Liberty Cloud Provisioning

Setup and Configuration

This guide provides details of the capabilities and configuration options for the sample WebSphere Liberty workflows. It assumes that the reader has a basic understanding of z/OSMF's Cloud Provisioning and Management for which the Liberty workflow samples were designed. The creation and management of templates, resource pools, domains and tenants is not discussed in this document, but more information can be found here: http://www.redbooks.ibm.com/redpapers/pdfs/redp5416.pdf.

Included Content

The sample Liberty workflows contain three main directories: **tools**, **properties** and **workflows**. The tools directory simply contains a helper workflow used to create a registry for datasource information. The properties directory houses the workflow_variables.properties file. This is the only file that should need to be updated. From here, you can define all of the variables used in the workflows as well as set flags to enable various features. The workflows directory contains all of the workflow XML files as well as three sub directories. The **templates** directory contains several REXX and JCL scripts and a shell script that are executed at various steps in the workflows. The **config** directory contains Liberty server config template files which are copied and populated when a server is provisioned (These do not need to be edited). Lastly, the **apps** directory contains sample applications that can be deployed when a server instance is created.

Provisioning Steps (provision.xml)

When a Liberty server is provisioned, the steps within provision.xml are executed. This table outlines the basic steps that are performed.

Step	Description
1	Determine the id of the invoking user
2	Determine the user directory (user's home or mounted file system)
3	Allocate network resources (IPs and or ports)
4	Create the user directory in the user's home dir or as a new mounted file system)

5	Create the Liberty server
6	Run basic configuration setup
7	Configure SSL (SAF or file-based depending on request)
8	Configure the server to bind to a DB2 subsystem (if requested)
9	Configure the server to start as a z/OS started task (if requested)
10	Configure SAF security profiles and appropriate access
11	Start the newly provisioned server instance

Actions (actions.xml)

A series of actions can be performed on a provisioned Liberty instance. These actions are listed in the actions.xml which defines which workflows to execute for each action. This table outlines the actions that are available in this sample.

Actions	Description	Workflow Definition File
start	Start the server	startServer.xml
stop	Stop the server	stopServer.xml
db2Bind	Bind to a specified DB2 subsystem	db2Bind.xml
db2Unbind	Unbind from a DB2 subsystem	db2Unbind.xml
deprovision	Deprovision the server instance	deprovision.xml
checkStatus	Check the status of the server instance and update the INSTANCE_STATUS variable with either active or inactive.	checkStatus.xml

Properties (workflow_variables.properties)

This file is used to define all of the variables used in the workflows. You can also set flags to enable various features.

Property	Description
Basic Configuration Properties	
WLP_INSTALL_DIR	The path to the existing Liberty installation
JAVA_HOME	The Java home directory
TEMP_DIR	The absolute path to a directory to store temporary files created and removed at runtime
APPS_DIR	The absolute path to a directory containing applications to install in a provisioned server
Network Resource Properties	
WLP_ENABLE_DVIPA	Flag indicating whether or not to use z/OSMF's DVIPA support
IP_ADDRESS	The existing IP address to be used for all provisioned servers
SAF Domain Access Properties	
UNAUTH_USER	The unauthenticated user id to use when setting up SAF security for the provisioned instances.
SSL Authentication Properties	
WLP_ENABLE_SSL_SAF_CERTIFICATES	Flag indicating whether to create and use SAF certificates or java file based certificates for SSL authentication

SIGN_WITH	An existing certificate authority used to sign newly created SAF certificates
CERTIFICATE_EXPIRATION_DATE	The expiration date for newly created SAF certificates
File System Creation Properties	
MOUNT_POINT	The directory where server instances will be provisioned. If blank, the server will be created in the requesting user's home directory
WLP_CREATE_ZFS	Flag indicating whether or not to create and mount a new file system for each provisioned instance.
VOLUME	The volume on which to allocate new file systems
PRIMARY_CYLINDERS	The number of primary cylinders to use for each new file system
SECONDARY_CYLINDERS	The number of secondary cylinders to use for each new file system
FILE_SYSTEM_HLQ	The HLQ to use when creating file system datasets
Start and Stop Properties	
START_INSTANCE	Flag indicating whether or not to start the server automatically after provisioning.
START_SERVER_AS_STARTED_TASK	Flag indicating whether the server will be configured to start as a z/OS started task or as a USS process
PROCLIB	The PROCLIB dataset where stc procedures will be added when starting servers as started tasks

CONSOLE_NAME	The name of the console to use for issuing start and stop commands for started tasks.
DB2 Connection Type Properties	
DB2_CONNECTION_TYPE	The DB2 connection type when DB2 binding is requested. Valid values are 2 or 4
WLM Resource Properties	
DEFINE_WLM_RULE	Flag to indicate whether or not to create classification rule in WLM service definition
Approval and Run-As IDs	
CONSOLE_RUN_AS_ID	The id to run console commands
CONSOLE_APPROVAL_ID	The id to approve workflow steps that run console commands (start and stop)
SECURITY_RUN_AS_ID	The id to run security tasks
SECURITY_APPROVAL_ID	The id to approve workflow steps that run security tasks
MOUNT_RUN_AS_ID	The id to run file system creations and mounts
MOUNT_APPROVAL_ID	The id to approve workflow steps for creating and mounting file systems
PROCLIB_RUN_AS_ID	The id to add and remove members from the proclib
PROCLIB_APPROVAL_ID	The id to approve workflow steps to add and remove proclib members

Options and Setup

Approvals and Step Execution

Typically, workflow steps are executed under the ID of the user requesting the provisioned instance. Additionally, z/OSMF provides a mechanism for workflow steps to be executed under a specified user id. This is particularly useful when a step requires elevated security access (i.e. creating SAF profiles).

There are a series of steps throughout the Liberty workflows that leverage this capability. Each of these steps will have both an approver and a “run as” user. The approver will need to approve their respective steps before the software services template can be run. Once approved, the workflow will then execute that step under the “run as” user id.

These user ids are set in the properties file and are either appended with “_APPROVAL_ID” or “_RUN_AS_ID.”

Your WLP installation

The minimum supported level of Liberty that can be used for cloud provisioning is 8.5.5.8. This is due to a fix in Liberty that allows dynamic virtual IPs (DVIPAs) to be used for each provisioned instance. You can download supported quarterly Liberty drivers as well as monthly betas from wasdev.net. You can then point the WLP_INSTALL_DIR property to the “wlp” directory of your new Liberty installation.

Liberty Server Config Files

All Liberty server configuration file templates are packaged with the workflow definition files. At runtime, these config files are automatically copied to the newly created server’s config directory and a bootstrap file is created with variable values specific to the provisioned server instance.

Server Naming

z/OSMF provides a unique instance name (max 8 characters) based on a pooling prefix specified at software services template creation time. This unique name is used to name the Liberty server on the file system as well as in other internal locations where unique naming is required.

Pooled Network Resources (DVIPAs and Ports)

There are various configuration options in regards to network resource allocations. z/OSMF’s Configuration Assistant (CA) offers the ability to create pools of ports as well as dynamic virtual IP

addresses (DVIPAs). Leveraging this, the Liberty workflows are able to dynamically allocate unique IPs for each instance or unique ports on a common IP address. Described below are the two main options offered. In either case, network resource pools must be created for each Software Service template.

1. WLP_ENABLE_DVIPA : TRUE

   With this option, unique IP addresses will be allocated for each Liberty instance. This requires a pool of IP addresses to be configured in the CA. Additionally, a pool must be created with the following ports: 9080, 9443. These ports are used for the HTTP and HTTPS ports and are allocated as “shared.” This shared allocation is a requirement of the CA to prevent other workflows from allocating those ports as “explicit.”

2. WLP_ENABLE_DVIPA : FALSE

   With this option, all Liberty server instances will share the same IP address set via the IP_ADDRESS property. This requires a pool of ports to be configured in the CA from which the HTTP and HTTPS ports will be explicitly allocated.

Start and Stop options

Liberty servers can start either as USS processes or as z/OS started tasks. These workflows allow either option via the START_SERVER_AS_STARTED_TASK property. This flag can either be set to TRUE or FALSE (case sensitive). If TRUE, a procedure will be created using a unique z/OSMF provided name, added to your PROCLIB, and used to start the server. It will then be removed when the server is deprovisioned. The CONSOLE_NAME variable can be used to specify the console name to use when issuing server starts and stops as started tasks. If set to ‘defcn’ z/OSMF will use “<CONSOLE_RUN_AS_ID>CN” as the console name.

Starting as a started task requires two separate approvals since the associated steps must be run as an elevated user. First, PROCLIB_RUN_AS_ID must be set in the properties file with a user id with authority to add and remove members from your proclib. Second, CONSOLE_RUN_AS_ID must be set with a user id with authority to execute console commands. Likewise, the PROCLIB_APPROVAL_ID and CONSOLE_APPROVAL_ID properties will need to be set with the IDs of user who will initially approve each respective step.

SSL Authentication

By default, on initial startup, a Liberty server will create Java, file based certificates for SSL authentication. For better security, you can set the WLP_ENABLE_SSL_SAF_CERTIFICATES

property to “TRUE” in the properties file to have the workflow create a unique SAF based certificate and keyring for each server instance. Optionally, you can set the SIGN_WITH property to the name of an existing Certificate Authority that will be used to sign each dynamically created certificate. You can also set the CERTIFICATE_EXPIRATION_DATE property to the date (yyyy-mm-dd) at which each certificate will expire. Any dynamically created certificate is deleted when the server is deprovisioned.

SAF certificate creation requires elevated authority. If enabled, the SECURITY_RUN_AS_ID will need to be set with the user id under which these steps will be run. Likewise, the SECURITY_APPROVAL_ID must be set with the id of the user to approve these authorized steps.

Server Instance Location

The directory in which server instances will be provisioned can be specified using the MOUNT_POINT variable in the properties file. This directory must be given 755 permissions. When set, a new directory named “wlp-<requesting_user>-<instance_name>” will be created for each server instance under the MOUNT_POINT directory. If not set, server instances will be created in the invoking user’s home directory.

File System Creation

The Liberty workflows provide an option to create and mount a new file system for each provisioned server instance. This can be accomplished by setting the WLP_CREATE_ZFS property to TRUE (case sensitive). A new dataset will be created based on the z/OSMF provided instance name (see the “Server Naming” section above), and a high-level qualifier set by the FILE_SYSTEM_HLQ variable in the properties file. The file system will be mounted in the newly created server instance directory (see section above). Additionally, the VOLUME, PRIMARY_CYLINDERS, and SECONDARY_CYLINDERS must be set.

This feature also requires elevated authority in order to create and mount file systems. The MOUNT_RUN_AS_ID and MOUNT_APPROVAL_ID properties are used to specify the appropriate id to run the steps and approve the steps respectively.

Group Access

When file system creation is enabled, by default, the file system and Liberty instance will be owned solely by the user who requested the instance. Optionally at runtime, an invoking user can specify an existing SAF group name that as a whole will be given access to the file system and server. This allows for additional administration and collaborative sharing as needed.

To use this feature, an invoking user simply needs to provide a SAF group name when prompted at runtime. The newly created environment and server will be created around this security group.

DB2 Binding

The Liberty workflows provide the option to bind a provisioned Liberty instance to an existing DB2 subsystem using a Type 2 or a Type 4 JDBC connection. The connection type is specified in the DB2_CONNECTION_TYPE property in the properties file. To use this functionality, a z/OSMF registry must first be created with the appropriate information for a particular DB2 subsystem. You can run the db2_registry_add_subsystem.xml workflow that will automatically prompt for all the information needed (driver location, subsystem name, etc.) and create a registry with that data using the subsystem as the registry name. An admin just needs to run this workflow once to create the registry, and any Liberty workflow will then be able to access it and configure a data source with that information. For a type 4 connection, the registry must contain the username and password of the DB2 subsystem. The password can be entered into the registry in plain text, but for security purposes, it is recommended to first 64bit encode the password and enter the encoded string in the password field.

From a Liberty perspective, an invoking user just needs to provide the DB2 subsystem name and the desired JNDI name for the datasource in the prompts at runtime or when the db2Bind action is invoked on a previously provisioned instance.

WLM Resource Pool

The Liberty workflows can be set to define rules for use with z/OSMF WLM Resource Pools. This can be done by setting the DEFINE_WLM_RULE variable to TRUE in the properties file. In order to use this functionality, WLM resource pools must be properly defined prior to server provisioning. If the Liberty server is configured to start as a started task, the WLP resource pool must be setup with the STC subsystem. If the server is configured to start as a USS process at command line, the WLP resource pool must be setup with the OMVS subsystem.

SAF Security

The Liberty workflows automate the creation of Liberty specific SAF security profiles as well as the permission of requesting users and groups to those profiles. This allows the server to start as a started task when requested as well as access the Liberty Angel process for SAFCRED authentication and other security related tasks.

Additionally, a security domain and EJBROLE profile are created for each server instance based on the unique z/OSMF provided instance name. This isolates security enabled application access to just the user or group associated with the Liberty instance. As part of this automated setup, the SAF domain must have an associated “unauthorized user” associated with it. The z/OSMF server already has an unauthorized user id (IZUGUEST) used for its SAF domain, so by default, each provisioned WLP instance will use this id as well. There is no need change this user id; however, if desired, you can change it using the UNAUTH_USER variable in the properties file. Any id specified for that variable will need to be defined correctly by the instructions found here.

When deprovisioned, the security domain and EJBROLE profiles are deleted since these are specific to the server's unique instance name. However, read access to the Liberty SAF profiles remain in case the user or group is associated with other provisioned Liberty server instances.

The commands that create and administer these profiles are contained within the files/wlp-authorized-services-setup.rexx and files/wlp-authorized-services-teardown.rexx scripts.

The SECURITY_RUN_AS_ID and SECURITY_APPROVAL_ID as outlined above in the SSL Authentication section are both used to run and approve this security setup and teardown.

Provisioning for Other Users

The Liberty workflows are run under the id of the user logged in to the z/OSMF console. Any server provisioned during that session will be owned by that user. However, in some scenarios, it may be valuable to have one user id interact with the z/OSMF console and provision servers on behalf of other users. For example, an application may submit provision requests to z/OSMF via REST calls under one user id. In those cases, an internal WLP_USER variable can be set to another user id and passed on the REST call. The server will then be provisioned for that user. Typically the WLP_USER variable is resolved at runtime to the id of the user logged in, but if overwritten, this resolution step is skipped.

Server Status

The checkStatus action is available to verify if a server instance is active or inactive. While there is no visual representation of this status on the z/OSMF web interface, the INSTANCE_STATUS variable is updated with either "active" or "inactive." This variable can be read by extended REST applications such as the z/OS Provisioning Toolkit (ZOSPT) or other portals extensions.

Installing Applications

If desired, the Liberty workflows can install applications into a newly provisioned server. Simply place any applications in a directory of your choice and specify the absolute path to this directory in the APPS_DIR variable in the workflow properties file. When this variable is set, the workflow will copy any files in that directory to the dropin directory in a newly provisioned server. When the server starts, these applications in dropin will be started as well.

Sample Applications

Three sample applications are packages with the workflow files. These applications can be used to test various aspects of your server. To use these applications, please see the “Installing Applications” section above.

1. <your_hostname>:<your_port>/CloudTestServlet

   This is a basic servlet application to test and validate that the server is up and running.

2. <your_hostname>:<your_port>/CloudTestSecureServlet

   This application will test your server’s security. A pop up window should appear asking for SAF credentials. Only the requesting id or group (if group access was requested) should have access to this application due to the EJBROLE and SAF domain setup by the workflow.

3. <your_hostname>:<your_port>/CloudTestJDBC

   This application will test your server’s connection to a DB2 subsystem. The application will run through a series of steps to create, insert, remove, and drop a database table. Note that his app will only work with the default JNDI name of jdbc/db2.

Prerequisites

- A Liberty driver must be installed.
- Java must be installed.
- A Liberty Angel process must be started on the target system(s) where Liberty servers will be provisioned. This is needed for SAF authentication and other requested z/OS specific features.
- All RACF specific JCL and REXX scripts must be altered to the spec of other third party security vendors if RACF is not being used.