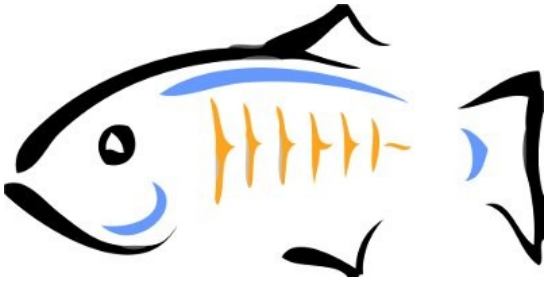

# POC DESIGN DOCUMENT

---

*Glassfish application versioning*

SERLI

Romain GRECOURT romain.grecourt@serli.com
Hervé SOUCHAUD herve.souchaud@serli.com
Mathieu ANCELIN Mathieu.ancelin@serli.com
Pierre Henri DEZANNEAU pierrehenri.dezanneau@serli.com

10 March 2010

# 1 Summary

# 2 Introduction

## 2.1 Glassfish deployment

Currently, the Glassfish deployment system allow only one version of an application to exist at a time in a domain. So, if a user wants to deploy new versions of a deployed application, he needs to undeploy the old one first, and then deploy the new version in its domain. If the user wants to switch back to the old application, he needs to do the same process.

In the case of a 24/7 service, this kind of operation can be problematic because the application will be unavailable for some time and the service will be interrupted.

## 2.2 Application versioning

An application versioning service will allow multiple versions of the same application to exist in a Glassfish domain and only one of these versions to be enabled. This service will provide facilities to switch between application versions and to avoid interruption of service during switches.

This system will be administrable from every administration client (CLI, GUI, etc ...), easy to use and virtually transparent for the user.

# 3 Serli's Proof of Concept

After reading the ticket about applications versioning in the Glassfish bug tracker (https://glassfish.dev.java.net/issues/show_bug.cgi?id=4100) we started to think about a proof of concept to show how it can be done in Glassfish V3.

The following points are proposal on how the things can be done to implement a versioning service in glassfish.

## 3.1 Versions storage

The first thing to consider in a versioning system is how to store different versions of applications in a Glassfish domain.

We chose to create an alternate file tree inside the domain to store each version of a deployed app. We decided to add “version folders” inside applications folders to store each version of a particular app as showed in illustration 1.

From here, the versioning system and administration commands will use these folders to manage (deploy, undeploy, enable, disable, switch, rollback, etc ...) applications versions.

This tree will further be defined as “versioning repository” or “versioning filesystem” in this document.

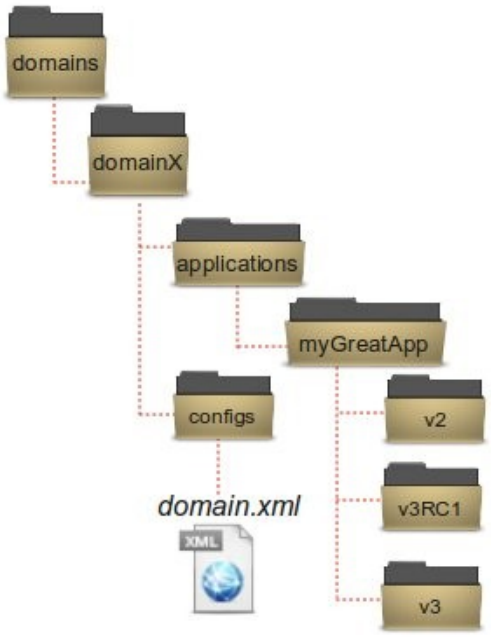


*Illustration 1: Versioning filesystem inside a domain*

## 3.2 Versioning service

The versioning service is a new module in Glassfish deployment. The aim of this module is to manage a registry of applications with its associated versions. This module collaborate closely with administration commands to manage its applications versions registry and to refer to the actual deployed version of a particular application.

The service also provides information from its registry to let other services know the versioning status of a particular application.

The illustration below describes how a versioned deployment works using the versioning service.

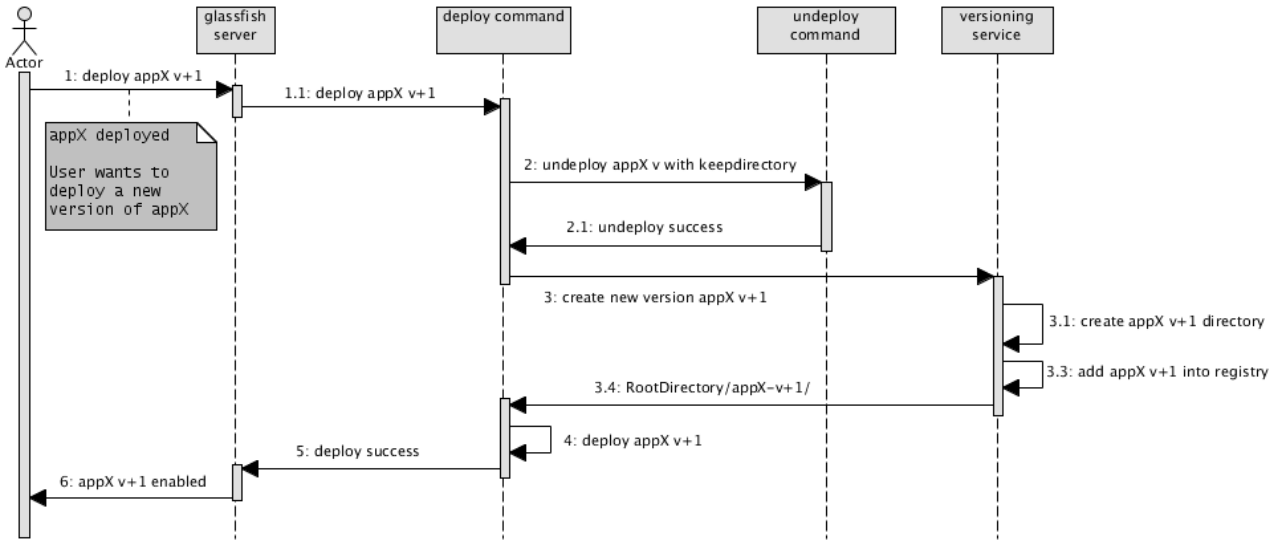


*Illustration 2: Deploying a new application version*

## 3.3 Impact on existing administration commands

Adding a versioning service in Glassfish deployment impacts the related administration commands.

The basic operations for a versioning system are to deploy and undeploy a version of an application. As these operations already exist in administration commands, there is no reason to create new commands.

The deploy command is modified to check if the application can be versioned and to deploy it using the process described in point 3.2.

The undeploy command undeploys the actual version of the app and deletes it from the versioning repository. It's also possible to add an option to delete all versions of the app in the versioning repository.

There are other commands that need to be modified. For instance enable and disable commands needs to be modified to know which version is actually targeted by the versioning system.

## *3.4 New administration commands*

One of the main feature of application versioning is the switch between several versions of the same application at runtime. As this feature has no equivalent in the current administration command, we thought that the best way to do it easily is to add a new administration command named “SWITCH”.

We can easily imagine to use the “SWITCH” command this way :

```
asadmin switch --targetVersion=v3RC1 myGreatApp
```

In this case, the “targetVersion” argument is a version of “myGreatApp.ear” present in the versioning repository (as describe in point 3.1).

This command will undeploys the actual version of myGreatApp.ear and automatically deploys the “targetVersion” version of the application as described in 3.2 process.

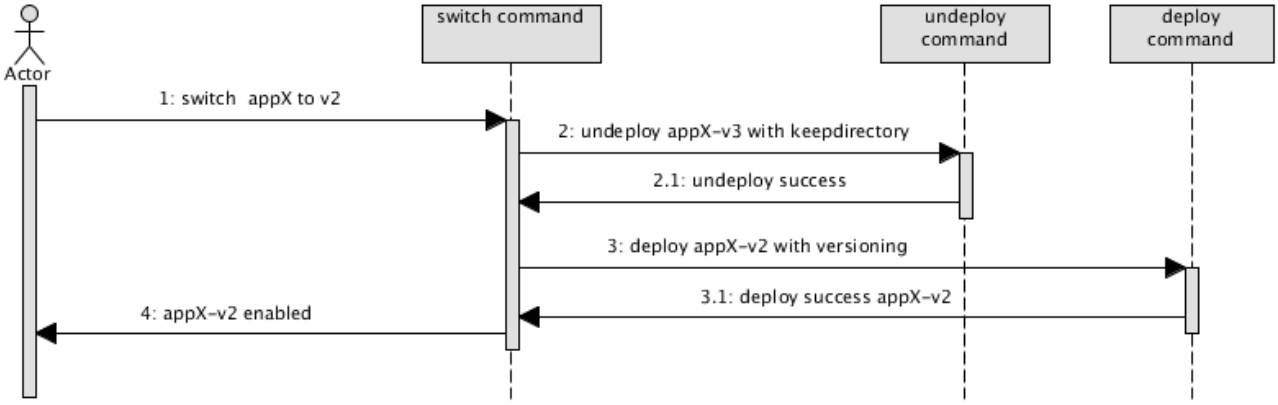


*Illustration 3: Basic switch version command*

Another possible use case is the rollback command. We considere this command as a transaction rollback. The typical scenario for this use case would be :

A versioned application is deployed but unfortunately a regression is observed which wasn't diagnosed by unit tests or integration tests. So, the application is dysfunctional and users can't access the app correctly. In this case, the administrator will use the rollback command like this :

```
asadmin rollback myGreatApp
```

This command will undeploy the current (dysfunctional) version of the app, delete it and deploy the last stable version. This command uses the “SWITCH” command, because the rollback is nothing more than a switch and a delete.

# 4 Current status

**POC features** :

- Manage different versions of the same application at runtime
- Add / remove version at runtime
- Switch application version at runtime
- Auto-versioning
- Version name auto-generation

**Not implemented at the time of writing this document** :

- Enable / disable command support
- Version rollback
- Configuration from file (maybe domain.xml)
- Custom version names

# 5 Differences with functional specification document

We didn't check V2 backward compatibility so maybe our versioning repository isn't really adapted to drop a V3 domain in a V2 app server.

Advanced version name management. We didn't introduce notions of version expression and version identifier used in administration commands.

Also, we didn't use concepts of current and default version.