Manual Tests Suite for Production Redeployment

Authors : seymi.hafsia@serli.com, rym.jabeur@serli.com
Reviewer : chris.woodrow@serli.com
Version : **3**
Release date : **2013/06/04**

This document describes all the functional tests that have been made during the development of Production Redeployment functionality on the Glassfish server.

These tests, for now done by hand, are intended to be automated and added to the DevTests and QuickLooks Tests.

First part is dedicated to validation tests, their goal is to checkout the production redeployment functionality we've achieved. This part also contains non-regression tests over standard glassfish functionalities.

Part II focuses on testing production redeployment's specific functionalities.

Part III is basically composed by the same tests as Part II in cluster environment.

Finally, Part IV relates to error handling tests.

Note: All the functional tests that are described below pass correctly on our current version.

Testing Protocol and conventions

Testing protocol

Each scenario contains :

- **Test case**
- **Scenario's description**
- **Commands and steps**
- **Observed results**

Scenarios are designed to be independant of each other.
Each scenario starts with a blank application and the test fixture is set up previously to the scenario's process.

Conventions

A set of color conventions is used in the above document :
- The green parts of code are the observed results after running **asadmin list-applications -l**.
- The red ones are error displayed on standard output.
- Lines in blue beginning with '**#**' are comments.

Part I : Basic commands

*In this part, the objective is to test the normal behavior of Glassfish versioning basic commands (**deploy**, **enable**, **disable**, **undeploy**), to check that we didn't altered their used.*

Scenario 1 : deploy

Only one active version for a single application.
We deploy two versions of a single application. Only the last deployed version may be active.

Scenario description

Deploy v1.0 of foo
Check v1.0 is deployed
Deploy v1.1 of foo
Check v1.1 is deployed and active and v1.0 is disabled

Execution :

```
asadmin deploy --name foo:1.0 foo.war
asadmin deploy --name foo:1.1 foo.war
asadmin list-applications -l
```

Observed Result :

```
NAME       TYPE      STATUS    EXTENDED_STATUS   RETIRES_ON
foo:1.0    web       disabled
```

```
foo:1.1       web         enable       active
```

Scenario 2 : deploy 2 applications

An application can only have one active version at the same time.

Deploy two distincts applications, use versioning, and check every application has only one active version

Scenario description :

Deploy v1.0 of foo
Deploy v1.0 of newfoo
Deploy v1.1 of foo
Check that foo:v1.1 and newfoo:v1.0 are deployed and active and that v1.0 is disabled

Execution :

```
asadmin deploy --name foo:1.0 foo.war
asadmin deploy --name newfoo:1.0 newfoo.war
asadmin deploy --name foo:1.1 foo.war
asadmin list-applications -l
```

Observed Result :

```
NAME        TYPE        STATUS    EXTENDED_STATUS   RETIRES_ON
foo:1.0     web         disabled
foo:1.1     web         enable       active
newfoo:1.1  web         enable       active
```

Scenario 3 : enable

Enabling of a previous version.

We want to test previous version of an application reenabling. After enabling foo:1.0 , foo:1.1 is disabled.

Scenario description :

Deploy v1.0 of foo
Deploy v1.1 of foo
enable v1.0 of foo
Check v1.0 is deployed and active and that v1.0 is disabled

Execution :

```
asadmin deploy --name foo:1.0 foo.war
asadmin deploy --name foo:1.1 foo.war
asadmin enable foo:1.0
asadmin list-applications -l
```

Observed Result :

```
NAME        TYPE        STATUS    EXTENDED_STATUS   RETIRES_ON
foo:1.0     web         enable       active
```

```
foo:1.1        web          disabled
```

Scenario 4 : disable

Disabling an application.

Disable the current version of an application, given application has no more active version.

Scenario description :

Deploy v1.0 of foo
Deploy v1.1 of foo
Enable v1.0 of foo (disables v1.1)
Disable v1.0
Check both versions 1.0 and 1.1 are disabled

Execution :

```
asadmin deploy --name foo:1.0 foo.war
asadmin deploy --name foo:1.1 foo.war
asadmin enable foo:1.0
asadmin disable foo:1.0
asadmin list-applications -l
```

Observed Result :

```
NAME        TYPE       STATUS    EXTENDED_STATUS   RETIRES_ON
foo:1.0      web        disabled
foo:1.1      web        disabled
```

Scenario 5 : undeploy

Undeploy a versioned application.

Deploying a single version of an application and undeploying it.

Scenario description :

Deploy v1.0 of foo
Deploy v1.0 of newfoo
Deploy v1.1 of foo
Undeploy v1.0 of newfoo
Checkout v1.1 of foo is still active, v1.0 of foo is still inactive and v1.0 of newfoo is undeployed

Execution :

```
asadmin deploy --name foo:1.0 foo.war
asadmin deploy --name newfoo:1.0 newfoo.war
asadmin deploy --name foo:1.1 foo.war
asadmin undeploy newfoo:1.0
asadmin list-applications -l
```

Observed Result :

```
NAME        TYPE       STATUS    EXTENDED_STATUS   RETIRES_ON
foo:1.0     web        disabled
foo:1.1     web        enable       active
```

Scenario 6 : undeploy all

Undeploy all the versions of an application.

After deploying two consecutive versions of an application, we now undeploy all the versions of the application.

Scenario description :

Deploy v1.0 of foo
Deploy v1.0 of newfoo
Deplpoy v1.1 of foo
Undeploy all versions of foo
Checkout v1.1 and v1.0 of foo have been undeployed and v1.0 of newfoo is still deployed

Execution :

```
asadmin deploy --name foo:1.0 foo.war
asadmin deploy --name newfoo:1.0 newfoo.war
asadmin deploy --name foo:1.1 foo.war
asadmin undeploy foo:*
asadmin list-applications -l
```

Observed Result :

```
NAME        TYPE       STATUS    EXTENDED_STATUS   RETIRES_ON
newfoo:1.1  web        enable      active
```

Part II : Production Redeployment

These are the test scenarios of Production Redeployment

Scenario 7 : deploy with fixed retiretimeout

Deploy a new version of a versioned application with a retiretimeout of 10 seconds.

Scenario description :

Deploy vbeta of foo
Deploy v1.0 of foo with retiretimeout of 10 seconds
Check that v1.0 is deployed and that vbeta is retiring
# More than 10 seconds later
Check that vbeta of foo is inactive

Execution :

```
asadmin deploy --name foo:beta foo.war
asadmin deploy --retiretimeout=10 --name foo:1.0 foo.war
asadmin list-applications -l
# More than 10 seconds later
asadmin list-applications -l
```

Observed Result :

```
NAME       TYPE      STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta   web       enable    retired                      [current_time + 10 sec]
foo:1.0    web       enable    active
# After 10 seconds, foo is disabled.
NAME       TYPE      STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta   web       disabled
foo:1.0    web       enable    active
```

Scenario 8 : enable with fixed retire timeout

Enabling previous version of an application with a retire timeout of 10 seconds.

Scenario description :

Deploy foo vbeta
Deploy foo v1.0
Enable foo vbeta with a retire timeout of 10 seconds
Check that v1.0 is retiring and that vbeta is active
#After 10 seconds
Check that v1.0 is disabled

Execution :

```
asadmin deploy --name foo:beta foo.war
asadmin deploy --name foo:1.0 foo.war
asadmin enable --retiretimeout=10 foo:beta
asadmin list-applications -l
# More than 10 seconds later
asadmin list-applications -l
```

Observed Result :

```
NAME       TYPE      STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta   web       enable    active
foo:1.0    web       enable    retired                      [current_time + 10 sec]

# More than 10 seconds later, second asadmin list-applications -l
NAME       TYPE      STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta   web       enable    active
foo:1.0    web       disabled
```

Scenario 9 : Retired version reactivation

Reenable of a retiring version. Deploy two consecutive versions of an application using production redeployment, before original version is deactivated, enable retired version.

Scenario description :

Deploy vbeta of foo
Deploy v1.0 of foo with a retire timeout of 10 seconds
Enable vbeta of foo with a retire timeout of 10 seconds
# More than 10 seconds later
Check v1.0 is disabled

Execution :

```
asadmin deploy --name foo:beta foo.war
asadmin deploy --retiretimeout=10 --name foo:1.0 foo.war
# foo:beta is in retirement.
asadmin enable --retiretimeout=10 foo:beta
asadmin list-applications -l
# After 10 seconds, foo:1.0 is disabled.
asadmin list-applications -l
```

Observed Result :

```
NAME       TYPE     STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta   web      enable    active
foo:1.0    web      enable    retired                [current_time + 10 sec]
# After 10 seconds, foo:1.0 is disabled.
NAME       TYPE     STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta   web      enable    active
foo:1.0    web      disabled
```

Scenario 10 : Request Redirection

Requests redirection test. When two versions of the same applications are deployed at the same time using Production Redeployment, pre existing sessions (prior to new version deployment) requests must be directed to retired version meanwhile new requests will be directed to new version.

Scenario description :

Deploy vbeta of foo
Open a session on foo (sess1)
Deploy v1.0 of foo with retiretimeout = 10
Open a session on foo (sess2)
Checkout that **sess1** requests are directed to foo vbeta while **sess2** requests are directed to foo v1.0

```
# After 10 seconds
Checkout that vbeta is disabled
Checkout that sess1 requests are directed to foov1.0
```

Execution :

```
asadmin deploy --name foo:beta foo.war
# Open a session (called sess1 for the example).
asadmin deploy --retiretimeout=10 --name foo:1.0 foo.war
asadmin list-applications -l
# Open a second session (called sess2 for the example).
# Check requests on sess1 et sess2
# After 10 seconds
asadmin list-applications -l
# Check requests on sess1 et sess2
```

Observed Result :

```
NAME        TYPE      STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta    web       enable    retired                       [current_time + 10 sec]
foo:1.0     web       enable    active
# Check that sess1 is still redirected on foo:beta and sess2 on foo:1.0 .
# After 10 seconds, foo:beta is disabled. Check that new request with sess1 are
# redirected on foo:1.0 .
NAME        TYPE      STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta    web       disabled
foo:1.0     web       enable    active
```

Scenario 11 : deploy with “All session expired” option

Sessions expiration on retired version test. We retire a version with retiretimeout set to -1, and then need to test if retired version is effectively disabled when all sessions that have been opened on previous version have expired.

Scenario description :

Deploy vbeta of foo
Start a session (sess1) on foo(vbeta)
Deploy v1.0 of foo with retiretimeout = -1
Start a session (sess2) on foo(v1.0)
Check that sess2 requests are correctly directed to v1.0
Check that v1.0 of foo is deployed and vbeta is retiring (expecting all sessions on vbeta to expire)
Wait for session expiration
Check that vbeta of foo is disabled

Execution :

```
asadmin deploy --name foo:beta foo.war
```

```
 # Start a session (sess1)
 asadmin deploy --retiretimeout=-1 --name foo:1.0 foo.war
asadmin list-applications -l
# wait for sess1 to expire.
asadmin list-applications -l
```

Observed Result :

```
NAME       TYPE     STATUS    EXTENDED_STATUS  RETIRES_ON
foo:beta   web      enable    retired            All session expired
foo:1.0    web      enable    active
# wait for sess1 to expire
NAME       TYPE     STATUS    EXTENDED_STATUS  RETIRES_ON
foo:beta   web      disabled
foo:1.0    web      enable    active
```

Scenario 12 : enable with “All session expired” option

Version enable with “All session expired” option. Previous version is enabled with the “All session expired”, previous version is disabled when all sessions that have been previously created are expired or finished.

Scenario description :

Deploy vbeta of foo
Deploy v1.0 of foo (disables vbeta)
Open session (sess1) on foo (v1.0 at this time)
Enable vbeta with retiretimeout = -1
Open session (sess2) on foo (vbeta at this time)
Checkout that vbeta is deployed and v1.0 is retiring
Checkout that sess2 requests are correctly directed to vbeta
Expect sess1 expiration
Checkout that is disabled

Execution :

```
asadmin deploy --name foo:beta foo.war
asadmin deploy --name foo:1.0 foo.war
# Start a session (called sess1 for the example)
asadmin enable --retiretimeout=-1 foo:beta
asadmin list-applications -l
 # wait for sess1 to expire
asadmin list-applications -l
```

Observed Result :

```
NAME       TYPE     STATUS    EXTENDED_STATUS  RETIRES_ON
foo:beta   web      enable    active
foo:1.0    web      enable    retired            All session expired
```

```
# wait for sess1 to expire
NAME        TYPE       STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta     web         enable      active
foo:1.0      web         disabled
```

Part III : Production Redeployment in cluster mode

These are the test cases of Production Redeployment in cluster mode. We've created a cluster “cluster1” with a single instance “instance1”.
Scenarios are the same as previous ones, scenario N-cluster refers to N-equivalent scenario in part II.

Scenario 7-cluster

Execution :

```
asadmin deploy --name --target cluster1 foo:beta foo.war
asadmin deploy --retiretimeout=10 --name foo:1.0 --target cluster1 foo.war
asadmin list-applications -l cluster1
# More than 10 seconds later
asadmin list-applications -l cluster1
```

Observed Result :

```
NAME        TYPE       STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta     web         enable      retired                       [current_time + 10 sec]
foo:1.0      web         enable      active
# After 10 seconds, foo:vbeta  is disabled.
NAME        TYPE       STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta     web         disabled
foo:1.0      web         enable      active
```

Scenario 8-cluster

Execution :

```
asadmin deploy --name foo:beta  --target cluster1 foo.war
asadmin deploy --name foo:1.0 --target cluster1 foo.war
asadmin enable --retiretimeout=10 --target cluster1 foo:beta
asadmin list-applications -l cluster1
# More than 10 seconds later
asadmin list-applications -l cluster1
```

Observed Result :

```
NAME        TYPE       STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta     web         enable      active
foo:1.0      web         enable      retired                       [current_time + 10 sec]
```

```
# More than 10 seconds later, second asadmin list-applications -l
NAME       TYPE     STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta   web      enable    active
foo:1.0    web      disabled
```

Scenario 9-cluster

Execution :

```
asadmin deploy --name foo:beta --target cluster1 foo.war
asadmin deploy --retiretimeout=10 --name foo:1.0 --target cluster1 foo.war
# foo:beta is in retirement.
asadmin enable --retiretimeout=10 --target cluster1 foo:beta
asadmin list-applications -l cluster1
# After 10 seconds, foo:1.0 is disabled.
asadmin list-applications -l cluster1
```

Observed Result :

```
NAME       TYPE     STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta   web      enable    active
foo:1.0    web      enable    retired             [current_time + 10 sec]
# After 10 seconds, foo:1.0 is disabled.
NAME       TYPE     STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta   web      enable    active
foo:1.0    web      disabled
```

Scenario 10-cluster

Execution :

```
asadmin deploy --name foo:beta --target cluster1 foo.war
# Open a session (called sess1 for the example).
asadmin deploy --retiretimeout=10 --name foo:1.0 --target cluster1 foo.war
asadmin list-applications -l cluster1
# Open a second session (called sess2 for the example).
# Check requests on sess1 et sess2
# After 10 seconds
asadmin list-applications -l cluster1
# Check requests on sess1 et sess2
```

Observed Result :

```
NAME       TYPE     STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta   web      enable    retired             [current_time + 10 sec]
foo:1.0    web      enable    active
# Check that sess1 is still redirected on foo:beta and sess2 on foo:1.0 .
# After 10 seconds, foo:beta is disabled. Check that new request with sess1 are
# redirected on foo:1.0 .
NAME        TYPE       STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta    web        disabled
foo:1.0     web        enable      active
```

Scenario 11-cluster

Execution :

```
asadmin deploy --name foo:beta --target cluster1 foo.war
# Start a session (sess1)
asadmin deploy --retiretimeout=-1 --name foo:1.0 --target cluster1 foo.war
asadmin list-applications -l cluster1
# wait for sess1 to expire.
asadmin list-applications -l cluster1
```

Observed Result :

```
NAME        TYPE       STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta    web        enable      retired                   All session expired
foo:1.0     web        enable      active
# wait for sess1 to expire
NAME        TYPE       STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta    web        disabled
foo:1.0     web        enable      active
```

Scenario 12-cluster

Execution :

```
asadmin deploy --name foo:beta --target cluster1 foo.war
asadmin deploy --name foo:1.0 foo.war
# Start a session (called sess1 for the example)
asadmin enable --retiretimeout= -1 --target cluster1 foo:beta
asadmin list-applications -l cluster1
# wait for sess1 to expire
asadmin list-applications -l cluster1
```

Observed Result :

```
NAME        TYPE       STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta    web        enable      active
foo:1.0     web        enable      retired                   All session expired
# wait for sess1 to expire
NAME        TYPE       STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta    web        enable      active
foo:1.0     web        disabled
```

Scenario 13 : version retirement without the DAS up

Version is deployed with a production retirement of 30 seconds in cluster mode, during that time we stop the DAS, we expect the version to be disabled anyway.

Scenario description :

Deploy vbeta of foo
Deploy v1.0 of foo with retiretimeout = 30
Checkout that v1.0 is deployed and vbeta is retiring
Stop DAS
After 30 seconds check that vbeta is disable in i1

Execution :

```
asadmin deploy --name foo:beta --target cluster1 foo.war
asadmin list-applications -l cluster1
# Start session
asadmin deploy --name foo:1.0 --target cluster1 --retiretimeout=30 foo.war
asadmin list-applications -l cluster1
 # wait for 30 seconds
```

Observed Result :

```
NAME       TYPE      STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta    web       enable     active
# Start session
NAME       TYPE      STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta    web       enable     retired               [current_time + 10 sec]
foo:1.0     web       enable     active
 # wait for 30 seconds
 # check that http://<i1 url + port>/foo/ is pointing on v1.0
```

Part III : Error handling

Scenario 14

Trying to change context root on version upgrage.

Scenario description :

Deploy vbeta of foo
Deploy v1.0 of foo with a different context root (“newfoo”)
Check vbeta of foo is still deployed

Execution :

```
asadmin deploy --name foo:beta foo.war
asadmin deploy --retiretimeout=10 --contextroot=newfoo --name foo:1.0 foo.war
asadmin list-applications -l
```

Observed Result :

```
Error during deployment. Command deploy unsuccessful.
ERROR: active version of application "newfoo" uses a different context root : foo.
NAME       TYPE      STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta   web       enable    active
```

Scenario 15

Trying to add a new version of an application with a retiretimeout (using production redeployement) without any pre existing version.

Scenario description :

Deploy v1.0 of foo
Check the application has not been deployed

Execution :

```
asadmin deploy --retiretimeout=10 --name foo:1.0 foo.war
asadmin list-applications -l
```

Observed Result :

```
Error during deployment. Command deploy unsuccessful.
ERROR: there is no active version of application "foo".
Nothing to list.
```

Scenario 16

Trying deploy a new version of an application with a retiretimeout when the deployed version is disabled.

Scenario description :

Deploy disabled vbeta of foo
Deploy v1.0 of foo with a retire timeout
Check vbeta of foo is still deployed and enabled and v1.0 has not been deployed

Execution :

```
asadmin deploy --enable=false --name foo:beta foo.war
asadmin deploy --retiretimeout=10 --name foo:1.0 foo.war
asadmin list-applications -l
```

Observed Result :

```
Error during deployment. Command deploy unsuccessful.
ERROR: there is no active version of application "foo".
NAME       TYPE      STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta   web       disable
```

Scenario 17

Trying to to enable a version with a retire timeout when no version is enabled.

Scenario description :

Deploy disable vbeta of foo
Deploy disabled v1.0 of foo
Enable v1.0 of foo with retire timeout of 10 seconds
Check both vbeta and v1.0 are deployed and disabled

Execution :

```
asadmin deploy --enable=false --name foo:beta foo.war
asadmin deploy --name foo:1.0 --enable=false foo.war
asadmin enable --retiretimeout=10 foo:1.0
asadmin list-applications -l
```

Observed Result :

```
Error during deployment. Command deploy unsuccessful.
ERROR: there is no active version of application "foo".
NAME        TYPE      STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta    web       disable
foo:1.0     web       disable
```

Scenario 18

Trying to deploy two successive versions with retire timeout (original timeout not expired).

Scenario description :

Deploy vbeta of foo
Deploy v1.0 of foo with retiretimeout of 10 seconds
Deploy v2.0 of foo with retiretimeout of 10 seconds (why vbeta is retiring)
Checkout that vbeta of foo is retiring, v1.0 of foo is active and v2.0 has not been deployed

Execution :

```
asadmin deploy --name foo:beta foo.war
asadmin deploy --retiretimeout=10 --name foo:1.0 foo.war
asadmin deploy --retiretimeout=10 --name foo:2.0 foo.war
asadmin list-applications -l
```

Observed Result :

```
Error during deployment. Command deploy unsuccessful.
ERROR: there is already two active versions of application "foo".
NAME        TYPE      STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta    web       enable    retired             [current_time + 10 sec]
```

```
foo:1.0        web         active
```

Scenario 19

Trying to enable a third version of an application while second one is deployed and original one is being retired.

Scenario description :

Deploy vbeta of foo
Deploy v1.0 of foo with a retiretimeout of 10 seconds
Deploy disabled v2.0 of foo
Enable v2.0 of foo (while v1.0 is retiring)
Checkout that vbeta is retiring, v1.0 is active and v2.0 has not been deployed

Execution :

```
asadmin deploy --name foo:beta foo.war
asadmin deploy --retiretimeout=10 --name foo:1.0 foo.war
asadmin deploy --name foo:2.0 --enable=false foo.war
asadmin enable --retiretimeout=10 foo:2.0
asadmin list-applications -l
```

Observed Result :

```
Error during enable. Command enable unsuccessful.
ERROR: there is already two active versions of application "foo".
NAME       TYPE      STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta   web       enable    retired              [current_time + 10 sec]
foo:1.0    web       active
```

Scenario 20

Trying to use Production Redeployement without specifying deployed version name.

Scenario description :

Deploy vbeta of foo
Deploy foo without specifying the version
Checkout that vbeta is still deployed and that new version has not been deployed

Execution :

```
asadmin deploy --name foo:beta foo.war
asadmin deploy --retiretimeout=10 foo.war
asadmin list-applications -l
```

Observed Result :

```
Error during deployment. Command deploy unsuccessful.
ERROR: Production Redeployment require the option --name.
NAME       TYPE      STATUS    EXTENDED_STATUS   RETIRES_ON
foo:beta   web       enable    active
```