**Production Deployuri One Pager**

**1. Introduction**

**1.1. Project/Component Working Name:**

Support a new command to deploy the application by accepting the path as URI

**1.2. Name(s) and e-mail address of Document Author(s)/Supplier:**

Jeremy\_Lv: [lvsongping@cn.fujitsu.com](mailto:lvsongping@cn.fujitsu.com)

**1.3. Date of This Document:**

02/26/2013

**2. Project Summary**

**2.1. Project Description:**

Support a new command to deploy the application by accepting the path as URI

**2.2. Risks and Assumptions:**

This feature is done by external contributor Jeremy and there is certain risk associated with external contributors.

**3. Problem Summary**

**3.1. Problem Area:**

To allow user deploy with a URI directly instead of doing a two-step process, obtain the file associated with the URI and then deploy.

**3.2.** **Justification:**

The feature is more of ease of use, a convenience for avoiding a separate download step for an application that is available at a URI.

**4. Technical Description:**

**4.1. Details:**

**4.1.1. Expected behavior**

**4****.1.1.1. DAS mode**

With production deployuri feature, the user will be able to deploy the application by accepting the path as a URI type. Some of the situations that will be realized in the DAS mode will be as follows:

1. Deploy the local application to the local server: This syntax is only aimed at the file:// syntax, the admin client will convert the file:// syntax to File syntax, and then the server will handle the process similar to the deploy command.
2. Deploy the remote application to the local server: This syntax is aimed at the http:// syntax and ftp:// syntax, the admin client will download the remotely application to the local temp directory(i.e: $Users\AppData\Local\Temp), then the server will accept the temp file address and handle the process similar to the deploy command.

1. Deploy the remote application to the remote server: When the application server is remotely and we need to specify the host address, the admin client will download the application to the temp directory first and then the remote server will handle the process similar to the existing remote deploy command.

**4.1.1.2. Standalone Instance mode/cluster mode**

The deployuri command support the application to be deployed to the instance/cluster, there exist many situations as follows:

1. Deploy the local application to the instance delivered by local server: This syntax is only aimed at the file:// syntax, the local admin client will convert the file:// syntax to File syntax, Then the application will be deployed to the sever, finally, the local server will deliver the application to the instance/cluster which is need to be deployed.
2. Deploy the local application to the instance delivered by the remote server: This syntax is only aimed at the file:// syntax, the local admin client will convert the file:// syntax to File syntax, Then the application will be deployed to the remote server, finally, the remote server will deliver the application to the instance/cluster which is need to be deployed.
3. Deploy the remote application to the instance/cluster delivered by local server: Firstly, the admin client will handle the URI and download the application to the temp directory. Secondly, the admin client will send the temp directory to the local server and the local server will handle the process to deploy the application. Finally, the local server will deliver the application to the related instance/cluster to make sure the application to be deployed to the instance.
4. Deploy the remote application to the instance/cluster delivered by remote server: Firstly, the admin client will handle the URI and download the application to the temp directory. Secondly, the admin client will send the temp directory to the local server and the remote server will handle the process to deploy the application. Finally, the remote server will deliver the application to the related instance to make sure the application to be deployed to the instance/cluster.

**4.1.2. Restricted features**

* --upload: The option of --upload is only support to the file syntax.
* We needn’t support the proxy option because it can be set into the domain.xml if any users need to set it.

**4.1.3. Reason why support a new command**

1. As the original syntax about deploy command should be support, It is hard to reuse it to declare a command parameter which will both support the File and URI. It is because the file path can only accept one type of parameter.
2. The new command will keep things simple. It doesn’t need to change any codes related to the deploy command. We can define a new type of command parameter to support the URI syntax.

**4.1.4. Support use case examples**

**4.1.4.1. Context of examples**

For the command of deployuri to be applied, there exist some tips for the syntax as follows:

* The URI resource of the application must be exists. (When it comes to the )
* The suffix about the URI must be ended with the .jar, .war, .ear or .rar to make sure the application can be downloaded to the temp dist exactly. (This situation is only related to the http protocol and ftp protocol)
* It will support the application to be deployed as directory when it comes to the file:// syntax.

|  |
| --- |
| $ asadmin deployuri file:/e:/test.war |

|  |
| --- |
| $ asadmin deployuri　–-host host\_address　file:/e:/test.war |

|  |
| --- |
| $ asadmin deployuri　–-host host\_address　file:/e:/test\_dir/apps |

|  |
| --- |
| $ asadmin deployuri ftp://username:password@host:21/test.war |

|  |
| --- |
| $ asadmin deployuri  http://java.net/jira/secure/attachment/50467/test\_sample1.war |

|  |
| --- |
| $ asadmin deployuri –-host host\_address --target instancei1 ftp:/e:/test.war |

|  |
| --- |
| $ asadmin deployuri –-host host\_address --target instancei1　ftp://username:password@host:21/test.war |

|  |
| --- |
| $ asadmin deployuri　--host host\_address --target instancei1 http://java.net/jira/secure/attachment/50467/test\_sample1.war |

|  |
| --- |
| $ asadmin deployuri　–-host host\_address --target cluster1  http://java.net/jira/secure/attachment/50467/test\_sample1.war |

**4.1.5. Not support use case examples**

**4.1.5.1. Context of examples**

|  |
| --- |
| $ asadmin deployuri  http://dldx.csdn.net/fd.php?i=271450501666412&s=35428f4bec874a3f4124a14beb2ec986 |

As the suffix String about the above http address is not end with .jar, .war, .ear or .rar, I think some of the error messages should be thrown out to the user when it comes to this situation.

**4.2. Bug/RFE Number(s):**

<http://java.net/jira/browse/GLASSFISH-12699>

**4.3. In** **Scope:**

The deployuri command is able deploy most of the application which path is specified as file:// syntax, http protocol and ftp protocol.

**4.4. Out of** **Scope:**

**4.4.1.. http protocol**

1). When it comes to the http protocol, there exist many situations as follows:

1. I think it is unreasonable to support two options like username and password, we should offer some friendly messages to the user under this situation. When we got the URI address base on the http protocol but we can’t download the application from the outside website because the website needs some user certification, it is hard and illegal to pass the username and password from the GF admin client or server to the other website’s database in a common situation.
2. When the suffix about the offered URI is not end with .jar, .war or .ear. I think we should offer some error messages to this situation because there’s no JDK to analysis these file types according to http protocol.(As the HttpURLConnection can judge other content type except .jar, .war, .ear or .rar).
3. We don’t support the application deployed as a directory module when it comes to this situation.

**4.4.2. Ftp protocol**

When it comes to the ftp protocol, there exist situations as follows:

1. It is no need to define two options as username and password, it is because the standard about ftp will be like <ftp://username:password@host:port/filename>, we can divided it into pieces to make sure we can connect to the ftp server and download the application to our local temp directory.
2. We don’t support the application deployed as a directory module when it comes to this situation.

**4.5. Interfaces:**

**4.5.1 Public Interfaces**

The deployuri will support the options as follows:

* --name:　Name of the deployable component.
* --contextroot: Valid only if the archive is a web module. defaults to filename without extension.
* --force: If set to true, redeploys the component even if the specified component has already been deployed or already exists.
* --precompilejsp: Whether the JSPs are compiled during runtime.
* --verify: If set to true and the required verifier packages are installed from the Update Tool, the syntax and semantics of the deployment descriptor is verified.
* --retrieve: Retrieves the client stub JAR file from the server machine to the local directory.
* --type: The packaging archive type of the component that is being deployed.
* --properties: Optional keyword-value pairs that specify additional properties for the deployment.
* --target: Specifies the target to which you are deploying.
* --libraries: A comma-separated list of library JAR files.
* --lbenabled: This option controls whether the deployed application is available for load balancing. The default is true.
* --enabled: Allows users to access the application.
* --dbvendorname: Specifies the name of the database vendor for which tables are created.
* --keepstate:　This option controls whether web sessions, SFSB instances, and persistently created EJB timers are retained between redeployments
* --createtables:If specified as true, creates tables at deployment of an application with unmapped CMP beans.
* --uniquetablenames: Guarantees unique table names for all the beans and results in a hash code added to the table names.
* --deploymentplan: Deploys the deployment plan, which is a JAR file that contains GlassFish Server descriptors.
* --generatermistubs: If set to true, static RMI-IIOP stubs are generated and put into the client.jar. If set to false, the stubs are not generated
* --availabilityenabled: This option controls whether high-availability is enabled for web sessions and for stateful session bean (SFSB) checkpointing and potentially passivation.
* --asyncreplication: This option controls whether web session and SFSB states for which high availability is enabled are first buffered and then replicated using a separate asynchronous thread.

I think all of the options about deploy command would be support when it comes to the deployuri command except the --upload because the deployuri command reference the logical of the deploy command.

**4.5.2 Private Interfaces**

None.

**4.5.3 Deprecated/Removed Interfaces:**

None.

**4.6. Doc Impact:**

Update man pages for the new/modified CLI commands

Application Deployment Guide/Developer Guide for new features

**4.7. Admin/Config Impact:**

CLI:

* Support a new command to accept the type of the path as a URI.

Admin console:

* Define a new text area in the deployment page where user can enter the URI address. After the user is finish writing the URI address, we can press the “OK” button and the application can be deployed as URI.

**4.8. HA Impact:**

No impact.

**4.9. I18N/L10N Impact:**

No impact.

**4.10. Packaging, Delivery & Upgrade:**

**4.10.1. Packaging**

The code could be packaged/delivered in the existing admin module and deployment module. (Presumably in the same module with the deploy command)

**4.10.2.** **Delivery**

No impact.

**4.10.3. Upgrade and Migration:**

No impact.

**4.11. Security Impact:**

No impact.

**4.12. Compatibility Impact**

No impact.

**4.13. Dependencies:**

**4.13.1 Internal Dependencies**

None.

**4.13.2 External Dependencies**

If we support the application to be deployed with ftp:// syntax, I think we should inject external jar file called org.apache.commons.net.jar

**4.14. Testing Impact:**

Add new tests (automated) to test production deployuri feature.

**5. Reference Documents:**

**6. Schedule:**

**6.1. Projected Availability:**

* To Be Determined

**6.2. Future schedule**

* Maybe there needs to be support a new command called redeployuri in the feature