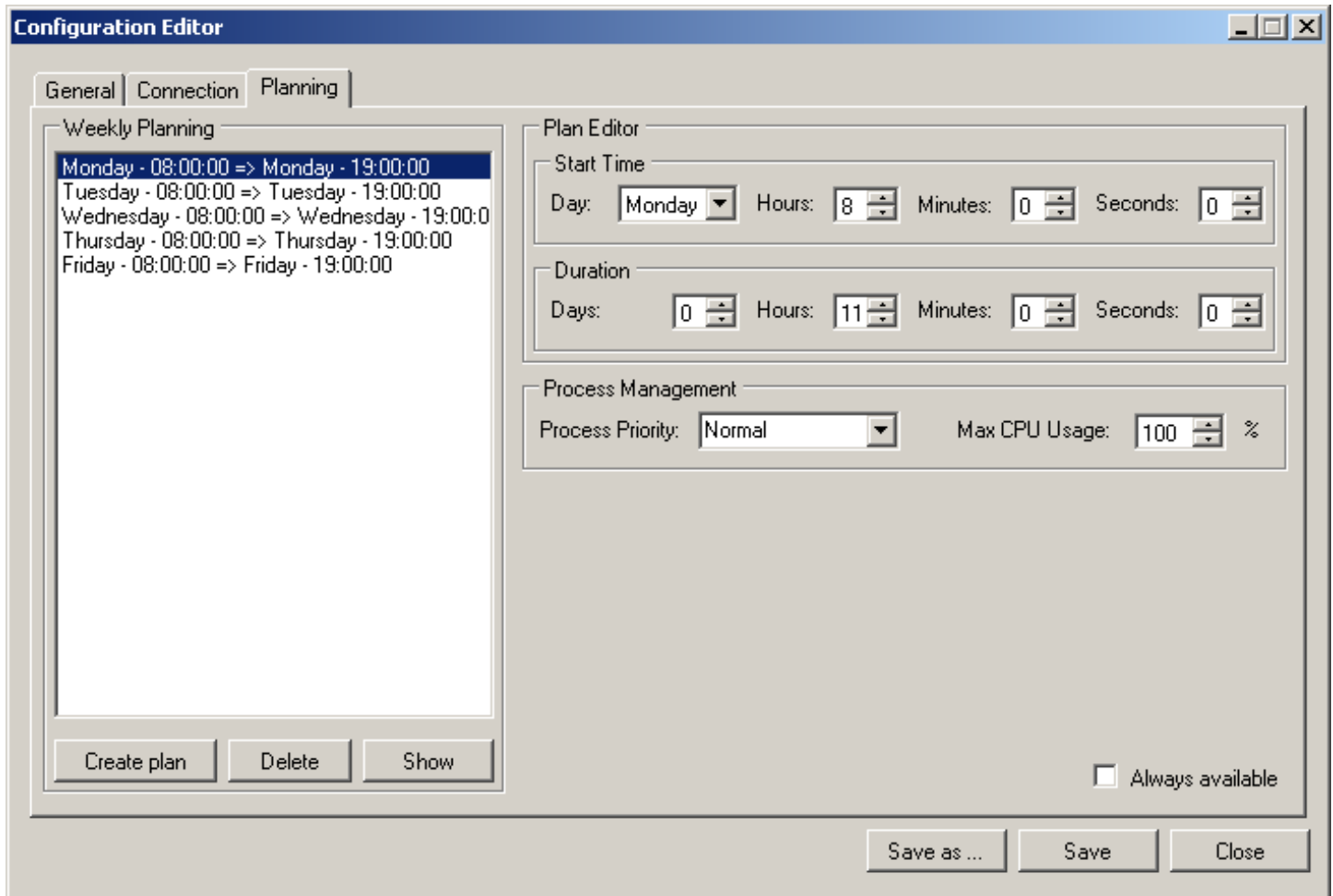


Figure 5.5. Configuration Editor window - Planning Tab

In the Planning Tab, depending on the selected connection type, the agent will initiate it according to a weekly planning where each plan specifies the connection start time as well as the working duration. The agent will end the connection as well as the ProActive Runtime process and its child processes when the plan duration has expired.

Moreover, it is possible to specify the JVM process Priority and its CPU usage limit. The behavior of the CPU usage limit works as follows: if the JVM process spawns other processes, they will also be part of the limit so that if the sum of CPU% of all processes exceeds the user limit they will be throttled to reach the given limit. Note that if the Priority is set to RealTime the CPU % throttling will be disabled.

The "Always available" makes the agent to run permanently with a Normal Priority and Max CPU usage at 100%.

5.6. Start the agent

Once you have configured the agent, you can start it clicking on the "Start" button of the ProActive Agent Control window. However, before that, you have to ensure that a resource manager has been started on the address you specified in the agent configuration. **You do not need to start a node since it is exactly the job of the agent.**

Once started, you may face some problems. You can realise that an error occurred by first glancing at the color of the agent tray icon. If everything goes right, it should keep the blue color. If its color changes to yellow, it means that the agent has been stopped. To see exactly what happened, you can look at the runtime log file located into the agent installation directory and named **Executor<runtime number>Process-log.txt**.

The main troubles you may have to face are the following ones:

- You get an access denied error: this is probably due to your default java.security.policy file which cannot be found. If you want to specify another policy file, you have to add a JVM parameter in the agent configuration. A policy file is supplied in the scheduling directory. To use it, add the following line in the JVM parameter box of the agent configuration ([Figure 5.3, “Configuration Editor window - General Tab”](#)):

```
-Djava.security.poly=<scheduler directory>/config/security.java.policy-client
```

- You get an authentication error: this is probably due to your default credentials file which cannot be found. In the "Connection" tab of the Configuration Editor ([Figure 5.4, “Configuration Editor window - Connection Tab \(Resource Manager Registration\)”](#)), you can choose the credentials file you want. You can select, for instance, the credentials file located at <scheduler directory>/config/authentication/scheduler.cred or your own credentials file.
- The node seems to be well started but you cannot retrieve it in a Java code for example: in this case, make sure that the port number is the good one. Do not forget that the runtime port number is incremented from the initial resource manager port number. You can see exactly on which port your runtime has been started looking at the log file describing above.

5.7. Other information

The agent was successfully tested on Windows XP, 2003, Windows Vista and Windows 7. Some problems with service installation can occur on Windows NT.

The ProActive Windows Agent is written in C# and uses .Net Framework 3.5

Third-party libs used:

- C# JobObjectWrapper Api (JobManagement.dll) under Microsoft Permissive License (Ms-PL) v1.1
- C# Log4Net Api (log4net.dll - 1.2.10.0) under Apache License v2.0