Manual

Staging tool for Payara / GlassFish 4

Version:	*1.4*
Date:	*06.04.2016*

Amendment log:

Version	Date	Authors	Changes
1.1	18.11.2015	Cidália Mira, C. Schaarschmidt	Initial release
1.2	30.11.2015	Andreas Letsche, FG-931	First review
1.3	17.12.2015	Cidália Mira	Second review and additional information.
1.4	06.04.2016	Cidália Mira	Added new functions and clarify some conflicting information.

Table of contents

1 Introduction

This document describes the staging process in the Master Solution Java Application Server Premium Payara / GlassFish 4 project environment at BMW and how to prepare a web application accordingly, as well as the differences between Master Solution Java Application Server Premium Payara / GlassFish 4, Master Solution Java Application Server Premium V5.0 (based on GlassFish 2) and Master Solution Java Application Server Premium V6.0 (based on GlassFish 3). It is intended primarily for the developers and for the persons responsible for technical maintenance of these web applications.

It is strongly recommended to read the documentation of the master solution prior to reading this documentation. The documentation can be found on the master solution website (java.muc > Master Solutions) under “Basics / External view”.

The staging process has been introduced with version 5.0 of the master solution for application servers. Previous master solutions used a different process, which is not covered by this document.

Contents:

Chapter 2 of this document will give you an overview of what the staging tool does and why this approach was chosen.

Chapter 3 describes the procedures for initial setup, packaging and deployment of an application.

Chapter 4 describes the configuration elements provided by the staging tool.

Chapter 5 contains the syntax reference of the staging tool’s front end.

Chapter 6 contains some hints for troubleshooting.

Chapter 7 contains the differences introduced with Payara / GlassFish 4 master solution.

2 Overview

2.1 Staging process and staging tool

Before a project is finally deployed to the production servers, it runs through a sequence of other stages:

1. Test
2. Integration
3. Approval
4. Production

Deployments on each of these stages must be performed using the staging tool. The tool streamlines the process of bringing a project and its configuration to the various stages successively.

Please note that the use of the tool is **mandatory**. Only applications packaged with the staging tool will be accepted for Production or Approval.

2.2 Principle of operation

The main objective of the staging tool is to separate the configuration options provided by web operations and those provided by the web application developers. The application configurations can be split further in a general part and a stage specific part, e. g. to configure different credentials for each stage.

After the initial setup, the project (means the web application(s), WAR- or EAR file(s) and the configuration files) is packed into an archive file (using the command *./*`GFserver4 create-tar`), the so-called “tarball”. The “tarball” is then used for deployments on the following stages. By splitting the project configuration into a general part and a stage specific part, the “tarball” can contain the project configuration settings for **all** stages. The staging concept allows one properly configured “tarball” to be deployed to all stages without further customizing.

During deployment the staging tool merges web operations specific configuration files and project / web application specific configuration files. The resulting “complete configuration” is then used to build the “clusterDomain” and “nodes” via DAS (Domain Administration Server) from scratch each time the “tarball” is deployed.

2.3 Benefits

The procedure mentioned above grants a number of benefits:

- Using the same “tarball” for all stages ensures that the web application goes into production at least with the same general configuration settings it has been tested with. The stage depended settings can only be used / tested in the respective stage.
- Default application server configuration settings (suitable for approx. 90% for all web applications operated) are provided by web operations, so application developers can concentrate on the configuration settings for their web application mainly.
- Configuration changes using the admin console (deployed on DAS) are possible **for testing purposes only** in test and integration stage but will be ignored / removed during staging in production **by intention.**

- The main goal is to be able to generate a “fresh clusterDomain” based on well-defined XML files from web operations and application operations. This approach reduces the risk of pushing wrong configurations to production since only changes in the web applications and web operations XML configuration files are relevant.
- The “web application specific configuration” can be kept as part of the web application resources in the source code repository on application side and thus providing a history of changes.
- The possibility to reset an application server environment to its initial state and afterwards start with a “freshly configured clusterDomain”.

3 Usage

Important notes:

- All *path names* of commands, files and directories in this chapter are relative to the project directory ("/www/<PROJECT_SHORT_NAME>/") and are marked in *italic font*.
- The "Domain Administration Server" is the Administration Payara / GlassFish 4 instance and it is used for "special purposes"/administration in each Payara / GlassFish 4 cluster domain.
    - It is mostly referenced as "DAS" in this document. "DAS" is the "owner" of each Payara / Glassfish 4 "clusterDomain".
    - It is needed for recreation and reconfiguration of the "clusterDomain" as well as for the first start attempt of any dependent Payara / Glassfish 4 instance/node because these instances/nodes must download the project configuration (web application + configuration) from DAS and store in their instance local directory under *nodes*.
    - The "**admin console**" is deployed on "DAS" as well but **does not play any role at BMW for configuration of the cluster** because configuration is not done via admin console.

3.1 Initial setup of a project environment

In order to bring a project (meaning web application and its configuration) into the staging process for the first time, the following steps are necessary. This initial setup is usually taking place on the test stage. If no test stage is used or the test stage shall not be used for this purpose it is possible to use any integration environment for this.

1. As result of the provisioning process web operations provides a "fresh" project environment with default configuration settings using the latest available config version. Regarding config version please refer to chapter 4.
2. Application developers or technical maintenance copy the web applications EAR/WAR-file(s) into the *config* directory.
3. Application developers configure the project specific settings under *config/conf.d* (see chapter 4. "Configuration")
4. Application developers or technical maintenance function initialize the "clusterDomain" and start the GlassFish instances by using the GFserver4 control script (respectively staging tool). The steps are the following:

Payara 4.1 / GlassFish 4
Run *./GFserver4 stop --all* **on each hosting server** of the cluster to stop all instances
Run *./GFserver4 remove-domain* on the hosting server running DAS (Deletes the *clusterDomain* and *nodes* directory.)
Run *./GFserver4 restore* on the hosting server running DAS to "freshly" create the *clusterDomain* directory and its contents via DAS.
Run *./GFserver4 start --all* **on each hosting server** (beginning with the hosting server where DAS is running) to start all GlassFish4 instances of the project environment including DAS itself. **Important Note:** During the first start of a Glassfish 4 instance an "instance specific local domain" is created under *nodes* and the web application is downloaded from DAS. Therefore DAS needs to be running in order to be able to start the Glassfish4 instances for the first time.
Stop DAS *./GFserver4 stop --admin*

3.2 Configuration procedure

The configuration of a project environment that has been provisioned involves two steps:

1. Edit the project specific XML files in the directory *config/conf.d* (see chapter 4. "Configuration")
2. Reconfigure and restart the GlassFish4 instances.

To reconfigure the project environment, there are two different methods available, called *restore* and *reconfigure*. The first one will completely recreate the *clusterDomain* directory and its content via DAS from scratch. It is needed in the following cases:

- The JVM-, GlassFish- or config version needs to be changed
- An element has been removed from one or more of the XML configuration files (reconfiguration will **not** remove existing configuration settings from the project environment (content of *clusterDomain* directory. Only recreation will.)
- The existing project environment must be cleaned up. In all other cases, a reconfiguration is sufficient.

1. The steps for recreation of the domain are the same as described in the Initial Setup:

Payara 4.1 / GlassFish 4
Run `./GFserver4 stop --all` **on each hosting server** to stop all instances including DAS.
Run `./GFserver4 remove-domain` on the hosting server running DAS to delete the *clusterDomain* and *nodes* (contains instance specific local domains) directory.
Run `./GFserver4 restore` on the hosting server running DAS to recreate the *clusterDomain* directory and its contents.
Run `./GFserver4 start --all` **on each hosting server** (beginning with DAS) to start all Payara / GlassFish 4 instances
Stop DAS `./GFserver4 stop --admin`

2. The steps for reconfiguration are

Payara 4.1 / GlassFish 4
Run `./GFserver4 configure` on the hosting server running DAS to create the *clusterDomain* directory and its contents.
Run `./GFserver4 stop --all` **on each hosting server** to stop all Payara / GlassFish 4 instances including DAS.
Run `./GFserver4 start --all` **on each hostingserver** (beginning with DAS) to start all Payara / GlassFish 4 instances including DAS.
Stop DAS `./GFserver4 stop --admin`

Please note the differences:

- For **restore** the Payara / GlassFish 4 instances must be stopped and the directories *clusterDomain* and *nodes* removed beforehand. The *clusterDomain* is created from scratch.
- For **configure** the Payara / GlassFish 4 instances must be running and **nothing is deleted**. The *clusterDomain* directory is kept as is, only the desired settings in the project specific XML configuration files are applied to the contents in the *clusterDomain* directory.

3.3 Packaging of “tarballs” for the mandatory standard procedure

The packing of “tarballs” must be done as follows:

- The project configuration (meaning the web application and its configuration) is packed into an archive file (using the command `./GFserver4 create-tar` command), the so-called "tarball".
- The "tarball" must be created with the `./GFserver4 create-tar` command. This command packs the directory *config* (including all content within this directory), and it will try to pack the optional directory *project-synced*, but no other. This command will fail if *config* directory doesn't exist and will warn you if the *project-synced* directory doesn't exist. The "tarball" will be placed under your *project-dir/tmp* directory. In case *tmp* directory doesn't exist, it will be created by the command.
- The web applications WAR or EAR files must be located within the *config* directory or one of its subdirectories. They must be part of the "tarball".
- For each change there must be a documentation file available and named exactly *config/README* (an existing README file can be modified or overwritten within changes).Within the README file a technical contact person who is available during the whole change procedure has to be provided.
- The *README* file must contain any requirements regarding special approval by the technical contact for certain steps of the deployment procedure if applicable. If there are any non-standard requirements, then these must be documented here as well.

  Important:

  The **special agreement process** which is documented under "http://web.bmwgroup.net/ => Java => Special Agreements" has to be considered beforehand if **NON standard configuration settings** need to be used. Especially for "decided by request special agreements" it is necessary to get in contact with FG-931 prior to requesting such changes via the README file as FG-931 internals are not available 24x7 on demand.
- The "tarball" deployment procedure itself (as described below) is fixed to follow a strict standard and to allow automation and cannot be modified.
- The "tarball" must be deployable without any modifications of content. Scripts that dynamically modify configuration files are not allowed. Profile based settings must be used for this.

3.4 "Tarball" deployment procedure

To deploy a "tarball", always follow the fixed, standard procedure below. All commands of the deployment procedure must be executed by the project user (QQ-user). The procedure is the same for the first time deployments as well as for redeployments.

The following chapters define the fixed, standard procedure. If you decide, for a good and valid reason, to differ from this standard procedure this has to be documented clearly in the README file! Web operations strongly discourage to differ from the standard if it is not really necessary.

3.4.1 Preparation

In the past "tarballs" have often been placed directly into the project directory *"/www/<PROJECT_SHORT_NAME>/"*. To keep the project directory clean please place your "tarballs" inside the *tmp* subdirectory. This allows for example much easier removal of "old tarballs" if necessary. The command `./GFserver4 extract-tar` will search for the "tarball" inside this subdirectory.

Payara 4.1 / GlassFish 4
Provide the "tarball" inside *tmp* directory on the server where DAS is running (which is as per standard the hosting server with the lower number (e.g. lpapp401vm for a cluster running on lpapp401vm/lpapp402vm.). Please always ensure unique names of "tarballs" to prevent mix-up of "tarballs".
Extract the README file and the (optional) project specific prechange script(s): *`./GFserver4 extract-readme –f <tarball_filename>`* *`./GFserver4 extract-prechange –f <tarball_filename>`* (optional)
Read the README file

3.4.2 Deployment

Payara 4.1 / GlassFish 4
If it exists, run the prechange script (optional): `./GFserver4 exec -f config/hooks/prechange`
Stop the Payara / GlassFish 4 instance(s) and DAS by executing the following command on each hosting server of the cluster: `./GFserver4 stop --all`
Clean currently existig *config* directory on the hosting server where DASis running: `./GFserver4 remove-config`

Clean the content of the directories *project-local* and/or *project-synced* if you **need to delete** content in these directories (optional):
`./GFserver4 remove-project-local`
`./GFserver4 remove-project-synced`

Extracts the “tarball” located under *tmp/<tarball_filename>*, this will extract only the directories *config*, *project-synced* if they are contained in the “tarball” (everything else which is in the “tarball” will be ignored by this command) : `./GFserver4 extract-tar -f <tarball_filename>`
If it exists, run the change script (optional): `./GFserver4 exec -f config/hooks/change.`
Remove “old” *clusterDomain* and *nodes* (contains node specific domains) directories: `./GFserver4 remove-domain`
Recreate the *clusterDomain* directory and its contents. `./GFserver4 restore` (Note: This will implicitly start and stop DAS)
Start the application server instances by executing the following command on each hosting server of the cluster (beginning with the hosting server where DAS is running): `./GFserver4 start --all`
If it exists, run the postchange script `./GFserver4 exec -f config/hooks/postchange`
Stop DAS `./GFserver4 stop --admin`

Important Notes:

- The scripts in the directory *config/hooks* are optional and the use requires a special agreement. See section "Additional Scripts" for a detailed explanation. If the execution of one of them fails, the entire change might fail and a rollback might be necessary if the issue cannot be found and fixed on short term basis.
- The content of the *config* directory **must be** deleted before unpacking the "tarball" in order to successfully execute the deployment procedure, the content of the *project-synced* directory **must be** deleted as well.
  - ATTENTION:
    - If the content of the directories *project-synced* is **not** deleted it will be replaced with data from the "tarball". Existing files that do not exist in the "tarball" are **not** deleted!
    - To delete existing, individual files from *project-synced* and *project-local* directories, it is absolutely necessary to request this explicitly in the README file!
- When extracting the "tarball", error messages related to *project-synced* not found in the archive can be ignored, if these directories are not present in the "tarball".

3.4.3 Upgrading web applications without loss of availability

This feature was introduced with config version 20110202 of GlassFish3 master solution V6 and it requires a pre-approved special agreement from web operations
(see "web.bmwgroup.net > Java > Special Agreements for more information").

Please be aware of the restrictions described below. You cannot use this feature for any of these changes, because we cannot ensure it works as expected and some don't work at all:

- Change JAVA version
- Change Payara / GlassFish 4 version
- Change config-version
- Delete or rename any resource
- Only compatible web applications can be deployed

From Sun GlassFish Enterprise Server High Availability Administration Guide:

Application Compatibility

Rolling upgrades pose varying degrees of difficulty depending on the magnitude of changes between the two application versions.

If the changes are superficial, for example, changes to static text and images, the two versions of the application are compatible and can both run at once in the same cluster.

Compatible applications must:

- *Use the same session information*
- *Use compatible database schemas*

- *Have generally compatible application-level business logic*
- *Use the same physical data source*

[…]

If the two versions of an application do not meet all the above criteria, then the applications are considered ***incompatible.*** *Executing incompatible versions of an application in one cluster can corrupt application data and cause session failover to not function correctly. The problems depend on the type and extent of the incompatibility. […]*

The application developer and administrator are the best people to determine whether application versions are compatible. If in doubt, assume that the versions are incompatible, since this is the safest approach.

If your application change fulfils all the requirements described above, to request a change with zero downtime document it in the change ticket and the README file.

Steps to follow for a rolling-upgrade (no downtime deployment):

Payara 4.1 / GlassFish 4
Execute “remove-prerolling-upgrade” (optional) `./GFserver4 remove-prerolling-upgrade`
Extract the README file and the (optional) prerolling-upgrade script: `./GFserver4 extract-readme –f <tarball_filename>` `./GFserver4 extract-prerolling-upgrade –f <tarball_filename>` *(optional)*
Read the contents of the README file
Execute “config/hooks/prerolling-upgrade” (optional) `./GFserver4 exec -f config/hooks/prerolling-upgrade`
Extract the “tarball” this will extract only the directories *config*, *project-synced* and *project-local* if they are contained in the “tarball” (everything else which is in the “tarball” will be ignored by this command): `./GFserver4 extract-tar –f <tarball_filename>`
Execute config/hooks/rolling-upgrade (optional) `./GFserver4 exec -f config/hooks/rolling-upgrade`
Start with rolling-upgrade mode and reconfigure existing *clusterDomain* `./GFserver4 rolling-upgrade`
Restart one Payara / GlassFish 4 instance after the other starting with the hosting server where DAS is running `./GFserver4 stop --all start --all status`
Finish rolling-upgrade mode `./GFserver4 end-rolling-upgrade`
If it exists, run the postrolling-upgrade script (optional) `./GFserver4 exec –f config/hooks/postrolling-upgrade`
Stop DAS `./GFserver4 stop --admin`

3.4.3.1 Balancer Manager Feature:

There is a potential risk of losing requests during restart of Payara / GlassFish 4 instances. Starting with master solution premium 6 based on GlassFish 3 it has been possible to disable the requests to be directed to a particular GlassFish instance on the “Apache” using “mod_proxy_balancer” and “mod_proxy_balancer_manager” prior to stop and respectively restart these instance, so that requests will not be sent anymore to GlassFish instances which are being stopped or restarted.

This feature is disabled by default. To activate this feature please follow the description in the respective documentation on the website “http://web.bmwgroup.net/” in the FAQs section.

DE: web.bmwgroup.net > FAQ > Wie kann ich Balancer Manager Support beim GlassFish 3 oder Payara / GlassFish 4 aktivieren?

EN: FAQ -> How do I enable Balancer Manager Support for Glassfish3 or Payara / GlassFish4?

3.5 Final testing

Before handing over a “tarball” to web operations, application operations must

1. Configure the necessary connection pools (JDBC, MQ etc) for **all** environments
2. Configure performance-relevant settings (e.g. execute thread settings) for **all** environments.
3. Test the “tarball” deployment in integration.

For the final test of the “tarball”, the project must execute the change in the integration environment in the same way as web operations will do in production. Only “tarballs” that have been successfully tested in integration may be handed over to web operations.

Application operations will be held responsible for incidents caused by incorrect “tarballs” or by project specific scripts. Web operations will **not** check the contents of the “tarball”.

During deployment in the production environment, the integration environment must be available for web operations for reference in case there are issues during deployment.

3.6 Caveats

Paying attention to the following list of caveats may save you some time and unnecessary support calls.

- The application server instances must be started and stopped via the *GFserver4* command only. The use of *asadmin* tool which is built-in into Payara / GlassFish 4 is discouraged.
- The commands *GFserver4 configure or GFserver4 restore* must be executed on the hosting server where DAS is running.
- UNIX environment variables (e.g. JAVA paths) will be ignored when running the GFserver4 command. They will **not** be carried over to the Payara / GlassFish4 environment.
- Configuration changes via admin console will not be persistent and transferred to the next stage. Configuration has to occur in the XML configuration files under config/conf.d/[1…8][0…9]_*.xml.

- The “tarball” may only contain the directories *config* and *project-synced*. Other directories are not allowed and therefore ignored.
- Application developers may only use a predefined set of configuration options (JVM options and dotted names). If any additional option is needed, which is not available in the predefined set, you must request it from web operations team on a “per-web environment-basis”. The respective special agreement process needs to be followed.
- Any changes must be done on the hosting server (mostly the one with the lowest index) where DAS is running because the synchronization between the different instances only happens in one way. From the hosting server running DAS to the other hosting servers running Payara / GlassFish4 instances.
- Be aware that the NAS shares (application server shares) are **not shared** between the Payara / GlassFish 4 instances. Every Payara / GlassFish4 instance on every hosting server get its own dedicated application server share.
- Keep the project directory “clean” to avoid mix-up and/or confusion and to ensure a good overview for everybody.

4 Configuration

4.1 Directory structure

The following directories within the GlassFish4 project directory are relevant for application developers or technical maintenance functions:

/www/<PROJECT_SHORT_NAME>/**project-synced**

Working directory for a **small amount** of project specific static content/data that needs to be synchronized from the hosting server where DAS and the first Payara / GlassFish4 instance is running to all other hosting servers in the cluster (see section split project storage). Synchronization is happening during recreation and configuration of the *clusterDomain* directory and its contents within the staging process. The UNIX tool "rsync" will be used to synchronize the content of the "source *project-synced*" directory to the "destination *project-synced*" directories. The respective targets (NAS-shares) are mounted on the first hosting server to ease this synchronization process.

/www/<PROJECT_SHORT_NAME>/**logs/project**

Directory for project log files. May be specified via Java property "com.bmw.mastersolutions.gf.project.logs"

/www/<PROJECT_SHORT_NAME>/**project-local** (optional)

This directory is the working directory for project specific content which is only available locally on the respective hosting server for the respective Payara / GlassFish4 instance. The path is accessible through the Java property "com.bmw.mastersolutions.gf.project.data". This directory will **never** be synchronized and must never be used as mount point for any shared NAS share. It is meant to be used for temporary local files and dynamically created local content (by the application) which doesn't need to be shared by the application with any other Payara / Glassfish 4 instance/node.

/www/<PROJECT_SHORT_NAME>/**project-shared** (optional)

"**Symbolic link!!**" to the mount point for a shared NAS based file system. Makes sense when you need application files/data to be shared between all the Payara / GlasFish 4 instances/nodes and you need a large amount of data (> 500 MB) on the respective shared share. The *project-synced* directory (see above) mustn't be used for large data amounts!! Large amount of data can be stored on a so called "**Shared Web Share**" (<100 GB) or "**NAS (iStore) Application Share"** (Size limit according to NAS operations)**.** For more information regarding shared shares please refer to the storage documentation on the website "http://web.bmwgroup.net/" under the section "Additional Services".

/www/<PROJECT_SHORT_NAME>/**clusterDomain,**
/www/<PROJECT_SHORT_NAME>/**nodes**

This directory is automatically created by staging tool. Must be deleted before running `GFserver4 restore`, otherwise **do not touch** the contents in those directories! These directories are for **internal use only!** Do not use them directly in any of your scripts nor within your CLASSPATH configuration.

/www/<PROJECT_SHORT_NAME>/**config**

This is the main interesting directory for application developers and technical maintenance team. All files required for configuration with the exception of *project-synced* (if really necessary must be located within this file structure) contains:

- web application(s) respectively EAR- or WAR file(s)
- project configuration files
- additional libraries (optional)
- additional hook scripts for deployment (optional)

The following sections describe the structure within the *config* directory.

/www/<PROJECT_SHORT_NAME>/**config/conf.d**

Configuration files for the project. All configurations (with the exception of JAAF) are done here.

/www/<PROJECT_SHORT_NAME>/**config/lib**

Normally, libraries should always be bundled into the EAR or WAR file of the project. However, in some cases it is more convenient to use “applibs”. In this case, the library may be placed here. (Please note that “applibs” are not included by default, usage is configured at application level)

/www/<PROJECT_SHORT_NAME>/**config/endorsed**

Additional libraries provided by web operations if necessary.

/www/<PROJECT_SHORT_NAME>/**config/hooks (optional)**

Optional directory for additional scripts to be execute before, during or after a change. See section “Additional scripts” for a detailed description.

/www/<PROJECT_SHORT_NAME>/**config/trust.d** (optional)

Optional directory for trust certificates in PEM format. If the directory exists and is not empty, a “trust store” will be created under *config* and loaded by the Payara / GlassFish 4 instances from that directory. The certificates must be in a format that the Java keytool accepts. The alias name inside the “trust store” is derived from the filename by stripping the suffix (e.g. filename *123.456.pem* becomes the alias *123.456*).

/www/<PROJECT_SHORT_NAME>/**config/key.d**

Projects with additional certificate requirements (e.g. use of client certificates for two way SSL based authorization) which cannot be addressed with the existing central key stores, may provide their own project specific certificates.
Certificates in PEM format and stored in config/key.d are added to a key store per hosting server which is loaded by the Payara / GlassFish 4 instances.

Important note: The directory structure above is mandatory. If a project does not adhere to this convention it has to take the complete responsibility for this deviation of the standard.

4.2 Configuration files

All configuration changes are located in the directory *config/conf.d*. The files in this directory are processed in alphabetical order. If a configuration item is defined more than once, the last definition will be used.

The filename pattern for project configuration files is "[1-8]*.xml" (in words: starting with a number between 1 and 8, and ending with the suffix ".xml").

The filename pattern for web operations configuration files is "[09]*.xml" (in words: starting with 0 or 9, and ending with the suffix ".xml"). These files are reserved for use by web operations and must not be changed by the project. If changed, the files will be replaced during the next recreation within the staging execution.

As the suffix implies, the format of these files is XML. The corresponding schema definition can be found at the location */wwws/master_solution_gfv4/glassfish/staging/staging-x.y.z* on every hosting server. (Hint: downloading and importing the definition file into an XML editor such as Eclipse allows auto completion and schema validation.)

It is possible to limit the scope of a configuration file to a specific environment (TEST, INT, APPROVAL or PROD) by setting the "profile" attribute within the file (see "http://java.muc/staging > Configuration Reference > Profile" for an example).

Project configuration files can be named arbitrarily as long as they follow the pattern described above. It is, however, recommended to use names similar to the following example:

1. One file called 10_<your-project-name>.xml. This file contains those settings, which apply to all environments, e.g. the web application to deploy or some JVM options.
2. One file for each environment with settings, which must be configured separately, e.g. JDBC settings. The recommended names are
   - 20_prod.xml
   - 30_approval.xml
   - 40_int.xml
   - 50_test.xml

   (**Important note:** the environment, to which a file applies, depends solely on the "**profile attribute**" within the file, **not** on the file name. The file name is just there for the user's convenience.)

4.3 Configuration items

4.3.1 Standard configuration items

Following is a list of standard items, which a project may configure.

- Heap size for Payara / GlassFish 4 instances
- Web applications (WAR- or EAR files).
- JDBC
- Java Mail
- MQ

- Dotted names
- JVM options
- Staging tool / Payara / GlassFish 4 version / JVM version

The complete configuration reference can be found at java.muc/staging > Configuration Reference

4.3.2 Non-standard configuration items

Additional Non-standard project specific items may be configured using dotted names. The use of dotted names as well as JVM options (other than those starting with "*com.bmw*") is restricted and must be requested from web operations based on the mandatory special agreement process. Separate requests must be issued for each project and for each stage/environment (test/int/approval/prod).

Heap sizes may only be configured within certain limits defined by web operations. These limits can be raised upon request and based on the mandatory special agreement process, too.

4.3.3 Staging tool version

The staging tool version defines a set of default configuration settings, e.g. Payara / GlassFish 4 version, JVM version, Java parameters, etc. Web operations may update default settings or scripts from time to time based on new config-versions. In order to benefit from optimized defaults and new features, the project has to set the staging tool version in the configuration element *config-version*.

All available config-versions are described in */wwws/master_solution_gfv4/glassfish/ChangeLog*. In case of doubt, use the latest one. Before changing the *config-version*, always read the change log first, as some changes may require further adjustment in the project configuration.

4.3.4 Config version

The config version defines the version of:

- GFserver4 script to be used. The GFserver4 scripts calls all the functions to be used for this master solution.
- Staging version
- Java version
- All the interface components such as JAAF version, etc.
- Payara /GlassFish version.

4.4 Additional scripts

The optional directory *config/hooks* may contain additional project specific hook scripts that are to be run before, during or after a change, named

- *prechange* executed before servers are stopped and the "tarball" is extracted
- *change* executed after the servers are stopped and the "tarball" is extracted
- *postchange* executed after the servers are started (again).

No other files are allowed in this directory.

Note: Starting with master solution premium 6 (GlassFish v3.x) a special permission is needed for the use of hook-scripts. Without a special-agreement, in test-/integration-environments a warning will appear. In productive environments the following message will appear:

```
[16:17:26] ERROR [HookScript:41] config/hooks/<hook-name> skipped,
no special agreement found
```

The hook scripts must be executed with *"./GFserver4 exec"* to ensure that the environment Variable PROFILE is set to PROD, APPROVAL, INT or TEST according to the application server environment. Any other environment variable may be set to any value.

The hook scripts must be executable (starting with a shebang line and execute bits set). They must terminate with exit code 0, if successful, or with any other code and a meaningful error message, if not. Exit codes other than 0 or abnormal script termination will lead to an abortion of the change.

Please note, that web operations may run reconfigure or recreate for emergency reasons at any time as part of the incident management process. Therefore, the hook scripts must not make any changes that are going to be overwritten by reconfiguration.

Web operations may deny execution of hook scripts if they conflict with good and best practices, enforcement of policies, interoperability and/or adequate resource usage.

The hook scripts also must be specified in the adjoining README documentation file (see Chapter “Packaging). Other scripts than these are not allowed without special agreement.

5 Tool Reference

The script *GFserver4* is the control script and front end of the staging tool.

The GFserver4 script has a built-in online help which may be used by calling "./GFserver4 --help" to get general information about the script and by calling "./GFserver4 <command> --help" to get information about the specific command.

Argument	Effect
`configure/reconfigure [--help]`	Uses the project specific settings in "/www/<PROJECT_SHORT_NAME>/config/conf.d" to configure the domain according to project specific needs by calling the "staging tool". **NOTE:** can only be used if a *clusterDomain* has already been created.
`create-domain [--help]`	Creates a new Payara / GlassFish 4 "domain" (/www/<PROJECT_SHORT_NAME>/clusterDomain/…) without using project specific settings in "HOME/config/conf.d". To configure the domain GFserver4 configure command has to be called.
`dump [--help]`	Generates a thread dump for DAS or any Payara / GlassFish 4 instance.
`encrypt-password [--help]`	Prompts for a password to encrypt (e.g. to use it for JDBC configuration settings.)
`end-rolling-upgrade [--help]`	Used to signal the end of a software change without down-time. **NOTE:** Not all kind of software changes can be done with-out downtime. The use of this command only guarantees no downtime if the software change is suited for this kind of changes.
`kill [[--admin|-a]|[-i N|--instance N]|[--all|-A]] [--help]`	Kills the specified Payara / GlassFish 4 instance(s). Tries to kill in a nice way (*kill -15*), if the process ignores the sig-nal, it will kill it with "*kill -9*".
`restore [--help]`	the same as 'create-domain' and afterwards 'configure' in just one command
`rolling-upgrade [--help]`	Used for a software change without downtime. **NOTE:** Not all kind of software changes can be done with-out downtime. The use of this command only guarantees no downtime if the software change is suited for this kind of changes.

`start [[--admin|-a]|[-i N|--instance N]|[--all|-A]] [--help]`	Starts the specified Payara / GlassFish 4 admin instance (DAS - with --admin or -a) or the i[0,1,2,…n] (managed) instance (with -i N or –instance N). NOTE: i[0,1,2,3,…n] instances will have to communicate with DAS to synchronize configurations at the first start. It is therefore necessary to start DAS before any first start of an instance to assure that the i[0,1,2,…n] instances always use the newest configuration.
`stop [[--admin|-a]|[-i N|--instance N]|[--all|-A]] [--help]`	Stops the Payara / GlassFish 4 admin instance (also called DAS) and the i[0,1,2,…n] managed instances. Details see at command 'start' described above.
`status [--help][--extended|-e]`	Provides status information of all Payara / GlassFish 4 instance(s). --extended | -e Show more details about processes.
`sync [--force]`	Syncs other shares with first share config directory content except for: *war files, *ear files, *and conf.d directory Also syncs directory *project-synced* between all the Payara / Glassfish 4 instances. When --force is used, even if your directory has more than 1GB amount of data, it will be synchronized. Please be aware that the synchronization of the data will take more time if the amount of data is large.
`remove-domain`	Will remove the content of the *clusterDomain* and *nodes* directories for all Payara/GlassFish 4 instances/nodes and ensure that no Java processes are running on any of the hosting servers for the respective Payara / GlassFish 4 instances, before removing.
`remove-config`	Will remove the contents of the *config* directory on the hosting server where DAS is running and ensure that no Java processes are running on any of the hosting servers for the respective Payara / GlassFish 4 instances, before removing. The *config/conf.d* subdirectory is only used for the *clusterDomain* creation and configuration by the staging tool and is not required on any other hosting server rather than the hosting server where DAS is running.
`remove-prechanges`	Will remove the *config/hooks/prechanges* hook script on the hosting server where DAS is running.

`remove-prerollingupgrade`	Will remove the *config/hooks/prerollingupgrade* hook script on the hosting server where DAS is running.
`remove-project-local`	Will remove the contents of the *project-local* directory on all hosting servers of the cluster.
`remove-project-synced`	Will remove the contents of the *project-synced* directory on all hosting servers of the cluster.
`extract-tar –f <tarball_name>`	Extract the “complete” content of the given “tarball” located under *tmp* directory
`extract-readme –f <tarball_name>`	Extract the README file from the given “tarball” located under *tmp* directory.
`extract-prechanges –f <tarball_name>`	Extract the prechanges hook script from the given “tarball” located under *tmp* directory.
`extract-prerollingupgrade –f <tarball_name>`	Extract the prerollingupgrade hook script from the given “tarball” located under *tmp* directory.
`Version`	Prints the version of Payara / Glassfish4, staging tool, GFserver4 script and jdk.

The arguments *start|stop|status|version|dump|kill[all]* may be used on any hosting server, whereas the remaining commands must be used on the hosting server where DAS is running.

The optional argument *-v* may be used up to three times (-vvv) before the other arguments to increase the verbosity of logging messages.

It will be possible to configure a project as “multihomed-project” if needed, without downtime. “Multihomed” means that it is possible to run several Payara / Glassfish 4 instances on one hosting server in a clustered environment allowing scale-up without having to host environments on additional hosting servers. To scale it down, you just need to not start this instances.

6 Troubleshooting

Common problems and error messages

Cannot parse xyz.xml

- Check the corresponding schema definitions in the XSD file found in */wwws/master_solution_gfv4/glassfish/staging/staging-x.x.x/projectConfig.xsd* and correct the XML file.

Configuration changes have no effect

- Did you issue a “GFserver4 restore or “GFserver4 configure/reconfigure command after changing the XML configuration files?
- Did you restart the Payara / Glassfish 4 instances after “GFserver4 reconfigure”
- Check for XML configuration items specified more than once in the configuration files.

7 Differences between the GlassFish2/3 and Payara /GlassFish 4 solutions

7.1 Project directory

The following list provides an overview regarding the main differences between the previous GlassFish 2,3 and Payara / GlassFish 4 based master solutions from a project configuration / staging point of view.

- Every Payara / GlassFish 4 instance will run on **its own** NAS based application server share (app-share). Previously the “app-share” has been shared among the GlassFish instances, now the “app-shares” are split between the GlassFish instances. The project directory “/www/<PROJECT_SHORT_CODE>/” will be the same on every hosting server. In fact the project directory will be a symbolic link to the mount point of the Payara / GlassFish 4 instance/node specific “app-share”.

Example:

```
livmapp401:/www/gf4pilot-multi # df -h .
Filesystem                                                      Size  Used Avail Use% Mounted on
nv01500766:/f01208/t026401/gf4pilot-multi-cchvak0s-i/app_node1  1.0G  268M  757M  27% /lfs/wwwmnt/gf4pilot-multi-cchvak0s-i-app_node1
livmapp402:/www/gf4pilot-multi # df -h .
Filesystem                                                      Size  Used Avail Use% Mounted on
nv01500766:/f01208/t026405/gf4pilot-multi-0aiab1n2-i/app_node2  1.0G  226M  799M  22% /lfs/wwwmnt/gf4pilot-multi-0aiab1n2-i-app_node2
```

- This change in the storage solution was decided by the web solution builder and operations team mainly for the following and very important reasons:
  - The Payara / GlassFish 4 solution must be **prepared for cloud scenarios** where shared storage between instances is “very uncommon”. The standard is “local/dedicated file storage” for every instance which normally shall not be used for web applications at all. Ideally web applications don’t rely at all to the underlying storage solution of the Payara / GlassFish 4 environment.
  - The Payara / GlassFish 4 solution must be **prepared for the next storage migration** from EMC to some other provider. It must be possible to migrate environments (one Payara / GlassFish 4 instance after the other) based on this solution without downtime to a different storage provider. File systems shared by all instances prevent such a downtime-less migration because it is impossible to migrate such file systems on one instance and later/after on the other instance.
- New directory “project-synced”
  - The content of the *project-synced* directory will be synchronized for every executed restore, create-domain and configure/reconfigure command via the *GFserver4* script.
  - It is possible to synchronize the content of the *project-synced* directory manually via the *GFserver4* sync command.

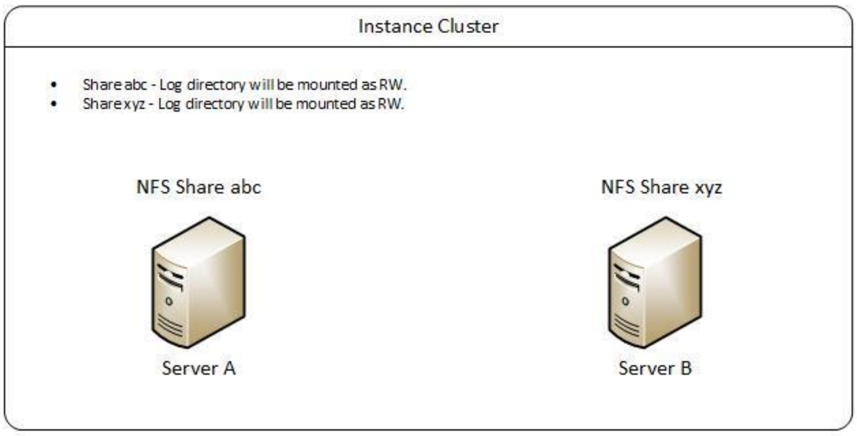
    - It is not recommended to have more than 1GB of data in the *project-synced* directory because of the delay which is created for doing the synchronization..
    - In case your web application requires more than 1GB for shared project data a shared share ("Shared Web Share" or "NAS (iStore) Application Share") has be used instead as solution for your web project.
- New directory "project-local"
    - The content of this directory will only be available for the "local" Payara / GlassFish 4 instance(s) which are running on **one** of the hosting servers. The content is not synced neither shared by all Payara / GlassFish 4 instance(s).
- New "directory" respectively "symbolic link project-shared"
    - This symbolic link is linked to the mount point for a shared NAS based file system. Makes sense when application files/data need to be shared between all the Payara / GlasFish 4 instances/nodes and a large amount of data (> 500 MB) on the respective shared share is needed. The *project-synced* directory mustn't be used for large data amounts!! Large amount of data can be stored on a so called "**Shared Web Share**" (<100 GB) or "**NAS (iStore) Application Share"** (Size limit according to NAS operations)**.** For more information regarding shared shares please refer to the storage documentation on the website "http://web.bmwgroup.net/" under the section "Additional Services".
- Missing directory "project-data"
    - This directory has been replaced via "project-synced" and "project-local". Depending on the use case on of these "new" directories may be used.
- The *config/conf.d* subdirectory, war and ear files will not be synchronized. Do **not** use config/conf.d subdirectory to store any special project specific data which needs to be available and equal for all Payara / GlassFish 4 instances.
- The way how log files must be accessed will change as well. Logfiles for a specific Payara / GlassFish 4 instance/node may only be found on the respective hosting server where this particular instance is running.
- Every action that will change project configurations in your environment must always executed on the hosting server where DAS is running.
- For more information regarding the new storage concepts please refer to: http://it.muc/rc/IT_Intranet/Services_en/webinfra/50_Web_services/Storage_operations/index.htm. (The documentation will be updated in January 2016.)

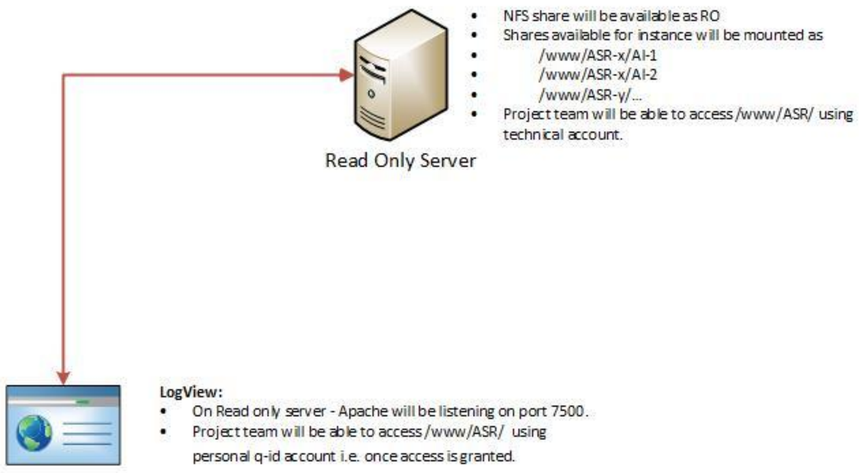
7.2 Software binaries Location

- Software available for use of your web application is located under "/wwws/master_solution_gfv4/" instead of "/wwws".
- Under this location you will find the Java software, the Payara / GlassFish 4 software as well as the JAAF software and drivers. The only software which is allowed to be used by Payara / GlassFish 4 will be located here. Do not use any software which is **not** located under "/wwws/master_solution_gfv4/".

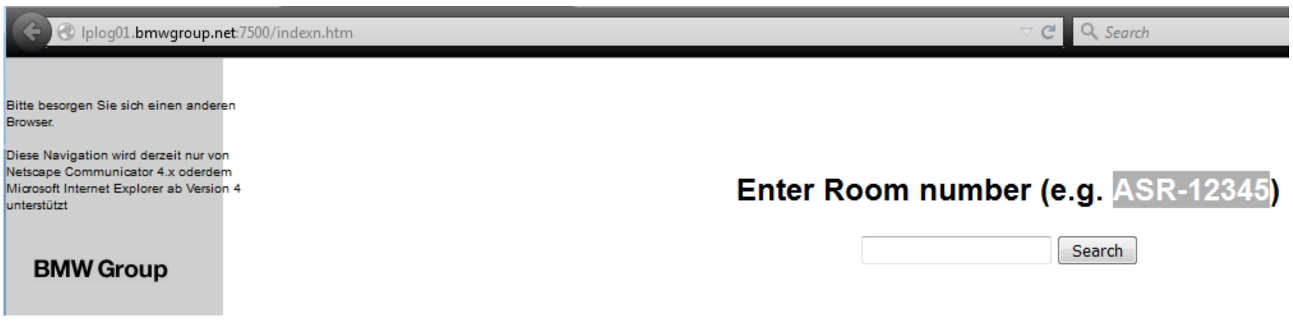
7.3 „LogFile Viewer" Changes:

- A new Logviewer concept/solution was introduced because of the "split storage" which is used for the application server shares of Payara / GlassFish 4 instances.
- As for previous Java EE master solutions all your project-shares will be mounted to special "ReadOnly servers" (lplog01.bmwgroup.net and lplog02.bmwgroup.net).
- The "Logfile Viewer web application" for Payara / GlassFish 4 instances is now running on these "ReadOnly servers". Previously it has been running on the hosting servers where the GlassFish instances are running.
- Your project-shares will be mounted under "/www/ASR-xxx/AI-xxx" (ASR-xxx and AI-xxx refers to the IDs of the configuration items which your web infrastructure received during provisioning of the environment. These IDs will be available via the Web Dashboard that can be accessed via the IT Self Service Portal (IT SSP) and the CMDB. .


Instance Cluster
• Share abc - Log directory will be mounted as RW.
• Share xyz - Log directory will be mounted as RW.
NFS Share abc
NFS Share xyz
Server A
Server B


• NFS share will be available as RO
• Shares available for instance will be mounted as
• /www/ASR-x/AI-1
• /www/ASR-x/AI-2
• /www/ASR-y/...
• Project team will be able to access /www/ASR/ using technical account.
Read Only Server
LogView:
• On Read only server - Apache will be listening on port 7500.
• Project team will be able to access /www/ASR/ using personal q-id account i.e. once access is granted.

- You will be able to access the log files on the ”ReadOnly servers” or via the “Logfile Viewer” web application Using the following mentioned URL:
  (To be substituted with a Balanced URL)
  http://lplog01.bmwgroup.net:7500/indexn.htm
  http://lplog02.bmwgroup.net:7500/indexn.htm
- On the “Logfile Viewer” web interface you must search for your ASR- or AI- IDs to access your log files.



Bitte besorgen Sie sich einen anderen Browser.

Diese Navigation wird derzeit nur von Netscape Communicator 4.x oderdem Microsoft Internet Explorer ab Version 4 unterstützt

BMW Group

Enter Room number (e.g. ASR-12345)

7.4 Commands Differences:

7.4.1 Initial Setup and configuration:

7.4.1.1 GFv2.1 (Master Solution V5.0)

GFv2.1 (Master Solution V5.0)	**Payara 4.1 / GlassFish 4**
Run `./GFserver stopall` **on each machine** of the cluster to stop all instances	Run *`./GFserver4 stop --all`* **on each hosting server** of the cluster to stop all instances
remove the directories *./clusterDomain* and *./nodeagents*	Run *`./GFserver4 remove-domain`* on the hosting server running DAS (Deletes the *clusterDomain* and *nodes* directory.)
Run `./GFserver recreate` on the machine running the DAS to create the domain and its configuration	Run *`./GFserver4 restore`* on the hosting server running DAS to “freshly” create the *clusterDomain* directory and its contents via DAS.
Run `./GFserver startall` **on each machine** (beginning with the admin server) to start the domain.	Run *`./GFserver4 start --all`* **on each hosting server** (beginning with the hosting server where DAS is running) to start all GlassFish4 instances of the project environment including DAS itself. **Important Note:** During the first start of a Glassfish 4 instance an “instance specific local domain” is created under *nodes* and the web application is downloaded from DAS. Therefore DAS needs to be running in order to be able to start the Glassfish4 instances for the first time.
Stop DAS *`./GFserver stopadmin`*	Stop DAS *`./GFserver4 stop --admin`*

7.4.1.2 GFv3.x (Master Solution V6.0)

GFv3.x (Master Solution V6.0)	**Payara 4.1 / GlassFish 4**
Run `./GFserver3 stop --all` **on each machine** of the cluster to stop all instances	Run *`./GFserver4 stop --all`* **on each hosting server** of the cluster to stop all instances
remove the directories *./clusterDomain* and *./nodes*	Run *`./GFserver4 remove-domain`* on the hosting server running DAS (Deletes the *clusterDomain* and *nodes* directory.)
Run `./GFserver3 restore` on the machine running the DAS to create the domain and its configuration	Run *`./GFserver4 restore`* on the hosting server running DAS to “freshly” create the *clusterDomain* directory and its contents via DAS.
Run `./GFserver3 start --all` **on each machine** (beginning with the admin server) to start the domain.	Run *`./GFserver4 start --all`* **on each hosting server** (beginning with the hosting serv-er where DAS is running) to start all GlassFish4 instances of the project environment including DAS itself. **Important Note:** During the first start of a Glassfish 4 instance an “instance specific local domain” is created under *nodes* and the web application is downloaded from DAS. Therefore DAS needs to be running in order to be able to start the Glassfish4 instanc-es for the first time.
Stop DAS *`./GFserver3 stop --admin`*	Stop DAS *`./GFserver4 stop --admin`*

7.4.2 Reconfiguration:

7.4.2.1 GFv2.1 (Master Solution V5.0)

GFv2.1 (Master Solution V5.0)	**Payara 4.1 / GlassFish 4**
Run `./GFserver reconfigure` on the machine running the DAS to reconfigure the domain.	Run *`./GFserver4 configure`* on the hosting server running DAS to create the *clusterDomain* directory and its contents.
Run `./GFserver stopall` **on each machine** of the cluster to stop all instances	Run *`./GFserver4 stop --all`* **on each hosting server** to stop all Payara / GlassFish 4 instances including DAS.
Run `./GFserver startall` **on each machine** (beginning with the admin server) to start the domain.	Run *`./GFserver4 start --all`* **on each hostingserver** (beginning with DAS) to start all Payara / GlassFish 4 instances including DAS.
Stop DAS *`./GFserver stopadmin`*	Stop DAS *`./GFserver4 stop --admin`*

7.4.2.2 GFv3.x (Master Solution V6.0)

GFv3.x (Master Solution V6.0)	**Payara 4.1 / GlassFish 4**
Run `./GFserver3 configure` on the machine running the DAS to create the domain and its configuration	Run *`./GFserver4 configure`* on the hosting server running DAS to create the *clusterDomain* directory and its contents.
Run `./GFserver3 stop --all` **on each machine** of the cluster to stop all instances	Run *`./GFserver4 stop --all`* **on each hosting server** to stop all Payara / GlassFish 4 instances including DAS.
Run `./GFserver3 start --all` **on each machine** (beginning with the admin server) to start the domain.	Run *`./GFserver4 start --all`* **on each hostingserver** (beginning with DAS) to start all Payara / GlassFish 4 instances including DAS.
Stop DAS *`./GFserver3 stop --admin`*	Stop DAS *`./GFserver4 stop --admin`*

7.4.3 Deployment

7.4.3.1 GFv2.1 (Master Solution V5.0)

GFv2.1 (Master Solution V5.0)	**Payara 4.1 / GlassFish 4**
If it exists, run the pre-change script `./GFserver exec con-fig/hooks/prechange`	If it exists, run the prechange script (optional): *`./GFserver4 exec -f con-fig/hooks/prechange`*
Stop the application server by executing this command on each machine of the cluster: `./GFserver stopall`	Stop the Payara / GlassFish 4 instance(s) and DAS by executing the following command on each hosting server of the cluster: *`./GFserver4 stop --all`*
Clean old configuration directory: `rm -r config/*`	Clean currently existig *config* directory on the hosting server where DASis running: *`./GFserver4 remove-config`*

	Clean the content of the directories *project-local* and/or *project-synced* if you **need to delete** con-tent in these directories (optional):
	`./GFserver4 remove-project-local`
	`./GFserver4 remove-project-synced`

Unpack the tarball: `tar xf $tarball config/ project-data/ htdocs/`	Extracts the “tarball” located under *tmp/<tarball_filename>*, this will extract only the directories *config*, *project-synced* if they are contained in the “tarball” (everything else which is in the “tarball” will be ignored by this com-mand) : *`./GFserver4 extract-tar –f <tar-ball_filename>`*
If it exists, run the change script `./GFserver exec con-fig/hooks/change`	If it exists, run the change script (optional): *`./GFserver4 exec -f con-fig/hooks/change.`*
Remove old domain files: `rm -r nodeagent clusterDomain`	Remove “old” *clusterDomain* and *nodes* (con-tains node specific domains) directories: *`./GFserver4 remove-domain`*
Re-create domain `./GFserver recreate`	Recreate the *clusterDomain* directory and its con-tents. *`./GFserver4 restore`* (Note: This will implicitly start and stop DAS)

Start the application server by executing this command on each machine of the cluster (beginning with the admin server): `./GFserver startall`	Start the application server instances by executing the following command on each hosting server of the cluster (beginning with the hosting server where DAS is running): *`./GFserver4 start --all`*
If it exists, run the post-change script `./GFserver exec –f config/hooks/postchange`	If it exists, run the postchange script *`./GFserver4 exec -f config/hooks/postchange`*
Stop DAS *`./GFserver stopadmin`*	Stop DAS *`./GFserver4 stop --admin`*

7.4.3.2 GFv3.x (Master Solution V6.0)

GFv3.x (Master Solution V6.0)	**Payara 4.1 / GlassFish 4**
If it exists, run the pre-change script `./GFserver3 exec -f config/hooks/prechange`	If it exists, run the prechange script (optional): *`./GFserver4 exec -f config/hooks/prechange`*
Stop the application server by executing this command on each machine of the cluster: `./GFserver3 stop --all`	Stop the Payara / GlassFish 4 instance(s) and DAS by executing the following command on each hosting server of the cluster: *`./GFserver4 stop --all`*
Clean old configuration directory: `rm -r config/*`	Clean currently existig *config* directory on the hosting server where DASis running: *`./GFserver4 remove-config`*

	Clean the content of the directories *project-local* and/or *project-synced* if you **need to delete** content in these directories (optional):
	`./GFserver4 remove-project-local`
	`./GFserver4 remove-project-synced`

Unpack the tarball: `tar xf $tarball config/ project-data/ htdocs/`	Extracts the “tarball” located under *tmp/<tarball_filename>*, this will extract only the directories *config*, *project-synced* if they are contained in the “tarball” (everything else which is in the “tarball” will be ignored by this command) : *`./GFserver4 extract-tar –f <tarball_filename>`*

If it exists, run the change script: `./GFserver3 exec -f con-fig/hooks/change`	If it exists, run the change script (optional): *`./GFserver4 exec -f con-fig/hooks/change.`*
Remove old domain files: `rm -r nodes clusterDomain`	Remove "old" *clusterDomain* and *nodes* (con-tains node specific domains) directories: *`./GFserver4 remove-domain`*
Re-create domain `./GFserver3 restore`	Recreate the *clusterDomain* directory and its con-tents. *`./GFserver4 restore`* (Note: This will implicitly start and stop DAS)
Start the application server by executing this command on each machine of the cluster (be-ginning with the admin server): `./GFserver3 start --all`	Start the application server instances by execut-ing the following command on each hosting server of the cluster (beginning with the hosting server where DAS is running): *`./GFserver4 start --all`*
If it exists, run the post-change script ./GFserver3 exec -f *config/hooks/postchange*	If it exists, run the postchange script *`./GFserver4 exec -f con-fig/hooks/postchange`*
Stop DAS *`./GFserver3 stop --admin`*	Stop DAS *`./GFserver4 stop --admin`*

7.4.4 Rolling Upgrade:

7.4.4.1 GFv2.1 (Master Solution V5.0)

GFv2.1 (Master Solution V5.0)	**Payara 4.1 / GlassFish 4**
delete prerolling-upgrade rm -f config/hooks/prerolling-upgrade	Execute “remove-prerolling-upgrade” (optional) `./GFserver4 remove-prerolling-upgrade`
Unpack the README and the (optional) new prerolling-upgrade script: tar xf $tarball config/hooks/prerolling-upgrade config/README	Extract the README file and the (optional) pre-rolling-upgrade script: `./GFserver4 extract-readme –f <tarball_filename>` `./GFserver4 extract-prerolling-upgrade –f <tarball_filename>` *(optional)*
Read the README file	Read the contents of the README file
Execute config/hooks/prerolling-upgrade: `./GFserver exec config/hooks/prerolling-upgrade`	Execute “config/hooks/prerolling-upgrade” (optional) `./GFserver4 exec -f config/hooks/prerolling-upgrade`
Unpack the tarball: tar xf $tarball config/ project-data/ htdocs/	Extract the “tarball” this will extract only the directories *config*, *project-synced* and *project-local* if they are contained in the “tarball” (everything else which is in the “tarball” will be ignored by this command): `./GFserver4 extract-tar –f <tarball_filename>`
Execute config/hooks/rolling-upgrade `./GFserver exec config/hooks/rolling-upgrade`	Execute config/hooks/rolling-upgrade (optional) `./GFserver4 exec -f config/hooks/rolling-upgrade`
reconfigure domain `./GFserver rolling-upgrade`	Start with rolling-upgrade mode and reconfigure existing *clusterDomain* `./GFserver4 rolling-upgrade`
restart one by one, starting with admin server `./GFserver stopall startall status`	Restart one Payara / GlassFish 4 instance after the other starting with the hosting server where DAS is running `./GFserver4 stop --all start --all status`
finish rolling-upgrade mode `./GFserver end-rolling-upgrade`	Finish rolling-upgrade mode `./GFserver4 end-rolling-upgrade`

If it exists, run the postrolling-upgrade script `./GFserver exec config/hooks/postrolling-upgrade`	If it exists, run the postrolling-upgrade script (optional) *`./GFserver4 exec -f config/hooks/postrolling-upgrade`*
stop Adminserver `./GFserver stopadmin`	Stop DAS *`./GFserver4 stop --admin`*

7.4.4.2 GFv3.x (Master Solution V6.0)

GFv3.x (Master Solution V6.0)	**Payara 4.1 / GlassFish 4**
delete prerolling-upgrade rm -f config/hooks/prerolling-upgrade	Execute “remove-prerolling-upgrade” (optional) *`./GFserver4 remove-prerolling-upgrade`*
Unpack the README and the (optional) new prerolling-upgrade script: tar xf $tarball config/hooks/prerolling-upgrade config/README	Extract the README file and the (optional) prerolling-upgrade script: *`./GFserver4 extract-readme -f <tarball_filename>`* *`./GFserver4 extract-prerolling-upgrade -f <tarball_filename>`* *(optional)*
Read the README file	Read the contents of the README file
Execute config/hooks/prerolling-upgrade: `./GFserver3 exec -f config/hooks/prerolling-upgrade`	Execute “config/hooks/prerolling-upgrade” (optional) *`./GFserver4 exec -f config/hooks/prerolling-upgrade`*
Unpack the tarball: tar xf $tarball config/ project-data/ htdocs/	Extract the “tarball” this will extract only the directories *config*, *project-synced* and *project-local* if they are contained in the “tarball” (everything else which is in the “tarball” will be ignored by this command): *`./GFserver4 extract-tar -f <tarball_filename>`*
Execute config/hooks/rolling-upgrade `./GFserver3 exec -f config/hooks/rolling-upgrade`	Execute config/hooks/rolling-upgrade (optional) *`./GFserver4 exec -f config/hooks/rolling-upgrade`*
reconfigure domain `./GFserver3 rolling-upgrade`	Start with rolling-upgrade mode and reconfigure existing *clusterDomain* *`./GFserver4 rolling-upgrade`*

restart one by one, starting with admin server `./GFserver3 stop --all start --all status`	Restart one Payara / GlassFish 4 instance after the other starting with the hosting server where DAS is running *`./GFserver4 stop --all start --all status`*
finish rolling-upgrade mode `./GFserver3 end-rolling-upgrade`	Finish rolling-upgrade mode *`./GFserver4 end-rolling-upgrade`*
If it exists, run the postrolling-upgrade script `GFserver3 exec –f config/hooks/postrolling-upgrade`	If it exists, run the postrolling-upgrade script (optional) *`./GFserver4 exec –f config/hooks/postrolling-upgrade`*
stop Adminserver `./GFserver3 stop --admin`	Stop DAS *`./GFserver4 stop --admin`*

8 Reference

8.1 Documentation on the BMW intranet

- Master Solution Java Application Server Premium on BMW intranet:
  java.muc > Master Solutions
- Staging tool on BWM intranet:
  http://java.muc/staging/
- Web Operations on BMW intranet:
  web.bmwgroup.net
- Solution Buildings Blocks:
  java.muc > Solution Building Blocks
- Apache Webserver
  http://apache.bmwgroup.net/
- Dispatcher
  http://dispatcher.bmwgroup.net/
- Reverse Proxy
  http://reverseproxy.bmwgroup.net/