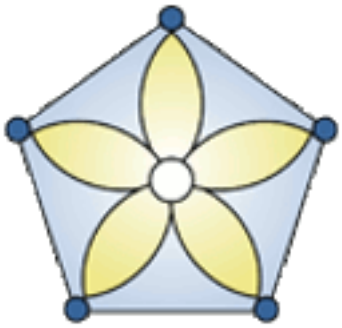


JaDOrT - User's guide

- -

Table of Contents

List of Figures

Chapter 1. Introduction

1.1. Purpose

The purpose of this user guide is to provide general information and usage instructions to the future users of the **JASMINe Deployment Orchestration Tool** (JaDOrT).

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. What the tool does

1.3.1. JOnAS versioning service

This service has been designed for dynamic redeployment of applications, without any application downtime and without users' sessions being lost:

- Deployment of a new version of an application does not require the undeployment of any previous version.
- Users that were on a previous version keep on using that version as long as their session on that version is active (for example, as long as they have not finished buying items on the previous version of the online trade web site). This guarantees that no user data will be lost, and that if there is any problem with the new version the old version is still available instantly.
- New versions of the application can be deployed using various strategies, for instance allow testing of the new version by a small community to ensure its readiness for production.

The versioning service achieves this by adding virtual contexts to all services that provide support for versioning. To use the versioning service:

1. Enable the versioning service in jonas.properties
2. Define the **Implementation-Version** attribute in your deployable file's (whether war, jar or ear) MANIFEST. Note that:
   - ANT, Maven as well as most IDEs can set any MANIFEST attribute automatically.
   - If the archive that will be deployed is an ear, the **Implementation-Version** defined in the MANIFEST of the ear will be used for all archives the ear contains.

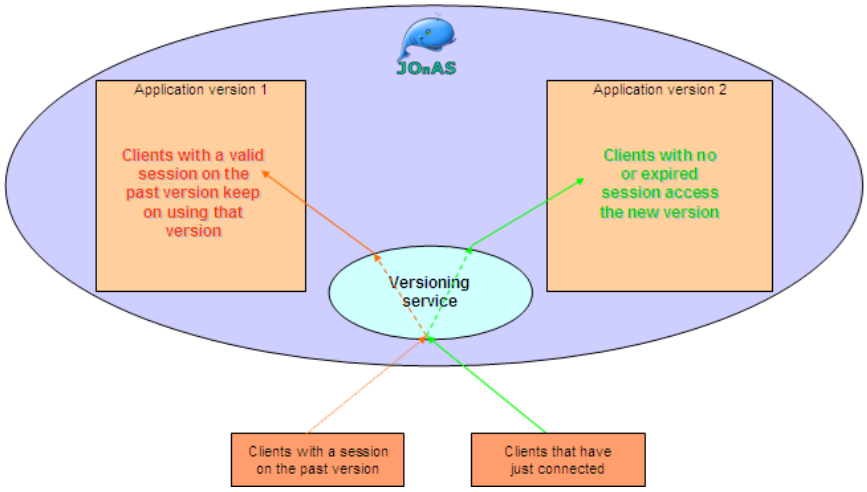
When the versioning service is enabled, application resources (web pages, EJBs, etc.) are accessed the following way:

- Each versioned application has a user (virtual) address. Each version of an application is renamed and bound to that virtual address. Each bound version has a policy (that can be changed in time in order to manage the deployment strategy):

  - **Private**: Can only be accessed by clients that satisfy some prerequisites; for example belong to a certain IP address group or provide a certain credential.
  - **Reserved**: Not accessible using the virtual address, therefore can only be accessed directly (using the versioned address).
  - **Disabled**: Only accessible by clients that have been using this version in the past (until their session expires). This guarantees that users will not lose their session data during redeployment.
  - **Default**: Version accessed by all clients that don't fit in any other policy.
- A user can access the application resource indirectly (using the virtual address) or directly (using the versioned address).
- If the user tries to access the application resource indirectly (using the virtual address), the versioning system:
  - First checks if that user is defined as a user that can access a version of the application with the **Private** access policy. If that is the case, the user is routed to that private version of the application.
  - Then checks if that user already has a session in a version of the application with the **Disabled** access policy. If that is the case, the user is routed to that disabled version of the application.
  - If neither of these cases is true, routes the user to the version of the application with the **Default** access policy. If the application does not define any default version, the user will see *"resource not found"* message.

This can be schematized as follows:

Figure 1.1. JOnAS versioning service



As this service allows seamless and interruption less upgrade and test of all applications, it is strongly recommended for all applications to refer version identifiers in their manifest files.

1.3.2. Orchestration

Version migration or maintenance within a set of servers (cluster or farm) must be orchestrated to ensure a good execution or a return to the initial situation.

JaDOrT:

- Gives a global view about the version migration or deployment process
  - Sets of servers and their members
  - Applications deployed on each set
- Automates all tasks
- Lets the administrator take control of the situation at any time
  - Cancel any step
  - Fix any problem manually (and tell the tool 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 or the average deploy time.

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 an operation will be doing actions on. This description includes:

- Application server (AS) instances:
  - Their configuration (cluster or farm)
  - Their JMX connectors
  - Their logical domains
  - Their daemon instances (typically used for starting and stopping the servers)
- Load balancer (LB) instances:
  - Their connectors
  - The link between LBs and ASs; i.e. the list of workers on a given LB redirecting requests to an AS.
- Virtual Machine Manager (VMM) instances:
  - Their connectors
  - The link between VMMs and ASs; i.e. which AS is running on which VMM 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 can also use JASMINe Design [http://jasmine.ow2.org/] 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 ?

It is the migration execution operation; it takes place in five sub steps:

- **Download 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.
- **Undeploy Old Version**: Undeploys the old version of the application, which results in the old version becoming inaccessible.
- **Erase Old Version**: Erases the old version of the application from the application server.

Important

you can not go back to the previous step once the "erase old version" task has been initiated.

Note

If the deployed application does not have any old version present, the two last steps are ignored.

2.4. What is a maintenance ?

It is the updating of servers or VMs operation. 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. It can be used for any kind of maintenance (server, VM).

It takes place in many sub steps:

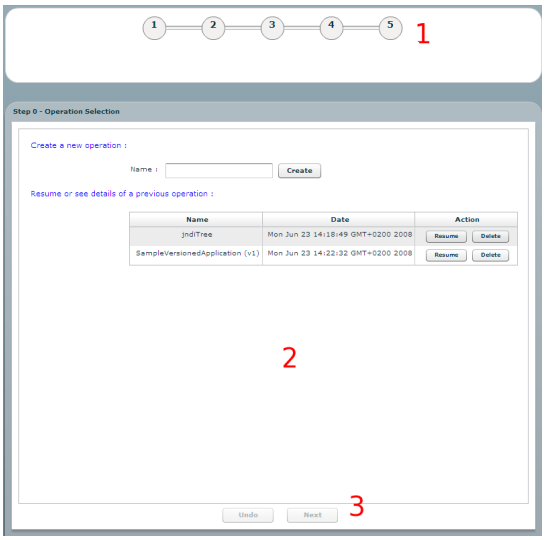
- **Disable Application**: The application on the server will be only accessible by clients that have been using this version in the past (until their session expires). This guarantees that users will not lose their session data during redeployment.
- **Stop worker**: Stop the worker(s) that associated with this server. Once this is done, the load balancer is expected not to route any requests to this server.
- **Stop server**: Stop the server or the virtual machine.
- **Maintenance**: Update the server instance. We clearly differenciate 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 deploy the specified VM Image on this host (therefore initializes the hard drives).
- **Start server** : Start the server or virtual machine.
- **Start worker**: Start the worker(s) that associated with this server. Once this is done, the load balancer is expected to start routing requests to this server.

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

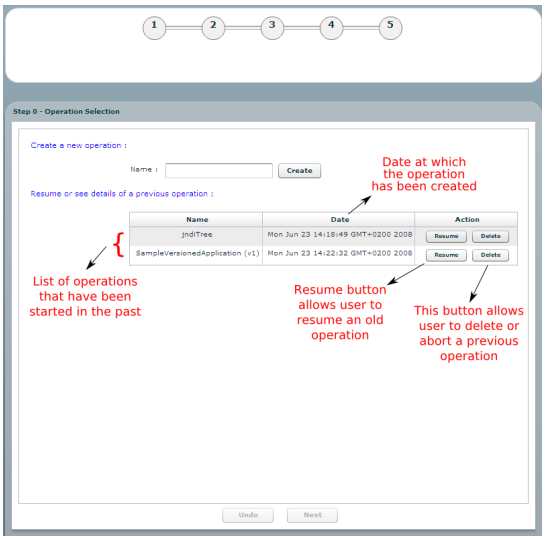


1. **Progress Step Panel** : this panel indicates the current step and the overall progress.
2. **Main Panel** : this panel depends on the current step and allows you the user to execute tasks.
3. **Button Panel** : this panel is used for navigation between steps.

3.2. Operation selection

The following diagram describes the Home Page of the tool.

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, simply click on the "Resume" button in the **"Action"** column of the table. When this button is clicked, the tool loads the operation data from the database and resumes the operation from where has been stopped. The operation then continues its execution.

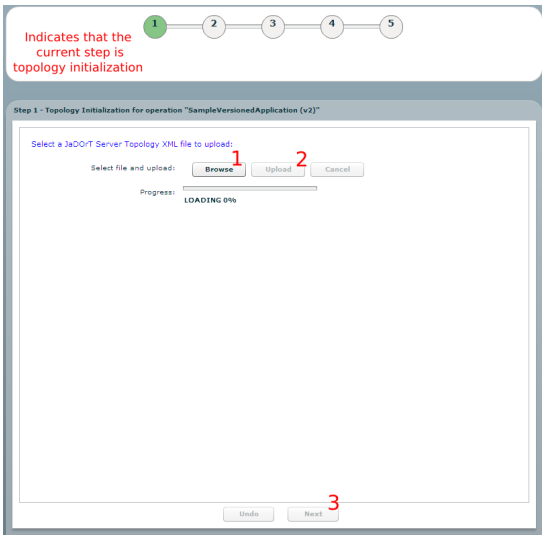
To delete or abort an operation, click the **"Delete"** button. This deletes all data associated with that operation (including all deployment logs).

Note

A previous operation is an operation that has been started by a user in the past and maybe has not been completed yet. For all operations, all logs are persisted for later analysis.

3.3. Topology initialization

Figure 3.3. Operation selection page



To initialize a topology the user must upload an .xml file that describes the topology (groups of servers, server info, JMX URL …).

The user must :

1. Select the topology XML file,
2. Click on the **"Upload"** button to load the topology
3. After loading the topology the button **"Next"** becomes enabled, and the user can click on it to go the next step.

Here is an example of topology XML file :

Figure 3.4. Example of a topology.xml file

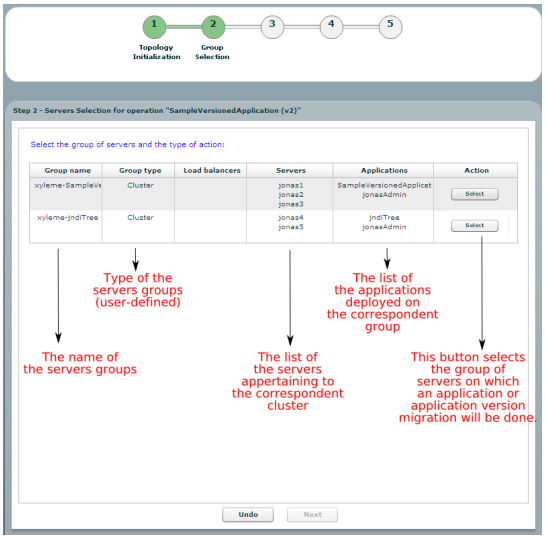
```
<?xml version="1.0" encoding="UTF-8" ?>
- <topology>
  - <group name="xyleme-SampleVersionedApplication" type="Cluster">
    - <servers>
      - <server name="jonas1" capacity="100">
          <JMXConnector URL="service:jmx:rmi://xyleme/jndi/rmi://xyleme:11099/jrmpconnector_jonas1" />
        </server>
      - <server name="jonas2" capacity="100">
          <JMXConnector URL="service:jmx:rmi://xyleme/jndi/rmi://xyleme:21099/jrmpconnector_jonas2" />
        </server>
      - <server name="jonas3" capacity="100">
          <JMXConnector URL="service:jmx:rmi://xyleme/jndi/rmi://xyleme:31099/jrmpconnector_jonas3" />
        </server>
      </servers>
    </group>
  - <group name="xyleme-jndiTree" type="Cluster">
    - <servers>
      - <server name="jonas4" capacity="100">
          <JMXConnector URL="service:jmx:rmi://xyleme/jndi/rmi://xyleme:41099/jrmpconnector_jonas4" />
        </server>
      - <server name="jonas5" capacity="100">
          <JMXConnector URL="service:jmx:rmi://xyleme/jndi/rmi://xyleme:51099/jrmpconnector_jonas5" />
        </server>
      </servers>
    </group>
  </topology>
```

You will see the result of this file in 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. Servers selection page



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

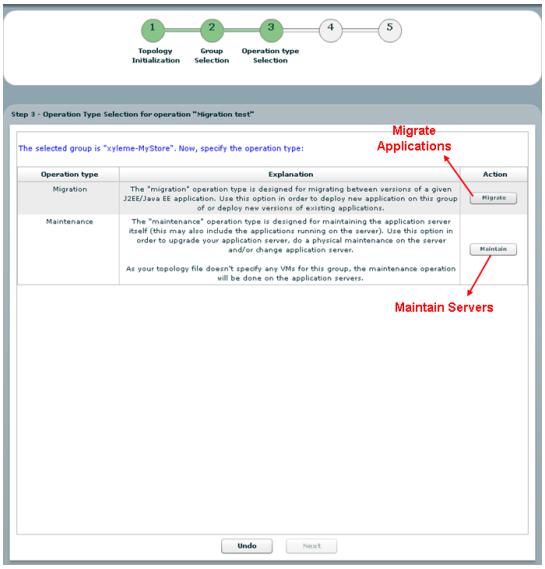
Note

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. Before reaching this step, JaDOrT will also verify that all servers, worker and VMs specified in the topology XML file are accessible.

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

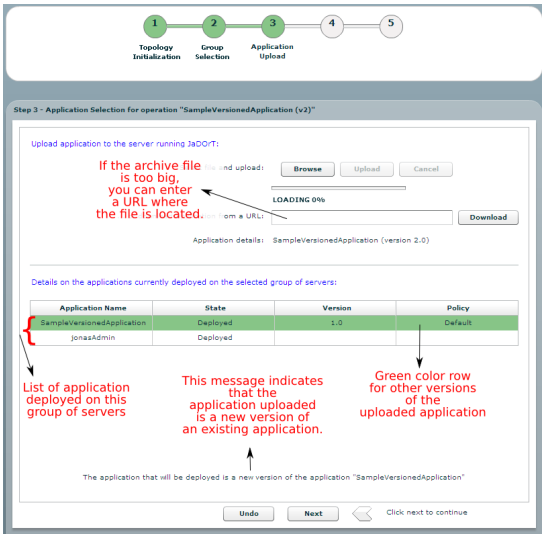


3.6. Migration

JaDOrT allows the deployment of applications on a group of JOnAS 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 size of the archive file is too big (the HTTP protocol can not upload files bigger than 4 MB), 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. Migration execution

Figure 3.8. The migration execution page

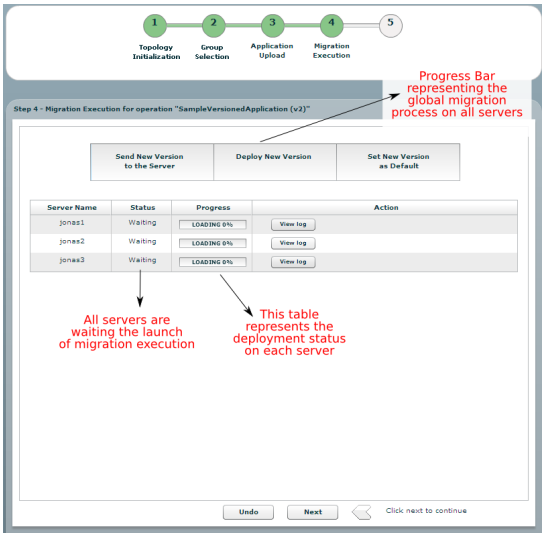
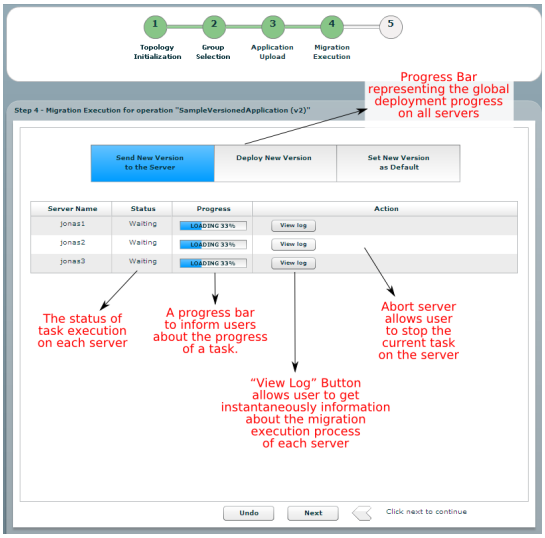


Figure 3.9. The migration execution page : Upload new version

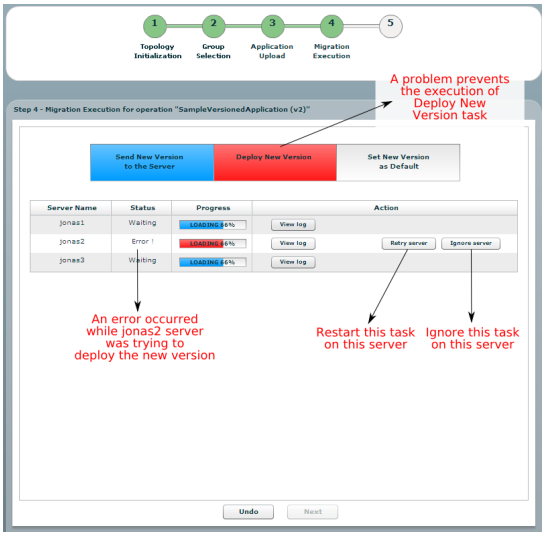


In the figure above, we found the following information :

- 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 3.10. 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.
- **Done OK**: Migration has been finished on all servers. The only action you can then take is to view logs.
- **Error!**: A problem prevents the execution of a task.
- **Running**: The task is executing

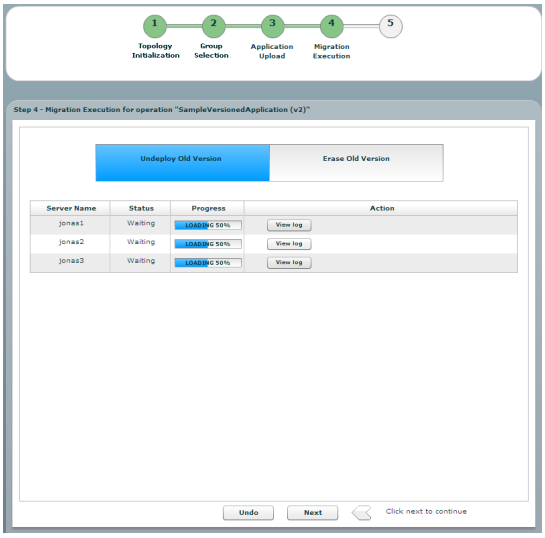
The **"View Log"** button pops up a window that shows all information about all executions on that server (errors and other logs).

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 3.11. The migration execution page : Undeployment



3.7. Maintenance

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.

It can be used for maintaining application servers or Virtual Machines(VM).

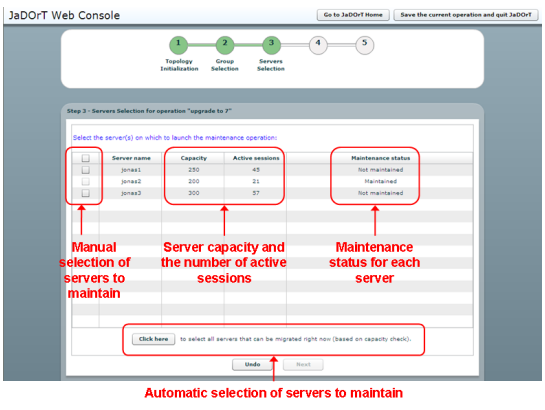
3.7.1. Servers Maintenance

The Servers 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

3.7.1.1. Servers Selections

In order to maintain Servers, you must firstly select the Servers that will be maintained.

Figure 3.12. 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).

In the figure above, we found the following information :

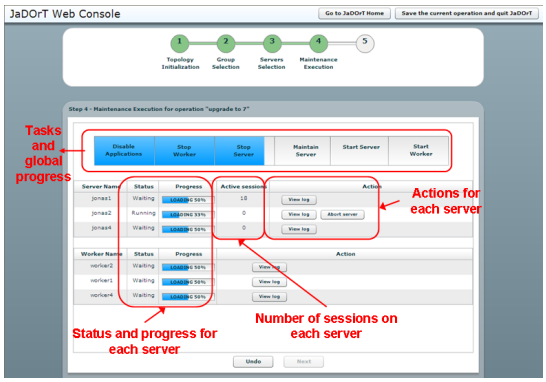
- Server's capacities
- Number of active session on each server
- Maintenance status: Maintained or Not Maintained

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

To start the servers maintenance execution, click the "Next" button in the bottom panel.

3.7.1.2. Servers maintenance execution

Figure 3.13. The servers maintenance execution page



It takes place in many sub steps:

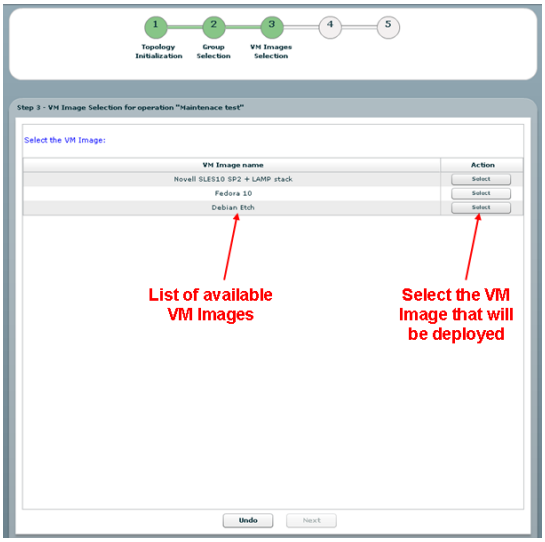
- **Disable worker**: Disable the associated worker on the load balancer. Once this is done, no new clients will get connected to this server.
- **Disable applications**: All applications on the server get disabled. As a result, only non-expired sessions will be allowed on the server.
- **Stop server**: Stop the server.
- **Maintain server**: Migrate the server to the new infrastructure.
- **Start server** : Start the new server instance.
- **Enable worker**: Enable the associated worker on the load balancer. Once this is done, this server becomes accessible to all clients.

3.7.2. VMs Maintenance

JaDOrT can be used for maintaining Virtual Machines(VM).

To maintain VM, you need to select the VM Image that will be deployed

Figure 3.14. VM Image Selection page



The following steps in the VM maintenance are the same as those of the servers maintenance.

The only difference is in the servers maintenance execution:

- **Stop server**: Stops the server and the VM.
- **Maintain server**: Deploy the selected VM Image. VM images can be selected for the whole group or on a per-server basis.
- **Start server** : Start the new VM and the server.

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-1.3.0-SNAPSHOT-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
```

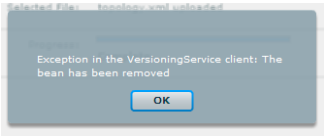
```
%cmd% loadTopology 1 "%~dp0\..\jadort-documentation-and-samples\src\main\resources\Topology
 with JOnAS servers.xml"
%cmd% getGroups 1
%cmd% selectGroup 1 "xyleme-MyStore"
%cmd% selectOperationType 1 "MIGRATE"
%cmd% getSelectedGroup 1
%cmd% createApplication 1 "%~dp0\..\jadort-documentation-and-samples\src\main\resources
\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.

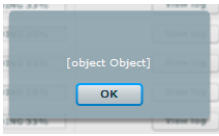
Error "Bean has been removed"

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.