JaDOrT

User's guide

v1.6

JaDOrT: User's guide

JASMINe Team

Publication date $Id: jadort_guide.xml 6873 2010-09-07 10:25:24Z alitokmen $

Table of Contents

List of Figures

Chapter 1. Introduction

1.1. What is JaDOrT for?

During its lifetime, any application server infrastructure must undergo three kinds of operations:

1. **Application deployment**: The deployment of a new application on the infrastructure.
2. **Application upgrade**: The deployment of a new version of an existing application.
3. **Maintenance**: Hardware or software maintenance that requires the affected server to be shut down and restarted.

The **JA**SMINe **D**eployment **Or**chestration **T**ool (JaDOrT) helps the administrator in the execution of these types of operations:

- Gives a global view of the infrastructure:
  - Groups of servers and their members
  - Applications deployed on each group
  - Links between reverse proxies (whether for load balancing, high availability or isolation purposes) and each server they're associated with
  - Virtual Machine infrastructures hosting these servers
- Does a full and flexible guidance in the execution of each type of operation (deployment, upgrade or maintenance):
  - List of steps for each type of operation, with a progress indicator
  - Precondition checking (server presence, application packaging, etc.) before starting deployment, upgrade or maintenance
  - User session management: servers or application versions that will be shut down are disabled (so new users don't connect to these servers) and only shut down when all users have finished working on them
  - Progressive deployment, upgrade and maintenance (for example, deploy on servers 1 and 2, then the next 10 servers, and so on)
- Lets the administrator take control of the situation at any time
  - Cancel any step
  - Fix any problem manually (and tell JaDOrT to "trust her / him")
- Persists all data in order to:
  - Undo any step
  - Resume a process
  - Analyze the deployment:
    - Errors and their causes

  - Full logs in order to analyze success
  - Calculate other statistics: typical errors that occur when deploying a given application, average deploy time, etc.

1.2. System requirements

JaDOrT is a standard Java EE application. Its requirements are as follows:

- **On the server side**: any Java EE 5 compliant application server with its EJB3, persistence and JMS services configured and activated. We recommend you to use the OW2 JOnAS 5 application server [http://jonas.ow2.org/], version 5.1.0 or newer.
- **On the client side, using the Web console**: any Internet browser with the Flash plug-in, compatible with Adobe Flash version 9 or later. JaDOrT doesn't have any specific proxy or firewall configuration; it uses standard server-client HTTP exchanges.
- **On the client side, using the EJB3 interface**: Java 5 or newer.

1.3. Supported infrastructure elements

JaDOrT can be used to manage infrastructures with many kinds of elements.

1.3.1. Application servers

- OW2 JOnAS 4.x and 5.x
- Glassfish 2.x
- JBoss 4.x, 5.x
- Oracle/BEA Weblogic 10.x
- IBM Websphere 7.x

1.3.2. OSGi gateways

- OW2 JOnAS 5.x
- Any OSGi gateway with the BundleManager installed

1.3.3. Reverse proxies

- Apache Tomcat mod_jk 1.2.x
- Apache httpd mod_proxy_balancer 2.2.x

1.3.4. Virtual Machines

For managing VMs, JaDOrT uses the JASMINe Virtual Machine Management (VMM) APIs (tested with version 0.6.3). That version supports the following:

- Hyper-V
- LibVirt (many other VMs, including KVM and VirtualBox)
- VMWare (based on the VMWare VirtualCenter driver)

- Xen

Chapter 2. Glossary

2.1. What is an operation ?

An operation is the general term for referring to a version migration or deployment process within JaDOrT.

2.2. What is a topology ?

JaDOrT uses topology files to describe the infrastructure on which you can carry out operations. This description includes:

- Application server instances:
  - Their configuration (cluster or farm)
  - Their JMX connectors
  - Their logical domains
  - Their daemon instances (typically used for starting and stopping the servers)
- Reverse proxy instances (these are called "Load Balancer" in the topology file):
  - Their connectors
  - The link between reverse proxies and application servers; i.e. the list of workers on a given reverse proxy redirecting requests to an application server.
- Virtual Machine Manager instances:
  - Their connectors
  - The link between Virtual Machines and application servers; i.e. which application server is running on which Virtual Machine Manager domain.

The **JaDOrT Samples and Documentation** distribution includes many examples of topology files as well as the XSD files defining the topology format. You can use the XML files in that distribution as a basis for creating your topology files. You will find that distribution on the JASMINe download page: http://wiki.jasmine.ow2.org/xwiki/bin/view/Main/Downloads [http://wiki.jasmine.ow2.org/xwiki/bin/view/Main/Downloads#jadort]

You can also use JASMINe Design [http://wiki.jasmine.ow2.org/xwiki/bin/view/Main/Design] to create your topology files graphically. JASMINe Design will then check constraints on your infrastructure (it can for example detect non-compatible cluster or loadbalancing configurations) and ease the integration with other JASMINe tools (monitoring, server deployment, automated configuration, etc.).

2.3. What is a migration ?

The term *migration* groups the first two types of operations cited in the introduction, namely **Application deployment** and **Application upgrade**.

A migration process can be executed on JOnAS 5 or OSGi servers.

2.3.1. Application deployment

The deployment of a new application is done in 3 steps:

- **Upload New Version**: The new application is sent to the application servers.
- **Deploy New Version**: The new application is deployed on the application servers.
- **Set the New Version as Default**: The policy of the new application is set to "Default"; in turn the old "Default" version of the same application becomes "Reserved". This means that:
  - Users that were on the old version of the application keep on using that version as long as their session on that version is active.
  - Newly connected users will use the new version.

2.3.2. Application upgrade

The deployment of a new version of an existing application is done in 5 steps.

The first 3 steps are the same as the steps for the deployment of a new application. Once the deployment of the new version is completed on all servers, these 2 steps are executed:

- **Undeploy Old Version**: Undeploys the old version of the application, which results in the old version becoming inaccessible. Users that had an active session on that version will be disconnected.
- **Erase Old Version**: Erases the old version of the application from the application servers.

Note

You cannot go back to the previous step once the "Erase Old Version" task has been initiated.

2.3.3. OSGi bundle deployment

The deployment of a new OSGi bundle is done in 3 steps:

- **Upload New Version**: The new OSGi bundle is sent to the OSGi gateways.
- **Undeploy Old Version**: Undeploys any old version of the same OSGi bundle.
- **Deploy New Version**: The new OSGi bundle is deployed on the OSGi gateways.

2.3.4. OSGi bundle upgrade

If any old version of the OSGi bundle was present on a gateway, the following step will be executed once the new version has been deployed on all OSGi gateways:

- **Erase Old Version**: Erases the old version of the OSGi bundle from the OSGi gateways.

Note

You cannot go back to the previous step once the "Erase Old Version" task has been initiated.

2.4. What is a maintenance ?

Maintenance is a process which can be used for any kind of updates (application server, hardware, VM, etc.) on the infrastructure (the third type of operation cited in the introduction).

JaDOrT allows the maintenance of a group of JOnAS, Glassfish, JBoss, WebLogic and/or WebSphere servers. In that case, JaDOrT handles the session and load management. It can be used for any kind of maintenance (server, VM).

2.4.1. General steps for a maintenance

Maintenance is done in five sub steps:

- **Disable worker**: Disable the associated worker on the reverse proxy. Once this is done, no new clients will get connected to this server.
- **Stop server**: Stop the application server and the VM.
- **Maintain server**: Update the server instance. JaDOrT clearly differenciates two cases:
  - If the topology did not define any Virtual Machines for this server, maintenance is manual.
  - Else, JaDOrT connects to the Virtual Machine Manager in order to create a new Virtual Machine host (based on the initial machine's configuration) and deploys the specified VM image on this host (therefore initializes the hard drives).
- **Start server**: Start the new VM and the new application server instance.
- **Enable worker**: Enable the associated worker on the reverse proxy. Once this is done, this server becomes accessible to all clients.

2.4.2. Maintenance of a farm

In a farm (as opposed to a cluster), user sessions are not replicated. As to guarantee that these won't be lost during the operation, JaDOrT will execute another step between the **Disable worker** and **Stop server** steps:

- **Disable applications**: All applications on the server get disabled. As a result, only non-expired sessions will be allowed on the server.

Moreover, before the server can be stopped, JaDOrT will also check that there are no active sessions on it.

2.4.3. Maintenance of a Virtual Machine

In a virtual machine infrastructure, JaDOrT will have created a new VM host during the **Maintain server** step. As a result, once all VMs in the infrastructure have been maintained, JaDOrT executes another step:

- **Destroy old VMs**: All old VM instances, that are not running anymore, are deleted to restore the disk space.

Note

You cannot go back to the previous step once the "Erase Old Version" task has been initiated.

Chapter 3. Web interface

3.1. Introduction

The following diagram describes the Home Page of the Web interface for JaDOrT.

Figure 3.1. JaDOrT home page

1. **Progress step panel**: This panel indicates the current major step and the overall progress.
2. **Main panel**: This panel shows the details for each step. We will be detailing it in each paragraph of this chapter.
3. **Button panel**: This panel is used for navigation between steps. It contains two buttons:
   - **Undo**: Go to the previous step.
   - **Next**: Go to the next step.

   Each of these buttons will only be available (non-grayed) if it can be clicked on. They're refreshed automatically.

3.2. Operation selection

The following diagram describes the home page of the Web interface.

Figure 3.2. Operation selection page

To create a new operation, you must enter the operation name and click on **Create** button.

To resume a previous operation, click on the **Resume** button in the **Action** column of the table. When this button is clicked, JaDOrT loads the operation data from the database and resumes the operation from where has stopped.

> ### Note
>
> A previous operation is an operation that has been started in the past. It might still be in progress or not; in all cases all logs are persisted.

To delete or abort an operation, click the **Delete** button. This deletes all data associated with that operation (including all deployment logs).

3.3. Topology initialization

Figure 3.3. Operation selection page

In this step, you must give an XML file that describes your infrastructure, called a *topology*. Here is an example of topology XML file:

Figure 3.4. Example of a topology.xml file

Note

The **JaDOrT Samples and Documentation** distribution includes many examples of topology files as well as the XSD files defining the topology format. You can use the XML files in that distribution as a basis for creating your topology files. You will find that distribution on the JASMINe download page: http://wiki.jasmine.ow2.org/xwiki/bin/view/Main/Downloads [http://wiki.jasmine.ow2.org/xwiki/bin/view/Main/Downloads#jadort]

You can also use JASMINe Design [http://wiki.jasmine.ow2.org/xwiki/bin/view/Main/Design] to create your topology files graphically. JASMINe Design will then check constraints on your infrastructure (it can for example detect non-compatible cluster or loadbalancing configurations) and ease the integration with other JASMINe tools (monitoring, server deployment, automated configuration, etc.).

To upload your topology file:

1. Click the **Browse** button and select the topology XML file
2. Click on the **Upload** button and wait for the **Topology loaded** message to appear
3. Once the message appears, the **Next** buttom gets enabled. Click on it to go to the next step.

3.4. Group selection

In that step, JaDOrT lists the groups of servers loaded from the uploaded XML file.

Figure 3.5. Server group selection page

Note

The list of applications deployed on the group of servers is not loaded from the topology XML file but rather obtained "on the fly" by asking the group of servers.

To select the group of server on which the migration or maintenance should be performed, click on the corresponding **Select** button.

Note

Once you click the **Select** button, JaDOrT verifies that all elements of that group are online (all servers are connected, all VMs and VM Managers are present and all reverse proxies accessible and properly configured). If any check fails, JaDOrT will show an error message telling which element is not accessible and for which reason (server shut down, VM host not present, worker name not present, etc.).

3.5. Operation type selection

In that step, you can choose the type oh the operation: Migration or Maintenance. For each choice, JaDOrT describes in detail what will be done based on the current selected group.

Figure 3.6. Operation type selection page

3.6. Migration

JaDOrT allows the deployment of applications on a group of JOnAS or OSGi servers. In that case, JaDOrT handles the proper uploading, activation and version management of applications.

3.6.1. Application selection

Figure 3.7. Application selection page

In order to select an application, you must upload the corresponding file by either:

- **Uploading it from your computer**: click the **Browse** button, select the archive and then click on the **Upload** button to upload the archive on the server where JaDOrT is deployed.
- **Downloading from a URL**: If the application is too big, you can enter a URL where the file is located and click to **Load** button. That button gets the file from the entered URL and saves it on the server where JaDOrT is deployed.

To start the application migration execution, click the **Next** button in the bottom panel.

3.6.2. Servers selection

JaDOrT supports migration in groups of servers. For example, you can deploy the application on the first 3 servers, then on the following 10 servers, and so on; in order to guarantee a certain quality of service or for pre-production testing and validation purposes.

Figure 3.8. Servers selection page

You have two ways to select servers:

- **Manual selection**: by checking the check box of the correspondent server.
- **Automatic selection**: by Clicking the **Click here** button at the bottom of the servers list. JaDOrT will then select all servers that can be maintained right now (based on capacity check).

JaDOrT displays three important informations:

- Each server's capacity
- Number of active session on each server
- Migration status: **Migrated** or **Not Migrated**

These information assist the choice of the server(s) to migrate.

Once some servers are selected, click the **Next** button in the bottom panel in order to continue.

Note

JaDOrT will come back to this screen after every **Migration execution** iteration, as long as there are non-migrated servers remaining.

3.6.3. Migration execution

Figure 3.9. The migration execution page

Figure 3.10. The migration execution page: Upload new version

In the figure above, you can find the following information:

- Current global task in the migration process is **Deploy New Version**
- **Deploy New Version** task is processing on servers jonas1, jonas2, and jonas3

When all values of status column are **Waiting**, it means that the current task is successfully completed and now the user is able to click to the **Next** button to execute the next task or click on **Undo** button to undo the last task.

Figure 3.11. The migration execution page: Deployment error

The status column in the table above represents the state of a task (sub-step) execution:

- **Waiting**: The server is waiting for the other servers to finish. If all servers are **Waiting**, it means that the task is completed on all servers, therefore that you need to click on the **Next** button in order to execute the next task.
- **Error!**: A problem prevents the execution of a task.
- **Running**: The task is executing

The **View Log** button pops up a window that shows all information about all executions on that server (errors and other logs).

When an error occurs, two new buttons will appear in the action column:

- **Retry server**: Executes the same action again on that server.
- **Ignore server**: Ignores the task for that server. That button is useful if the administrator has fixed the problem manually and wants to inform JaDOrT about it.

Since an old version of the same application was deployed on that server, the second step (undeploy and erase old version) is also executed.

Figure 3.12. The migration execution page: Undeployment

3.7. Maintenance

The server maintenance is designed for maintaining the application server itself (this may also include the applications running on the server). Use this option in order to upgrade your application server, do a physical maintenance on the server and/or change application server.

JaDOrT allows the maintenance on a group of JOnAS, Glassfish, JBoss, WebLogic and/or WebSphere servers. In that case, JaDOrT handles the session and load management.

JaDOrT supports servers that are deployed physically on a machine as well as servers on virtual machines.

3.7.1. Servers selection

JaDOrT supports maintenance in groups of servers. For example, you can maintain the first 3 servers, then on the following 10 servers, and so on; in order to guarantee a certain quality of service or for pre-production testing and validation purposes.

Figure 3.13. Servers selection page

You have two ways to select servers:

- **Manual selection**: by checking the check box of the correspondent server.
- **Automatic selection**: by Clicking the **Click here** button at the bottom of the servers list. JaDOrT will then select all servers that can be maintained right now (based on capacity check).

JaDOrT displays three important informations:

- Each server's capacity
- Number of active session on each server
- Maintenance status: **Maintained** or **Not Maintained**

These information assist the choice of the server(s) to maintain.

Once some servers are selected, click the **Next** button in the bottom panel in order to continue.

Note

JaDOrT will come back to this screen after every **Maintenance execution** iteration, as long as there are non-maintained servers remaining.

3.7.2. VM image selection

If you're maintaining Virtual Machines, JaDOrT will display the VM Images that can be deployed.

Figure 3.14. VM Image Selection page

3.7.3. Servers maintenance execution

Figure 3.15. The servers maintenance execution page

In the figure above, you can find the following information:

- Current global task in the migration process is **Stop Server**
- **Stop Server** task is processing on servers jonas1, jonas2, and jonas3 + workers worker1, worker2, and worker3

When all values of status column are **Waiting**, it means that the current task is successfully completed and now the user is able to click to the **Next** button to execute the next task or click on **Undo** button to undo the last task.

The status column in the table above represents the state of a task (sub-step) execution:

- **Waiting**: The server is waiting for the other servers to finish. If all servers are **Waiting**, it means that the task is completed on all servers, therefore that you need to click on the **Next** button in order to execute the next task.
- **Error!**: A problem prevents the execution of a task.
- **Running**: The task is executing

The **View Log** button pops up a window that shows all information about all executions on that server (errors and other logs).

When an error occurs, two new buttons will appear in the action column:

- **Retry server**: Executes the same action again on that server.
- **Ignore server**: Ignores the task for that server. That button is useful if the administrator has fixed the problem manually and wants to inform JaDOrT about it.

Chapter 4. Command line client

4.1. Goal

The aim of the command line client is to script your deployment execution. It provides the same functionalities as the Web interface, but it will be the script that will coordinate the progress of your operation. The client can also be wrapped into any other program.

The command line client is stateless: this means that you need to specify the operation identifier at each command, and that you can only execute one action at each call.

4.2. Available commands

The commands are as follows:

4.2.1. Operation management

- ***getOperationsList*** *:* print the operation list.

  Prints : the existing operations. For each: id, name, creation date. Each operation is separated by end of line, each record by a tab.

  Return value : none.

- ***deleteOperation*** *{operationId}* : delete the operation with id `operationId`.

  Prints : none.

  Return value : none.

- ***createNewOperation*** *{operationName}* : create a new operation with `operationName` as name.

  Prints : The Id of the created operation.

  Return value : the operation id of the new operation (integer).

4.2.2. Basic actions

- ***getCurrentStep*** *{operationId}* : get the current progress step for the operation with the given `operationId`.

  Prints : Name of the current step.

  Return value : none.

4.2.3. Operation initialization

- ***loadTopology*** *{operationId} {filePath}* : load the topology contains in the file `filePath` in the operation with id `operationId`. The file path is a URL, it can be local or remote.

  Prints : none.

Return value : none.

4.2.4. Group view and selection

- ***getGroups*** *{operationId}* : print the groups of the operation with id `operationId`.

  Prints : For each group: id, name, server list, application list. Each group is separated by end of line, each record by tab character.

  Return value : none.

- ***selectGroup*** *{operationId} {groupName}* : select the servers group named `groupName` on which the operation will be executed.

  Prints : none.

  Return value : none.

- ***selectOperationType*** *{operationId} {operationType}* : select the operation type: "`MIGRATE`" or "`MAINTAIN`".

  Prints : none.

  Return value : none.

- ***getSelectedGroup*** *{operationId}* : print the selected group of the operation `operationId`.

  Prints : id, name, server list, application list of the selected group. Each record is separated by tab character.

  Return value : none.

4.2.5. Operation configuration

- ***selectApplication*** *{operationId} {archivePath}* : select the archive located at `archivePath` as the application to deploy for the operation `operationId`. The archive path is a URL, it can be local or remote.

  Prints : none.

  Return value : none.

- ***getVMImages*** *{operationId}* : print the VM images available on the VMM managers of the operation with id `operationId`.

  Prints : For each VM image: id, name, uuid. Each VM image is separated by end of line, each record by tab character.

  Return value : none.

- ***selectVMImage*** *{operationId} {vmImageName}* : select the VM image that will be deployed on the VMs that are part of the maintenance operation.

  Prints : none.

  Return value : none.

- ***selectServers*** *{operationId} {serverName} ... {serverName}* : select the servers to maintain.

  Prints : none.

Return value : none.

- ***selectVMImageForServer*** *{operationId} {vmImageName} {serverName}* : select the VM image that will be deployed on the VM for the server `serverName`.

  Prints : none.

  Return value : none.

4.2.6. Operation execution

- ***canGoToNextStep*** *{operationId}* : is the operation `operationId` ready to go to next step ?

  Prints : `true` or `false`.

  Return value : 1 if true, 0 if false.

- ***next {operationId}*** : to go to the next step of the operation which has id `operationId`.

  Prints : none.

  Return value : none.

- ***canGoToPreviousStep*** *{operationId}* : is the operation `operationId` ready to go to previous step ?

  Prints : `true` or `false`.

  Return value : 1 if true, 0 if false.

- ***previous*** *{operationId}* : to go to the next previous of the operation `operationId`.

  Prints : none.

  Return value : none.

- ***getServerProgressList*** *{operationId} :* print the list of the server progress tracking beans.

  Prints : For each progress manager: id, server name, state, progress. Each deployment is separated by a line break and each record by a tab character.

  Return value : none.

- ***getWorkerProgressList*** *{operationId} :* print the list of the worker progress tracking beans.

  Prints : For each progress manager: id, server name, state, progress. Each deployment is separated by a line break and each record by a tab character.

  Return value : none.

- ***abortServer*** *{operationId} {serverName}* : aborts the operation that's currently in progress for the server named `serverName`.

  Prints : none.

  Return value : none.

- ***restartServer*** *{operationId} {serverName}* : restart the deployment operation on the server named `serverName`. This function has to be used when the last deployment operation hasn’t be successful (deployment state = error).

Prints : none.

Return value : none.

- ***checkServer*** *{operationId} {serverName}* : checks if the error of the server `serverName` is resolved. You can for example call this method once you've resolved the issue with the server by hand, to let JaDOrT check your fix. It is highly recommended to call this method before ignoring any errors.

  Prints : `true` if server is OK, `false` otherwise.

  Return value : 1 if server is OK, 0 otherwise.

- ***ignoreServer*** *{operationId} {serverName}* : ignore the last deployment operation on the server named `serverName`. This function has to be used when the last deployment operation hasn't be successful (deployment state = error).

  Prints : none.

  Return value : none.

- ***abortWorker*** *{operationId} {workerName}* : aborts the operation that's currently in progress for the worker named `workerName`.

  Prints : none.

  Return value : none.

- ***restartWorker*** *{operationId} {workerName}* : restart the deployment operation on the worker named `workerName`. This function has to be used when the last deployment operation hasn't be successful (deployment state = error).

  Prints : none.

  Return value : none.

- ***checkWorker*** *{operationId} {workerName}* : checks if the error of the worker `workerName` is resolved. You can for example call this method once you've resolved the issue with the worker by hand, to let JaDOrT check your fix. It is highly recommended to call this method before ignoring any errors.

  Prints : `true` if worker is OK, `false` otherwise.

  Return value : 1 if worker is OK, 0 otherwise.

- ***ignoreWorker*** *{operationId} {workerName}* : ignore the last deployment operation on the worker named `workerName`. This function has to be used when the last deployment operation hasn't be successful (deployment state = error).

  Prints : none.

  Return value : none.

4.3. Script examples

Here an example done using a Windows batch. Its use is limited since Windows batch cannot check for the return values of applications or read output.

```
rem the JaDOrT Console JAR command, change the path as needed
set cmd=java -jar target\jadort-console-jar-with-dependencies.jar
```

```
%cmd% createNewOperation test_migrate

rem use 1 as operationId because we are not able to put the return of the last command on a
 local variable
%cmd% loadTopology 1 "%~dp0\..\jadort-documentation-and-samples\src\main\resources\Topology
 with JOnAS servers.xml"
%cmd% getGroups 1
%cmd% selectGroup 1 "xyleme-MyStore"
%cmd% selectOperationType 1 "MIGRATE"
%cmd% getSelectedGroup 1
%cmd% createApplication 1 "%~dp0\..\jadort-documentation-and-samples\src\main\resources
\MyStore\version-1.0.0\MyStore.ear"

rem Go to the first migration step: upload application on each server
%cmd% next 1
%cmd% getServerProgressList 1

rem You have to wait for ' canGoNextToNextStep 1 == 1" (1 = true)
pause

rem Go to the second migration step: deploy the new application
%cmd% next 1
%cmd% getServerProgressList 1
```

Chapter 5. Troubleshouting

While using JaDOrT, some unexpected errors may occur. As with normal (i.e. user-caused) errors, these are displayed as errors on the interface. This section details the unexpected error messages that can be displayed by the Web interface.

Error "Bean has been removed"

If you get this error, you must close and re-open your browser. Note that JaDOrT can always resume your operation.

Error "object Object"

This message means that JaDOrT has been undeployed from the application server. Please check the error’s cause with the system administrator.

Other errors

For other error messages that are application bugs, the detailed error log is output on the application server.