Developer Manual

Staging process

Version:	*1.12*
Date:	*12.05.2014*
Status:	*released*
Author:	*Dietmar Neuss*
File:	*WebLogic12-appdev-manual-1.12.docx*

Amendment log:

Version	Date	Authors	Changes
1.0	21.09.2012	Dietmar Neuß	Initial release
1.1	15.01.2013	Dietmar Neuß	Some changes and formatting
1.2	31.01.2013	Dietmar Neuß	Some improvements
1.3	20.02.2013	Dietmar Neuß	Added description of configure
1.4	18.03.2013	Dietmar Neuß	Added note where to define private variables
1.5	19.03.2013	C. Schaarschmidt	Review
1,6	08.04.2013	Dietmar Neuß	Some enhancement
1.7	11.04.2013	C.Schaarschmidt	Refactoring
1.8	25.04.2013	D. Neuß	Changes at description of ‘deploytar’, new version of ‘config-version’ and hints of special-agreements
1.9	15.10.2013	D. Neuß	Added new command to WLserver12
1.10	30.10.2013	C. Schaarschmidt	review
1.11	17.12.2013	H. Spindler	Finalization for Publishing
1.12	12.05.2014	C. Schaarschmidt	Include feedback from pilots and Mr. Niklasch

Table of content

1 Introduction

This document describes the optional staging process in the Master Solution Java Application Server Premium WLS 12 environment at BMW and how to prepare a web application accordingly. It is intended primarily for the developers of these applications. Prior to reading this document, it is recommended to read the documentation of the master solution first. The documentation can be found on the master solution web site under "Documentation / External view".

The staging process is now being introduced with new version WLS 12 of the master solution for application servers. Previous WLS based master solutions used a different process, which can still be used and is documented at the end of 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 describes the procedures for deployment without using the staging commands.

Chapter 7 provides additional information about users and security files.

Chapter 8 contains some hints for troubleshooting.

2 Overview

2.1 Staging process and staging commands

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

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

Deployments to each of these stages are performed using the staging commands. These commands streamline the process of bringing a project and its configuration to the various stages successively.

Please note that the use of the tool is **not** mandatory but it is recommended.

2.2 Principle of operation

The main feature of the staging tool is the separation of configuration options provided by operations on one hand and those provided by the application developers on the other hand. Application configuration 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 and its configuration files are packed into an archive file, the so-called "tarball". The tarball is then used for deployments on the targeted stage. By splitting the configuration in a general part and a stage specific part, the tarball can contain the configuration settings for all stages. Once properly configured, tarballs can be deployed to all stages without further customizing.

2.3 Benefit

The procedure mentioned above grants a number of benefits:

- Using the generated tarball for the targeted stage ensures that the project goes into production with the same settings it has been tested with
- The possibility to reset a domain to default values and start with a fresh configuration
- Reduce risk of faulty tarballs by providing “WLserver12 maketar” command

3 Usage

Note: All path names of commands, files and directories in this chapter are relative to the WLS12 project home directory, for example /www/<myproject>/...

3.1 Initial Setup of a Project

In order to bring a project into the staging process for the first time, the following steps are necessary. (This initial setup is usually taking place on the test stage.)

1. Web operations provides a fresh application server environment with default settings
2. Edit `~/config/WLserver12.var` as required. Changes to Variables should be added at the end or may be lost during re-creation. Any private variables have to be defined **at the end** of 'WLserver12.var'! (and exported if used outside the script)
3. As a recommendation application developers copy the project's EAR/WAR-file into the *`config/applications`* directory on the application server.
4. Application developers configure the project settings on the server (see chapter “Change environment”) via admin console (https://<hostname>/<port>/console). Projects can use the user 'system' for this purpose.

3.2 Recreation of a Project / Factory Reset

In order to recreate a domain you may perform

`./WLserver12 create-domain.`

This will bring config.xml into initial state. Only WLserver12.var will keep its information. In order to perform a recreation of a domain (no deployment will be done) directory 'clusterDomain' has to be deleted beforehand.

3.3 Change environment

In order to change the environment (here in our example from test to integration)

1. in test environment:
   `./WLserver12 maketar`

   This will create a tarball for changes in any environment. If needed, the project can include and exclude files. Therefore write comma-separated list **at the end** of `'WLserver12.var'` with shell-wildcards:
   CUSTOM_INCLUDE_TAR_FILES="example.xxx myExample/*"
   CUSTOM_EXCLUDE_TAR_FILES="absurdity.xxx myExample/needless"

   Please note that all private variables have to be defined **at the end** of 'WLserver12.var'!

2. followed by (optional)
   `./WLserver12 previewtar <tarFile>` (previews the content of a fictitious tar-file if it would have been created with `'maketar'`)

3. in target stage (in this example in integration environment (created by operations)) perform the following steps:

- `./WLserver12 stop -A`
- `./WLserver12 deploytar <tarFile>`
  (**Note**: in productive environment, this will be done by operations!)

  This command first will extract the tarball into an already existing environment and create sym-links to **all** files and **all** directories based on the target-environment (e. g. config.xml → config.xml.INT).
  Next a `./WLserver12 configure` will be automatically executed thus adjusting `clusterDomain/config/config.xml` to current stage (all hostnames, ports etc.)
- `./WLserver12 start -A`

3.4 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. Use the same naming-concept as described in chapter 6.
   (if you follow the naming-concept, deploytar will create sym-links to files based on the target environment, e.g. `config.xml -> config.xml.INT` in integration)
2. Configure performance-relevant settings.
3. Test the tarball 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 may check the tarball (and possibly reject it), but is not required to do so.

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

3.5 Caveats

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

- The application server must be started and stopped via the `WLserver12` command.
- The commands `WLserver12 [make|deploy|preview]tar`, `WLserver12 create-domain` and `WLserver12 configure` must be executed on the node running the admin server.
- Environment variables (e.g. JAVA paths) will be ignored when running the `WLserver12` command. They will **not** be carried over to the WebLogic environment.

- Entries for non-standard configuration on items like
  “CUSTOM_PATCH”
  need documentation of special-agreement number in `'.WLserver12.extra'` and will be ignored unless properly documented by web-operations.

4 Configuration

4.1 Directory structure

The following directories within the WLS12 project home directory are relevant for application developers:

$HOME/logs/project

Directory for project log files.

$HOME/project-data

Directory to store project specific data.

$HOME/htdocs (optional)

Symbolic link to the mount point for a shared file system. Used in the (rare) case of content being generated by app-server and delivered by web server. Created only upon request.

$HOME/clusterDomain

Automatically created by the staging tool. Must be deleted before running `WLserver12 create-domain`.

$HOME/config

The main point of interest for application developers. All files required for configuration must be located within this file structure. It contains

- project configuration files
- additional libraries (optional)
- additional scripts for deployment (optional)

$HOME/config/applications

Here you should place project's EAR/WAR files.

$HOME/config/trust.d (optional)

Optional directory for trust certificates. If the directory exists and is not empty, a default trust store named *`generated/trust_v7.jks`* will be created and loaded with the certificates from `'conf.d'`. The certificates must be in a format that the Java keytool accepts (`*.pem` or `*.cer`). The alias name is derived from the filename by stripping the suffix (e.g. filename *`123456.pem`* becomes the alias *`123456`*).

$HOME/config/key.d (optional)

Project with additional requirements (which cannot be addressed with central files), may provide their own certificates. Certificates (files found with pattern `*.pem`) stored in `<project-home>/config/key.d` are added by `WLserver12` to the keystore.

$HOME/generated

Any generated password or certificate will be stored here.

4.2 Configuration items

4.2.1 Standard configuration items

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

- Heap sizes
- JAAF
- Application
- JDBC
- Java Mail
- MQ
- JMS
- Classpath
- JVM options
- Staging tool / WebLogic / JVM version

4.2.2 Non-standard configuration items

Heap sizes may only be configured within certain limits defined by web operations. These limits can be raised upon request.

Another non-standard items are “CUSTOM_PATCH'. Here are special-agreements required and have to be documented in '`.WLserver12.var`' by web-operations. Those configuration items will be ignored until properly documented.

4.2.3 Staging tool version

The staging tool version (config-version) defines a set of default configuration settings, e.g. WebLogic version, JVM version, Java parameters etc.. Web operations may update default settings or scripts from time to time. The initial setting of the config-version is done in `$PROJECT_HOME/WLserver12.operations`. In order to benefit from optimized defaults and new features, the project has to set the “staging tool” version in the configuration element *`config-version`* in `$PROJECT_HOME/config/WLserver12.var`

This document describes the staging tool versions from `v70:20130416` onward. If you are running an older version, upgrading to a current release is strongly recommended.

5 Tool Reference

The script `WLserver12` is the front end of the staging tool. The syntax is:

```
./WLserver12 [start|stop] --iN|-a
./WLserver12 create-domain
./WLserver12 version
```

Argument	Effect
`configure`	Configures a domain (adjustment of `./clusterDomain/config/config.xml` to the current stage). **NOTE:** can only be used if a domain has already been created.
`create-domain`	Creates a clean domain (without deployment) using the content of the files • `.Base.boot`, • `.Base.dat`, • `.WLserver12.extra` and • `.WLserver12.operations`. The user must stop all server instances and delete the directory *`clusterDomain`* beforehand. The domain must have been created once.
`deploytar <tarFile>`	1. Extract from `<tarFile>` 2. create sym-links for target stage: for **all** files and **all** directories that adhere to the naming convention `<filename>.[TEST|INT|PROD|APPROVAL]` a symlink `<filename>` → `<filename>.[TEST|INT|PROD|APPROVAL]` will be created 3. configures domain (adjusting `config.xml` to the current stage, replaces hostnames in `WLserver12.var` and so on).
`dump -iN| -a| -A`	create a thread dump of the instance N (`-iN`) or WebLogic adminserver (`-a`) or all (`-A`)
`encrypt-password [--string <StringToEncrypt>]`	Prompts for a password to encrypt (e.g. to use it for JDBC configuration settings.
`exec-WLST <script>`	Executes `<script>` with WebLogic Scripting Tool
`get-agreement`	Displays the current special agreements
`get-cp`	Displays the currently used `CLASSPATH`
`get-custom`	The customized configuration can be displayed
`get-extra`	displays special agreements and limits
`get-jvm -iN| -a| -A`	The current used JVM parameters and options for instance N (`-iN`) or all instances (`-A`) or admin-server (`-a`) can be displayed
`get-limit`	The operation-defined limits can be displayed

Argument	Effect
`get-valid`	All valid config-versions can be displayed
`help [command]`	Prints information
`info-store`	List the content of the keystore
`info-trust`	List the content of the trust store
`make-trust`	generate a new into generated/ this is useful if you need to add trust anchors not included in our standard trust store.
`maketar`	Creates tarfile Project can include or exclude own files. (comma separated list with shell-wildcards) e. g. CUSTOM_INCLUDE_TAR_FILES=”example.xxx myExample/*" CUSTOM_EXCLUDE_TAR_FILES="absurdity.xxx myExample/needless"
`ps`	ps for all jvms on this domain
`previewtar`	previews the content of a fictitious tar-file as it would be created by `maketar`
`restart -iN| -a| -A`	Stop and then start instance `(-iN)`, admin-server `(-a)` or all servers on this host `(-A)`

`start -iN| -a| -A`
`[-v]`
`[-C]`
	Starts the specified WebLogic instance (`-iN`) or admin-server (`-a`) or all (admin-server and all nodes -> `-A`). NOTE: instances will communicate with WebLogic Adminserver to synchronize configurations. It is therefore necessary to start the Adminserver before any instance, to assure that the instance(s) always use the newest configuration.
	Option '`-C`' will clear the cache and the temporary deployment (formerly known as 'rmtmp').
	Temporary deployment is stored in
	`<projectHome>/clusterDomain/servers/<wlsName>/stage`
	`<projectHome>/clusterDomain/servers/<wlsName>/tmp`
	`<projectHome>/clusterDomain/servers/<wlsName>/data`
	`<projectHome>/clusterDomain/servers/<wlsName>/cache`

`stop -iN| -a| -A`	Stops the WebLogic instance (`-iN`) or admin-server (`-a`) or all (`-A`). Details see at command '`start`' described above.
`status [-iN|-A]`	Provides status information of WebLogic instance(s).
`version [-v]`	Print version information for WLserver12, staging, config-

Argument	Effect
	version, JVM and application server. With option '`-v`' some update information will be displayed.

The arguments *`start|stop|dump|kill`* may be used on any node, whilst *create-domain* and *configure* and *deploytar* must be used on the node running the admin server.

The optional argument '`-v`' may be used to increase verbosity of logging messages.

6 Deployment without using the staging commands

As mentioned above the “old” way of deployment (without usage of “staging” commands) is still valid but not recommended.

6.1 Requirements for tar files

If you don't want to use the staging tool 'maketar' (see No. 3.3 or No. 5) you can create tar-files by your own. Tar files should exclude dispensable data (e.g. temporary deployments, logs, coredumps, backups, etc ...). A tar file should always contain a readme file (README, readme.txt, ...) with all instructions for the web-operations team to complete the softwarechange.

There are some prerequisites for tar files:

- files and directories must have the correct ownership (project user and project group)
- the main control scripts (`WLserver12`) must be a symbolic link to `/wwws/weblogic/webops/WLserver12`
- the tar file MUST be created in the project directory (`/www/myproject/`)

6.2 Creating tar files for software changes and go-live

6.2.1 Common procedure

There are two main ideas how to prepare tar-files for production:

- final configuration tar-file
- applicable configuration tar-file

6.2.1.1 Final configuration tar-file

The project provides the final main-configuration files for production itself (config.xml, WLserver12.var, jdbc_configuration.xml, jms_configuration.xml, etc.) so that the web-operations team only needs to extract the tar archive in production and restart the application server instances without making any changes like copying files. Additional tasks can be described in the readme-file.

6.2.1.2 Applicable configuration tar-file

The project keeps for each environment the needed configuration files with the proper suffix, e.g.:

- `config/WLserver12.var.PROD`
- `config/WLserver12.var.INT`
- `config/WLserver12.var.TEST`

- `clusterDomain/config/config.xml.PROD`
- `clusterDomain/config/config.xml.INT`
- `clusterDomain/config/config.xml.TEST`

- `clusterDomain/config/jdbc/my-jdbc-pool.xml.PROD`
- `clusterDomain/config/jdbc/my-jdbc-pool.xml.INT`
- `clusterDomain/config/jdbc/my-jdbc-pool.xml.TEST`

- `clusterDomain/config/jms/my-jms-pool.xml.PROD`
- `clusterDomain/config/jms/my-jms-pool.xml.INT`
- `clusterDomain/config/jms/my-jms-pool.xml.TEST`

The readme-file has to contain the step of copying the production files to the main filename, without the suffix (e. g. '`cp config.xml.PROD config.xml`'). Of course the project also can provide a shell script to execute these steps.

6.2.2 Modifications to configuration

When preparing tar files for changing the environment, some settings have to be modified in the configuration files by the project, e.g. server hostnames. These changes can be made in the way the project prefers (text-editor search and replace, UNIX command sed, etc.). This is a list of settings that has to be considered:

- replace hostnames in `config/WLserver12.var` (ADMIN_URL and LOCAL_SERVER)
- replace hostnames in `clusterDomain/config/config.xml`
- (optionally) adjust JDBC/JMS/etc. settings (hostname, port, …) in `clusterDomain/config/jdbc/…`, `clusterDomain/config/jms/…`
- (optionally) project specific settings

6.2.3 Go-live tar-file

When going live, you have to provide a full tar to web-operations – that means the whole configuration and deployment must be included. The following directories have to tar-ed on your integration environment:

- `config/` (including *.production files: `config.xml`, `jms`, `jdbc`, …)
  - `WLserver12.var*`
  - `applications/`
  - `jaaf/` (optional)
  - `lib/` (optional)
  - `patches/` (optional)
- `clusterDomain/servers/AdminServer/data/ldap/` (required if users/groups have been created for application use in the internal LDAP server of WebLogic in integra-

tion. Otherwise these groups/users will be missing in the other environment). As a suggestion project delivers file '`*.ldift`' which contains project-specific user. Those user have to belong to group '`bmwgroup`'!

- additional project specific files/directories e.g. `project-data/`

6.2.4 Software change tar-file

We recommend using full tar balls for changes.

The advantage of full tar balls is that you know always the exact version of all your files on PROD and you can test this combination in INT before handing the tar-file over to operations. This tar-file can later also be used to create test environments to investigate problems observed in PROD

7 Background Information

Information in this chapter is provided for interested parties to give more insight. If you are not interested, you can skip this chapter.

7.1 WebLogic-User

During installation the following (WebLogic)-users are created

- `system`
- `webops`
- `webControl`

7.1.1 system

This user is used by the project. In TEST- and INT this user is in group '`administrators`' with all permissions. In APPROVAL and PROD the group membership is being changed to '`monitors`' and therefore the user's permissions are set to 'read-only'.

During installation a randomly generated password will be created and stored in '`~/config/system-password`'. Changes to this file will take effect after the next create-domain.

7.1.2 webops

This user is used by web operations if admin console is needed. This user has all permissions in all environments! This user must not be modified or deleted!

7.1.3 webControl

This user is a technical user and is used to edit/read/config the file '`.Base.boot` ' in all environments. This user must not be modified or deleted!

7.2 Security background

The first room ordered for an APP-ID is called "origin". All remaining rooms ordered for the same App-ID are derived from this first installation. All rooms derived from the same origin share the same security information and tarballs can be deployed in any of them. However, tarballs created in environments from different origins, will fail to deploy, WebLogic will not be able to decrypt the information contained in the tarball, e.g. encrypted passwords.

Security information is stored in

- `.Base.dat`
  (saved `clusterDomain/security/SerializedSystemIni.dat` and contains key for encryption)
- `.Base.ldift`
  (saved `clusterDomain/security/DefaultAuthenticatorInit.ldift` and contains internal ldap-user configuration)

- `.Base.xml`
  (basic `clusterDomain/config/config.xml` and will contain base security informations)
- `.Base.boot`
  (saved `clusterDomain/servers/AdminServer/security/boot.properties`)

Those files have to be identical in all stages and are created the first time at 'first' (e. g. TEST) stage installation (base=origin).

8 Troubleshooting

Common problems and error messages

Configuration changes have no effect

- Did you clear the cache and restarting the server by performing "`WLserver12 start -A -C`" after changing the configuration?
- Check for configuration items specified more than once in the configuration files.

9 Reference

9.1 Documentation on the BMW intranet

Master Solution Java Application Server Premium WLS 12 on the BMW intranet:

http://java.muc/ > MasterSolutions