wastaas – WAS Traditional as a Service

This is the readme file for WASt as a Service (wastaas). WASt as a Service is available from the z/OSMF Software Configuration catalog. The service allows you to rapidly provision an IBM WebSphere Application Server traditional for z/OS. It is recommended that you first read the readme file for z/OSMF's Catalog UI that provides information on how to create catalog entries, provision instances, and perform actions on those instances.

After provisioning, details about the server, including IP address and ports, can be found by clicking on the newly provisioned instance in the Catalog UI and browsing the variables specific to that server instance. Users can also perform a series of actions on that server instance directly from the Catalog UI. Current available actions include start, stop, and deprovision.

The remainder of this readme file provides details of the Classic WebSphere application server z/OSMF workflows and of their associated files, and lists customization steps, prerequisites, and current limitations.

Recent Changes

PoC Driver 5 – Notes

- The templates used by the createRACF workflow have changed since driver 4. Be sure to recopy BBOSBRAC and BBOWBRA2 to the dataset identified by the RACF_TEMPLATE_DSN property in server_group_variables.properties before running the createRACF workflow with driver 5.

PoC Driver 7 – Notes

- If the createDB2Type2 or createDB2Type4 actions will be used, a z/OSMF registry entry containing information about the DB2 subsystem must exist. The registry entry contains the DB2 subsystem name, location name, host name and port number (required for type 4), user id and password (required for type 4), JDBC driver path, and the HLQ of the DB2 load libraries. **Note that the user id and unencrypted password for accessing DB2 are included in the registry, and that they are also specified in a temporary file used in the creation of the type 4 data source. This is a known issue.**
- The new DB2_REGISTRY_NAME variable identifies the registry entry for DB2.
- Before a type 2 data source created with the createDB2Type2 action can be used, depending on which DB2 statements are in the LNKLST concatenation, a STEPLIB DD statement may need to be added to the servant region proc pointing to the SDSNEXIT and possibly to the SDSNLOAD and SDSNLOD2 datasets. The intent is to automate this in a future drop.

- If the app server is up when the createDB2Type4 action is performed, the app server will need to be restarted before the new data source can be used.
- The following files have been added or modified to provide actions for creating data sources:
    - actions.xml
    - createDB2Type2.template
    - createDB2Type2.xml
    - createDB2Type4.template
    - createDB2Type4.xml
    - server_group_variables.properties
- The following files have been added or modified for other reasons:
    - provision.mf
    - provision.xml
    - deprovision.xml

PoC Driver 8 – Notes

- The following files have been modified to use the new resource pool api calls for reserving and releasing IP addresses and ports.
    - provision.xml
    - deprovision.xml

createRACF.xml

There are many RACF profiles, users, and groups that must be in place for Classic WAS application servers to function. The createRACF.xml workflow invokes the createRACF.sh script to build execs containing the required RACF commands to create these profiles, users, and groups based on provided templates. Before the provisioning environment can be used, the customer security administrator will need to either execute the execs created by this workflow, or ensure that the equivalent security profiles are in place. This workflow is intended to be run once during the initial setup of the provisioning environment.

The following table lists the workflow steps

Step	Description
1	Invoke createRACF.sh to create the RACF execs

provision.xml

This is the workflow that provisions an application server. It consists of a series of steps that creates, customizes and starts a base application server environment allowing a user to quickly and easily use this server for development and application deployment. The following table lists the steps which create a response file for the profile management tool, execute the tool, and then run jobs created by the tool.

Step(s)	Description
1	Prompt for SERVER_ID_NUM
2-20	Reserve port numbers to be used for new App Server
21	Get a dynamic IP address using z/OSMF's Configuration Assistant
22	Remove files in preparation for running the WAS for z/OS Profile Management Tool
23	Delete datasets in preparation for running the WAS for z/OS Profile Management Tool
24	Create response file in preparation for running the WAS for z/OS Profile Management Tool
25	Run the WAS for z/OS Profile Management Tool to create the setup jobs for the application server
26	Remove job cards from the setup jobs so that they can be included in following steps
27	Submit job BBOWCFS to create the zfs to contain the app server configuration files
28	Save information that will be used later when the app server is deprovisioned
29	Save IP addresses and port numbers
30	Submit job BBOSBRAM to create home directories for the user ids associated with the app server
31	Submit job BBOWHFSA to create and populate the app server configuration directories
32	Submit job BBOWPPFA to create the app server profile
33	Submit job BBOWPROC to create JCL used to run the app server
34	Reset the password for the administrative user id to a default value
35	Turn off Java shared class cache for the servant region
36	Start the app server

deprovision.xml

This is the workflow that deprovisions an application server. The steps include stopping the server, returning the dynamically allocated IP address and ports to the z/OSMF Configuration Assistant, and deleting the provisioned server directory and JCL procedures.

The following table lists the deprovisioning steps

Step	Description
1	Prompt for SERVER_ID_NUM - When this workflow is invoked

	from the z/OSMF Software Services Instance “perform” menu, the SERVER_ID_NUM will be provided automatically.
2	Reset the password for the administrative user id to a default value
3	Stop the app server
4	Retrieve the IP address and port IDs (This is used in the following step)
5	Return the dynamically allocated IP address
6 - 24	Free reserved ports
25	Clear servant region class cache
26	Unmount the provisioned server configuration file system
27	Delete the provisioned server configuration file system
28	Delete procs

createDB2Type2.xml

The createDB2Type2 workflow creates a DB2 JDBC provider and type 2 data source on a previously provisioned app server. The WASt server must have been provisioned prior to running this workflow.

Before a type 2 data source created with the createDB2Type2 action can be used, depending on which DB2 statements are in the LNKLST concatenation, a STEPLIB DD statement may need to be added to the servant region proc pointing to the SDSNEXIT and possibly to the SDSNLOAD and SDSNLOD2 datasets. The intent is to automate this in a future drop.

The following table lists the steps in this workflow.

Step	Description
1	Prompt for SERVER_ID_NUM and DB2_REGISTRY_NAME - When this workflow is invoked from the z/OSMF Software Services Instance “perform” menu, the SERVER_ID_NUM will be provided automatically.
2	Locate the DB2 registry entry
3	Obtain variable values from DB2 registry entry
4	Build wsadmin commands to create the JDBC provider and data source
5	Execute the wsadmin commands to create the JDBC provider and data source
6	Clean up temporary files

createDB2Type4.xml

The createDB2Type4 workflow creates a DB2 JDBC provider and type 4 data source on a previously provisioned app server. The WASt server must have been provisioned prior to running this workflow. If

the server is running when createDB2Type4 is run, it will need to be restarted before the datasource can be used.

Note that the user id and unencrypted password for accessing DB2 are included in the registry, and that they are also specified in a temporary file used in the creation of the type 4 data source. This is a known issue.

The following table lists the steps in this workflow.

Step	Description
1	Prompt for SERVER_ID_NUM and DB2_REGISTRY_NAME – When this workflow is invoked from the z/OSMF Software Services Instance "perform" menu, the SERVER_ID_NUM will be provided automatically.
2	Locate the DB2 registry entry
3	Obtain variable values from DB2 registry entry
4	Build wsadmin commands to create the JDBC provider and data source
5	Execute the wsadmin commands to create the JDBC provider and data source
6	Clean up temporary files

server_group_variables.properties

The server_group_variables.properties file contains the following configurable properties, which may need to be changed depending on customer requirements:

Property	Remarks
SERVER_GROUP_PREFIX	Two character prefix that is used to form cell names, proc names, user id names, and other names related to the provisioned app servers. The first character must be alpha, and the second character can be alpha or numeric.
CERT_AUTH_LABEL	The createRACF script will create RACF commands for defining a certificate authority. This parameter will be used as the label for the CA certificate. Note that the createRACF script does not actually execute the command for creating the certificate authority. It only creates the command. If customers prefer to use an existing CA, they can modify the related RACF commands before running them.
CONFIG_ROOT	Fully qualified r/w directory under which a mount point will be created for each provisioned app server
DEFAULT_ADMIN_PW	Each app server will have an associated admin user id. When the server is provisioned, the password for the admin

	id will be set to this value. The user will be forced to change the password on the first login to USS.
DOMAIN_NAME	IP domain name to be used for the app servers
EXEC_DATASET	Dataset containing the RMJOBCARD exec used to remove job cards from the zpmt.sh created JCL
JOB_STATEMENT_1 – JOB_STATEMENT_4	Reserved for future use
MAX_SERVERS	The maximum number of servers to be provisioned in this environment. This parameter is only used in the createRACF workflow.
PROCLIB	Name of a z/OS PROCLIB to contain the app server start up procs. See the provisioning environment setup section for important information about this parameter. **WARNING! IF YOU USE AN EXISTING PROCLIB DATASET, AND IF MEMBERS ALREADY EXIST WITH THE SAME NAMES AS THE MEMBERS BEING CREATED BY THIS AUTOMATION, THE EXISTING MEMBERS WILL BE OVERWRITTEN!**
RACF_COMMANDS_DSN	Partitioned dataset to be created by the createRACF workflow. This dataset will contain execs to issue the RACF commands required to set up the provisioning environment.
RACF_GROUP	Group that will own the RACF user ids associated with the app server
RACF_TEMPLATE_DSN	Name of partitioned dataset containing the templates used as input to the createRACF workflow
RACF_USER	RACF user id to be used to run the provision and deprovision workflows
STARTING_GID	Several USS groups are required for each app server. The provisioning process will automatically assign GIDs from a pool specified by the customer. This parameter specifies the first GID in the pool. The number of GIDs in the pool will be 10 times the maximum number of servers that can be provisioned in this environment (MAX_SERVERS).
STARTING_UID	Several USS users are required for each app server. The provisioning process will automatically assign UIDs from a pool specified by the customer. This parameter specifies the first UID in the pool. The number of UIDs in the pool will be 10 times the maximum number of servers that can be provisioned in this environment (MAX_SERVERS).
SYSPLEX_NAME	Name of the sysplex where the app servers will be provisioned
SYSTEM_NAME	Name of the z/OS system where the app servers will run
USER_HLQ	High level qualifier of the app server zfs dataset name. The

	remainder of the name will be formed from the SERVER_GROUP_PREFIX and the SERVER_ID_NUM followed by '.ZFS'
WAS_PATH	Directory containing the WAS product code. For example, /usr/lpp/zWebSphere/V8R5
WORK_PATH	Directory to contain zpmt.sh work files. This can usually be set to /tmp.
Z_CONFIG_AUTO_GID	Flag indicating whether RACF AUTOGID parameter is used to assign GIDs
Z_CONFIG_AUTO_UID	Flag indicating whether RACF AUTOUID parameter is used to assign UIDs
Z_CONFIG_HFS_VOLUME	Disk volume where the app server config ZFS will be allocated
Z_CONFIG_HFS_PRIMARY	Primary number of cylinders to be used when allocating the config ZFS
Z_CONFIG_HFS_SECONDARY	Secondary number of cylinders to be used when allocating the config ZFS
Z_CONFIG_SPECIFY_SMS	Acceptable values for this parameter are 'true' or 'false' (without quotes). This flag is used to determine whether to include the SMS related parameters in the zpmt.sh response file.
Z_CONFIG_DATACLAS	SMS data class to be associated with the config ZFS
Z_CONFIG_MGMTCLAS	SMS management class to be associated with the config ZFS
Z_CONFIG_STORCLAS	SMS storage class to be associated with the config ZFS
ZPMT_DATASET_HLQ	High level qualifier to be used when creating the dataset containing the config jobs produced by zpmt.sh.

The server_group_variables.properties file contains the following optional properties related to ports used by the app servers. See item number 4 in the Prerequisites section below for more details.

Property

WC_adminhost

WC_adminhost_secure

OVERLAY_TCP_LISTENER_ADDRESS

DAEMON_PORT

DAEMON_SSL_PORT

DCS_UNICAST_ADDRESS

WC_defaulthost

WC_defaulthost_secure

BOOTSTRAP_ADDRESS

ORB_SSL_LISTENER_ADDRESS

SIB_MQ_ENDPOINT_ADDRESS

SIB_ENDPOINT_ADDRESS

SIB_MQ_ENDPOINT_SECURE_ADDRESS

SIB_ENDPOINT_SECURE_ADDRESS

SIP_DEFAULTHOST

SIP_DEFAULTHOST_SECURE

The following additional optional properties may be included in server_group_variables.properties:

Property	Remarks
DB2_REGISTRY_NAME	If the createDB2Type2 or createDB2Type4 actions will be used, a z/OSMF registry entry containing information about the DB2 subsystem must exist. This variable identifies the registry entry for DB2.

TSO execs

The provision workflow invokes a TSO Rexx exec to remove job cards from the zpmt generated jobs so that they can be executed in subsequent workflow steps via JCL INCLUDE statements. This exec and an exec to create a user id to be associated with the workflows should reside in a dataset accessible by the workflow.

Exec	Description
RMJOBCRD	Invoked by the provision workflow to remove job statements from zpmt generated jobs.
ZOSMFRAC	Optionally executed one time during initial set up of the provisioning environment to create a user id that can be used to run the workflow.

TSO exec templates

The following templates reside in a z/OS dataset as input to the createRACF workflow.

Template	Description
BBOSBRAC	Used by createRACF to build one exec per app server. These resulting execs create most of the users and groups needed by the app server.
BBOWBRA1	Used by createRACF to build a single exec containing commands to activate classes and define common profiles required by all appservers.

BBOWBRA2	Used by createRACF to build one exec per app server. These resulting execs create profiles specific to the app server, create the unauthenticated user id, and permit the app server ids to the required profiles.

Other files

The following files reside in the same directory as the workflow xml files

Filename	Description
createRACF.sh	Shell script called by the createRACF workflow to create RACF commands by merging values from the server_group_variables.properties file with templates.
zpmt_response.template	Template used by provision.xml to create the response file used as input to zpmt.sh

Customization

All customization can be done from the above mentioned server_group_variables.properties file. The createRACF.xml, provision.xml, and deprovision.xml workflow definitions should not need to be modified. See the Provisioning Environment Setup section below for more information.

Prerequisites

1. WebSphere Application Server for z/OS (8.5.5.7 or above) must be installed using IM.
2. The SCEERUN, SCEERUN2, SIEALNKE, and SCLBDLL2 datasets must be in the LNKLST.
3. z/OSMF Configuration Assistant must be configured with a TCP/IP stack and a range of IP addresses. The provisioning workflow requests a virtual IP address for each app server.
4. Port ranges for the various WebSphere Application Server ports must be defined within z/OSMF Configuration Assistant.

   Background
   WebSphere Application Server uses TCPIP (and some UDP) extensively for communications. Nineteen different port numbers must be reserved for each application server. The provisioning process uses z/OSMF resource pool API calls to dynamically reserve ports. Three of the ports used by WAS must have different numbers for each application server:

   IPC_CONNECTOR_ADDRESS
   OVERLAY_UDP_LISTENER_ADDRESS
   SOAP_CONNECTOR_ADDRESS

   A CA general port range must be defined containing three port numbers for each server to be

provisioned.

Since each provisioned server has its own virtual IP address, the remaining sixteen WAS ports use numbers that are shared between the provisioned app servers. For example, the admin console for the first provisioned server may listen on 10.0.0.1:9080, the admin console for the second provisioned app server would listen on 10.0.0.2:9080, the third on 10.0.0.3:9080, … A specific port range must be defines in CA to support this function.

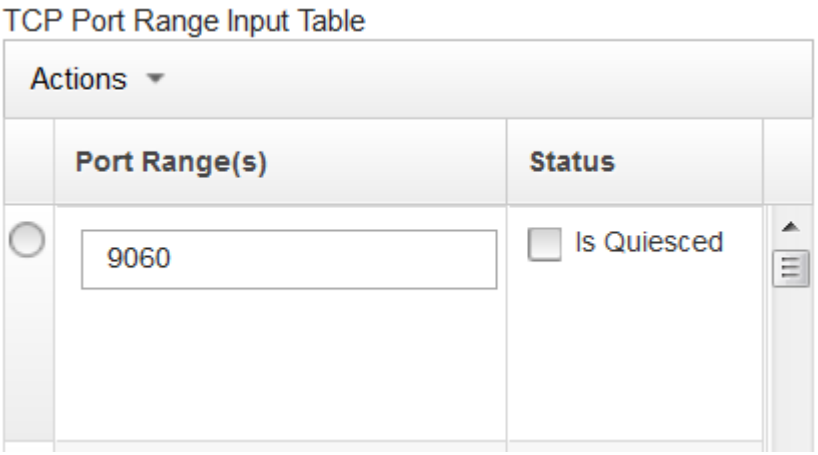
The procedures below describe how to create both the general port range to be used for IPC_CONNECTOR_ADDRESS, OVERLAY_UDP_LISTENER_ADDRESS, and SOAP_CONNECTOR_ADDRESS, and also the specific port range to be used for the other sixteen ports.

Setting up a general port range within CA
The following steps document the process of setting up a general port range within CA. For the most up to date information on using the Configuration Assistant for creating pools of ports, see the "Create a Port Allocation Range" section within the "Cloud: Getting Started" tutorial in the z/OSMF online help. You can find a link to the tutorials on the Configuration Assistant main page within z/OSMF.

- Navigate to the Configuration Assistant "Port Allocation Ranges" tab.
- Select "Actions" and then "New" to create a new port allocation range.
- Specify a name for the port range, and ensure that "Usage Type" is blank.
- Set "Selection Policy" to General.
- Ensure that "Status" is NOT set to Quiesced.
- Specify a range of port numbers as the "Port Range" in the "TCP Port Range Table". The range must include at least 2 TCP port numbers for each app server to be deployed. Ensure that the status is NOT set to "quiesced".
- Specify a range of port numbers as the "Port Range" in the "UDP Port Range Table". The range must include at least 1 number for each app server to be deployed. Ensure that the status is NOT set to "quiesced" and click "Save".

Setting up a specific port range within CA
The following steps document the process of setting up a specific port range within CA. For the most up to date information on using the Configuration Assistant for creating pools of ports, see the "Create a Port Allocation Range" section within the "Cloud: Getting Started" tutorial in the z/OSMF online help. You can find a link to the tutorials on the Configuration Assistant main page within z/OSMF.

- Navigate to the Configuration Assistant "Port Allocation Ranges" tab.
- Select "Actions" and then "New" to create a new port allocation range.
- Specify a name for the port range, and ensure that "Usage Type" is blank.
- Set "Selection Policy" to Specific.
- Ensure that "Status" is NOT set to Quiesced.
- Create rows in the TCP Port Range Input Table for each of the following ports:
  9060,9043,11004,5653,5654,9353,9080,9443,2809,11005,5558,7276,5578,7286,5060,5061
  The entries should look like this. Be sure that "Is Quiesced" is NOT checked for each entry:



- Click save.

Customizing port numbers

By default, the 16 shared WAS Classic application server ports use numbers documented for "Standalone Application Servers" in the Knowledge Center (http://www-01.ibm.com/support/knowledgecenter/SS7K4U_8.5.5/com.ibm.websphere.migration.nd.doc/ae/rmig_portnumber.html?lang=en). If you need to use different port numbers, you can follow this process.

- Modify the CA specific port range, adding a row to the TCP port range input table containing the new port number.
- Determine the name of the port that needs to change by reviewing the Knowledge Center article mentioned above.
- Update the appropriate variable in server_group_variables.properties with the new port number. The table below lists the variable names, which match the port names from the Knowledge Center.
- If you are using the z/OSMF Software Configuration Catalog to provision application servers, you will need to refresh the template to make the new port number take effect.

For example, to change the admin console to listen on port 9898, add port 9898 to the CA specific port range, and update this variable definition in server_group_variables.properties:

WC_adminhost : 9898

Port Name

WC_adminhost

WC_adminhost_secure

OVERLAY_TCP_LISTENER_ADDRESS

DAEMON_PORT

DAEMON_SSL_PORT

DCS_UNICAST_ADDRESS

WC_defaulthost

WC_defaulthost_secure

BOOTSTRAP_ADDRESS

ORB_SSL_LISTENER_ADDRESS

SIB_MQ_ENDPOINT_ADDRESS

SIB_ENDPOINT_ADDRESS

SIB_MQ_ENDPOINT_SECURE_ADDRESS

SIB_ENDPOINT_SECURE_ADDRESS

SIP_DEFAULTHOST

SIP_DEFAULTHOST_SECURE

5. Associate the new port ranges with the Network Resource Group linked to the tenant template definition.

Provisioning Environment Setup

1. Create a directory containing the following files.
   - createRACF.sh
   - createRACF.xml
   - provision.xml
   - deprovision.xml
   - server_group_variables.properties
   - zpmt_response.template
2. Place the RMJOBCRD and ZOSMFRAC execs in a z/OS dataset.
3. Place the BBOSBRAC, BBOWBRA1, and BBOWBRA2 templates in a z/OS dataset. Suggested DCB parameters are LRECL 255 and RECFM VB. This can be the same dataset where RMJOBCRD and ZOSMFRAC were stored.
4. Some parameter values in server_group_variables.properties can be allowed to default, but at a minimum, the following parameters should be reviewed and updated as needed:
   - SERVER_ID_NUM – This 3 digit decimal value (including leading 0's) will normally be passed to the workflow via the REST call. During installation verification, a value can be assigned to this variable within server_group_variables.properties to enable manually running the provision and deprovision workflows from the zosmf gui. Once that initial verification has been performed, this value should be commented out within server_group_variables.properties.
   - SERVER_GROUP_PREFIX – Two character prefix used to create the jobnames, proc names, cell name, and other names used by the app server. For instance, the app server control region started task will be named xxyyyS1, where xx is the SERVER_GROUP_PREFIX, and yyy is a 3 digit decimal number.

- CONFIG_ROOT - Fully qualified path under which a mount point will be created for each provisioned app server. Note, the user id associated with the workflow created in the next step must have r/w permissions to this directory.
- DEFAULT_ADMIN_PW - Each app server will have an associated admin user id. When the server is provisioned, the password for the admin id will be set to this value. Since the password is initially set as expired, the user will have to first log on to Unix Systems Services and change the password before using the admin id to log in to the admin console.
- DOMAIN_NAME - IP domain name to be used for the app servers.
- EXEC_DATASET – Name of the z/OS dataset containing RMJOBCRD and ZOSMFRAC.
- MAX_SERVERS – Maximum number of app servers to be provisioned. This is used by the createRACF workflow to determine how many sets of RACF command execs to create.
- PROCLIB – Name of the JES PROCLIB where the app server procs will be stored. It is recommended that a separate PDSE proclib dataset be used for this purpose. Depending on your setup, it may be possible to add the new proclib dataset dynamically to the JES2 proclib concatenation with a command similar to this:
  $T PROCLIB(PROC00),DD(11)=(DSN=ZWASLOCL.PDSE.PROCLIB)
  The deprovison workflow makes use of the IDCAMS DELETE command to remove procs from the dataset when a server is deprovisioned. It is possible to use an existing proclib dataset, if your site standards allow for automation to add and delete procs from the proclib dataset. **WARNING! IF YOU USE AN EXISTING PROCLIB DATASET, AND IF MEMBERS ALREADY EXIST WITH THE SAME NAMES AS THE MEMBERS BEING CREATED BY THIS AUTOMATION, THE EXISTING MEMBERS WILL BE OVERWRITTEN!**
- RACF_COMMANDS_DSN - Partitioned dataset to be created by the createRACF workflow. This dataset will contain execs to issue the RACF commands required to set up the provisioning environment.
- RACF_GROUP - Group that will own the RACF user ids associated with the app server. This group will be used together with IRR.PWRESET.OWNER.groupname to authorize the provision workflow to reset the password of the admin user. The ZOSMFRAC exec creates a group named ZOSMFWG that can be used for this purpose.
- RACF_TEMPLATE_DSN – Name of the dataset where BBOSBRAC, BBOWBRA1, and BBOWBRA2 were placed in the previous step.
- RACF_USER - RACF user id to be used to run the provision and deprovision workflow steps that reset the admin user id password. This must be the same id that is specified for PWRESET_USER. See the next step for more information on PWRESET_USER.
- STARTING_GID - Several USS groups are required for each app server. The provisioning process has the ability to automatically assign GIDs from a pool specified by the customer. This parameter specifies the first GID in the pool. The number of GIDs in the pool will be 10 times the maximum number of servers that can be provisioned in this environment (MAX_SERVERS). Alternatively, the RACF AUTOGID parameter, rather than the logic within the provisioning process, can be used to assign GIDs. To use this RACF function, see the description of Z_CONFIG_AUTO_GID below.
- STARTING_UID - Several USS users are required for each app server. The provisioning process will automatically assign UIDs from a pool specified by the customer. This parameter specifies the first UID in the pool. The number of UIDs in the pool will be 10 times the maximum number of servers that can be provisioned in this environment

(MAX_SERVERS). Alternatively, the RACF AUTOUID parameter, rather than the logic within the provisioning process, can be used to assign UIDs. To use this RACF function, see the description of Z_CONFIG_AUTO_UID below.

- SYSPLEX_NAME - Name of the sysplex where the app servers will be provisioned
- SYSTEM_NAME - Name of the z/OS system where the app servers will run
- UID0_USER, CONSOLE_USER, PWRESET_USER, PROCLIB_USER – These variables define which SAF user ids are to be associated with various steps in the workflow. See step 5 below for more information.
- USER_HLQ – A zfs will be created for each app server. The name of the zfs will be formed from this HLQ followed by a qualifier made up of the SERVER_GROUP_PREFIX and the SERVER_ID_NUM followed by '.ZFS'
- WAS_PATH - Directory containing the WAS product code. For example, /usr/lpp/zWebSphere/V8R5.
- WORK_PATH - Directory to contain zpmt.sh work files. This can usually be set to /tmp.
- Z_CONFIG_AUTO_GID – Set this parameter to 'true' (lower case) to include the AUTOGID parameter on the commands to create the OMVS groups required for the app servers. If this parameter is set to true, STARTING_GID should be set to 0.
- Z_CONFIG_AUTO_UID – Set this parameter to 'true' (lower case) to include the AUTOUID parameter on the commands to create the OMVS users required for the app servers. If this parameter is set to true, STARTING_UID should be set to 0.
- Other Z_CONFIG_*** parameters – The volume parameter can be used to specify the volume where the zfs for the app servers are to be allocated. SMS classes may also be optionally specified. The Z_CONFIG_SPECIFY_SMS parameter must be set to true (lower case) if the DATACLAS, MGMTCLAS, or STORCLAS parameters are used.
- ZPMT_DATASET_HLQ - High level qualifier to be used when creating the dataset containing the config jobs produced by zpmt.sh.

5. Create user(s) and a group to be associated with the workflow. The process for provisioning a server is composed of multiple steps. The SAF authorization required to perform some steps is different than the authorization required to perform other steps. It is possible that the person requesting the provisioning of a new application server via the z/OSMF software catalog will not have all of the required SAF authorization. For this reason, the properties file contains four variables (UID0_USER, CONSOLE_USER, PWRESET_USER, and PROCLIB_USER) that specify which user ids z/OSMF will use to run the steps within the provisioning workflow. You can specify the same user id for all four variables, or you can specify different user ids for each variable. These user ids correspond to the following types of required authorization:

UID 0, or access to the following RACF profiles in the UNIXPRIV class:
CONTROL access to SUPERUSER.FILESYS
UPDATE access to SUPERUSER.FILESYS.MOUNT
READ access to SUPERUSER.FILESYS.CHOWN or CHOWN.UNRESTRICTED

READ access to SUPERUSER.FILESYS.CHANGEPERMS
READ access to SUPERUSER.FILESYS.PFSCTL

Authority to use the TSO CONSOLE command

Access to IRR.PWRESET.OWNER.groupname, where groupname is the group specified by the RACF_GROUP parameter in server_group_variables.properties

Update access to the PROCLIB dataset defined by the PROCLIB variable

If you would like to use a single user to run all of the steps in the workflow, specify the same user id for each of the four variables within server_group_variables.properties. For example, to associate the ZOSMFWU id with all of the steps within the workflows, specify

UID0_USER : ZOSMFWU
CONSOLE_USER : ZOSMFWU
PWRESET_USER : ZOSMFWU
PROCLIB_USER : ZOSMFWU

The ZOSMFRAC exec provides sample RACF commands for creating a single user id for running all of the steps and the group required for resetting the admin user id password. Note that if you use ZOSMFRAC to create the user, the initial password will be expired. Remember to log in to Unix Systems Services and change the password before attempting to use the id to log in to zosmf.

6. Create and perform the createRACF.xml workflow in zosmf to create RACF execs in the dataset specified by the RACF_COMMANDS_DSN variable. When creating the workflow, specify server_group_variables.properties as the workflow variable input file. Also be sure to select 'assign all steps to owner user ID'. The user id used to create and perform the workflow will need read access to createRACF.xml and read and execute access to createRACF.sh.
7. It is VERY important for the appropriate profiles, groups, and ids to be created before attempting to provision an app server. Review the RACF execs created by the createRACF.xml workflow in the previous step. Run them in this order, modifying them as needed to match your installation standards. You may want to review the CN and NOTAFTER parameters on the GENCERT commands in the BBOWBRA1 and xxyyyW execs.
    - BBOWBRA1 to create the common profiles
    - xx001S to create the user ids for app server 1
    - xx001W to permit the app server 1 user ids to the profiles
    - xx002S to create the user ids for app server 2
    - xx002W to permit the user ids for app server 2
    - ....
8. Determine which dataset will be used to contain app server JCL procedures. See the notes on the PROCLIB property in step 4 for important information. If a new dataset will be used, you will need to create it and add it to the JES2 proclib concatenation at this point. **WARNING! IF YOU USE AN EXISTING PROCLIB DATASET, AND IF MEMBERS ALREADY EXIST WITH THE SAME NAMES AS THE MEMBERS BEING CREATED BY THIS AUTOMATION, THE EXISTING MEMBERS WILL BE OVERWRITTEN!**
9. Add statements to BPXPRMxx to ensure that any config ZFSes for provisioned app servers are remounted after an IPL. The ZFS names are in the format hhhhhhhh.ssnnn.ZFS, where hhhhhhhh is the value of USER_HLQ, ss is the value of SERVER_GROUP_NAME, and nnn is a 3 digit server id number.

Usage Notes

- The deprovision and stopServer workflows open console sessions to issue a command to stop the app server. If you are also using SDSF from the same user id that is used to run the workflow, the workflow will not be able to obtain the required console session, and the workflow will fail. Our plan is to change the method used for stopping the app server in the future.