# **MAP MODULE USING IBM URBAN CODE DEPLOY**

# **Introduction:**

This document is created to explain the Map Module configuration in WebSphere using IBM Urban Code Deploy.

Each module of a deployed application must be mapped to one or more target servers. During application installation, different deployment targets might have been specified. We use the Manage modules page of the administrative console to view and change mappings.

So, this process of module mapping using UCD have been automated.

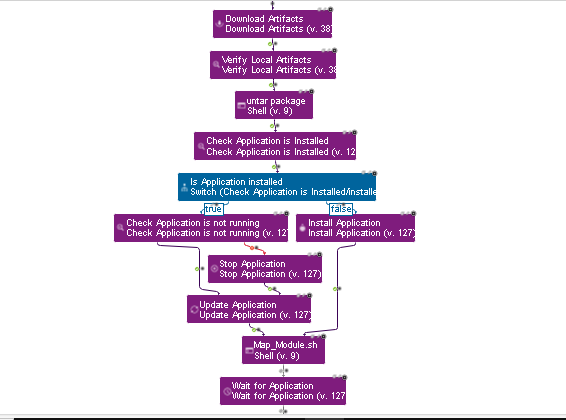
# **Prerequisites**:

Install UCD agent and java on the target server: To use this automation, UCD agent needs to be installed in the target server. Once UCD agent is installed, the target server will be visible in UCD console. While installing UCD agent, java also needs to be installed. This java will only be used by UCD agent and will be different from IBM java which will be used by WebSphere.

# **Implementation:**

In UCD, Components represent the deployable/installable items, like ear or war file, property file, etc. Ears are configured as components for each application.

For configuring manage module, the component process for each component which will be running while deployment will have to be updated.

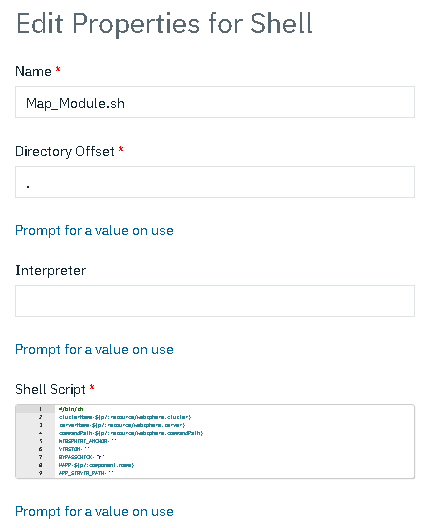


There are 2 scripts involved in this automation:

1. **Shell script**: “Map\_Module.sh” which will be capturing all the required details.
2. **Two Python scripts:** “ListWebserver.py” will be fetching the list of webservers and “WASUpdateModules.py” will be implementing the main manage module.

Map\_Module.sh

Each component process needs to be updated and add the Map\_Module.sh script for each component.



In this script, we are collecting all the required information which will be used to trigger the main python script for performing manage module.

For example: - Cluster name, Server name, Webserver list, wsadmin path, etc.

We will also trigger the python scripts through this shell script.

**Script:**

#/bin/sh

clusterName=${p?:resource/websphere.cluster}

serverName=${p?:resource/websphere.server}

commandPath=${p?:resource/websphere.commandPath}

WEBSPHERE\_ANCHOR=""

VERSION=""

BYPASSCHECK="Y"

MAPP=${p?:component.name}

APP\_SERVER\_PATH=""

mergedList=()

Fail\_Proc()

{

ERROR="WASMapModules.sh `hostname` `date +%d/%m/%Y\" \"%H:%M:%S` - $ERRORMESS"

echo $ERROR >&2

echo "WASMapModules.sh - 999 Mapping Modules Ended unsuccessfully" >&2

exit 65

}

webserverList=$(${p?:resource/websphere.commandPath}wsadmin.sh -lang jython -f /opt/IBM/WebSphere/script/ListWebserver.py | grep -v "WASX7209I: Connected to process")

echo "List of webservers is : $webserverList"

if [[ $clusterName == '' ]]; then

mergedList+=("$serverName")

mergedList+=("$webserverList")

mergedValues=$(printf "%s\n" "${mergedList[@]}")

echo "$mergedValues"

else

mergedList+=("$clusterName")

mergedList+=("$webserverList")

mergedValues=$(printf "%s\n" "${mergedList[@]}")

echo "$mergedValues"

fi

if [[ $commandPath == \*v7\* ]]; then

VERSION="v7"

echo $VERSION

WEBSPHERE\_ANCHOR="/opt/IBM/WebSphere/$VERSION"

echo $WEBSPHERE\_ANCHOR

elif [[ $commandPath == \*v8\* ]]; then

VERSION="v8"

echo $VERSION

WEBSPHERE\_ANCHOR="/opt/IBM/WebSphere/$VERSION"

echo $WEBSPHERE\_ANCHOR

else

VERSION="v6"

echo $VERSION

WEBSPHERE\_ANCHOR="/opt/IBM/WebSphere"

echo $WEBSPHERE\_ANCHOR

fi

if [ -d $WEBSPHERE\_ANCHOR ] && [ "`ls $WEBSPHERE\_ANCHOR | wc -l`" != 0 ]; then

:

else

ERRORMESS="E017 Cannot find $VERSION WebSphere on this server"

Fail\_Proc

fi

if [ -d $WEBSPHERE\_ANCHOR/AppServer ]; then

APP\_SERVER\_PATH=$WEBSPHERE\_ANCHOR/AppServer

else

if [ -d $WEBSPHERE\_ANCHOR/DeploymentManager ]; then

APP\_SERVER\_PATH=$WEBSPHERE\_ANCHOR/DeploymentManager

else

ERRORMESS="E018 Cannot detect that this is a WebSphere server"

Fail\_Proc

fi

fi

echo "MAPP value is:$MAPP"

echo "List of cluster nd webserver is: $mergedValues"

for i in $mergedValues;

do

echo $i

MAPRESULT=`"$APP\_SERVER\_PATH/bin/wsadmin.sh" -lang jython -f /opt/IBM/WebSphere/script/WASUpdateModules.py -javaoption "-Dwas.config.save=true -Dwas.deployconfig.apps=$MAPP -Dwas.deployconfig.targets=$i -Dwas.deployconfig.appsexc= -Dwas.deployconfig.bypasscheck=$BYPASSCHECK"`

ERRORCD=$?

echo "WASMapModules.sh - Mapping messages returned are:" >&2

echo "$MAPRESULT" >&2

if [ "${ERRORCD}" != 0 ]; then

ERRORMESS="E021 Failed invoking Jython script WASUpdateModules.py, Apps=$MAPPS, Exclusions=$MAPPEXP, Clusters=$MCLUSTER, error code=$ERRORCD"

Fail\_Proc

else

MAPRESULT=${MAPRESULT#\*MAPSTATUSBEGIN}

MAPRESULT=${MAPRESULT%MAPSTATUSEND\*}

RET=`echo $MAPRESULT | grep "MAPOK"`

if [ "$?" != 0 ]; then

ERRORMESS="E022 Jython script WASUpdateModules.py failed to query WAS, $MAPRESULT"

Fail\_Proc

else

echo "WASMapModules.sh - Applications mapped successfully" >&2

fi

fi

echo " "

echo "WASMapModules.sh - 000 Mapping Modules ended successfully on `hostname` `date +%d/%m/%Y\" \"%H:%M:%S`" >&2

done

ListWebserver.py

This is a python script which is used to list the webservers.

The script needs to be placed in the deployment manager in below location:

/opt/IBM/WebSphere/script/

**Script:**

serverList=AdminConfig.list("Server").split(lineSeparator)

for server in serverList:

if AdminConfig.showAttribute(server,"serverType")=="WEB\_SERVER":

serverName = AdminConfig.showAttribute(server, "name")

print(serverName)

WASUpdateModules.py

This is a python script which will perform the actual mapping of the targets.

It will take 4 parameters:

apps(the ear which needs to be mapped), targets(the target modules), appExceptions(any app which we do not want to map), bypasscheck

**Script:**

from java.lang import System

import sys

import re

def wasGetCell():

cell = AdminConfig.list('Cell').split(lineSeparator)[0]

return cell

def wasGetCellName():

return AdminConfig.showAttribute(wasGetCell(),'name')

nnameRe=re.compile("(?<=nodes\/).\*?(?=\/)")

#

# Get run time parameters

#

apps=System.getProperty("was.deployconfig.apps")

targets=System.getProperty("was.deployconfig.targets")

appExceptions=System.getProperty("was.deployconfig.appsexc")

bypasscheck=System.getProperty("was.deployconfig.bypasscheck")

print "Applications to map "+apps+"\n"

print "Applications to exclude "+appExceptions+"\n"

print "Target clusters/servers "+targets+"\n"

print "Bypass check "+bypasscheck+"\n"

if appExceptions:

appExceptions=appExceptions.split(',')

else:

appExceptions=['']

targetsQ=[]

errList=[]

targetPrefix="WebSphere:cell="+wasGetCellName()+","

if targets:

targets=targets.split(",")

if targets != ['']:

if targets==['ALL']:

print "Processing target ALL"+"\n"

serverList=AdminConfig.list("Server").split(lineSeparator)

for server in serverList:

if AdminConfig.showAttribute(server,"serverType")=="WEB\_SERVER":

nname=nnameRe.search(server).group(0)

targetsQ.append(targetPrefix+"node="+nname+",server="+AdminConfig.showAttribute(server,"name"))

elif AdminConfig.showAttribute(server,"serverType")=="APPLICATION\_SERVER":

if AdminConfig.showAttribute(server,"clusterName"):

nname=nnameRe.search(server).group(0)

targetQ=targetPrefix+"cluster="+AdminConfig.showAttribute(server,"clusterName")

if not targetQ in targetsQ:

targetsQ.append(targetQ)

else:

nname=nnameRe.search(server).group(0)

targetQ=targetPrefix+"node="+nname+",server="+AdminConfig.showAttribute(server,"name")

targetsQ.append(targetQ)

else:

processedtarget=[]

for target in targets:

print "Processing target "+target+"\n"

if (target.find('/')==-1):

clstr=AdminConfig.getid("/Cell:"+wasGetCellName()+"/ServerCluster:"+target+"/")

if clstr:

clstrName=AdminConfig.showAttribute(clstr,"name")

targetsQ.append(targetPrefix+"cluster="+clstrName)

processedtarget.append(target)

print "Cluster found "+clstrName+"\n"

continue

srvList=AdminConfig.list("Server").split(lineSeparator)

if srvList == ['']:

errList.append("Error: Target " + target + " is not a cluster or server name")

processedtarget.append(target)

continue

for srvr in srvList:

if (AdminConfig.showAttribute(srvr,"name")==target):

if (AdminConfig.showAttribute(srvr,"serverType") in ['WEB\_SERVER','APPLICATION\_SERVER']):

if not AdminConfig.showAttribute(srvr,"clusterName"):

nname=nnameRe.search(srvr).group(0)

srvrName=AdminConfig.showAttribute(srvr,"name")

targetsQ.append(targetPrefix+"node="+nname+",server="+srvrName)

print "Node and Server found "+nname+","+srvrName+"\n"

else:

errList.append("Error: Server " + srvr + " is member of a cluster. Please map modules to the cluster.")

else:

errList.append("Error: Target " + target + " is not a cluster or a valid for server types")

processedtarget.append(target)

else:

continue

else:

processedtarget.append(target)

nname=target[:target.find('/')]

sname=target[target.find('/')+1:]

srvr=AdminConfig.getid("/Cell:"+wasGetCellName()+"/Node:"+nname+"/Server:"+sname+"/")

if srvr:

if (not AdminConfig.showAttribute(srvr,"clusterName")):

nname=nnameRe.search(srvr).group(0)

targetsQ.append(targetPrefix+"node="+nname+",server="+AdminConfig.showAttribute(srvr,"name"))

print "Server: "+srvr+"\n"

else:

errList.append("Error: Server " + srvr + " is member of a cluster. Please map modules to the cluster.")

else:

errList.append("Error: Target " + target + " cannot find a server")

#

# Check all the targets were found

#

for target in targets:

targetfound="N"

for proctarget in processedtarget:

if proctarget==target:

targetfound="Y"

break

if targetfound=="N":

if bypasscheck=="N":

errList.append("Error: Cannot determine target type for "+target)

else:

print "Check bypassed for "+target+"\n"

if apps:

apps=apps.split(",")

else:

apps=['']

deployedApps = AdminApp.list().split(lineSeparator)

print "MAPSTATUSBEGIN\n"

if (deployedApps != ['']) and (apps != ['']):

for deployedApp in deployedApps:

if (((apps==["ALL"]) or (deployedApp in apps)) and (not deployedApp in appExceptions)):

modtext=AdminApp.view(deployedApp, ['-MapModulesToServers'])

modtext=modtext[modtext.find("Module:"):]

mods=modtext.split(lineSeparator)

modules=[]

for mod in mods:

if mod.find("Module: ")==0:

module=[]

module.append(mod[9:])

elif mod.find("URI: ")==0:

module.append(mod[6:])

elif mod.find("Server: ")==0:

mapping=mod[9:]

for targetQ in targetsQ:

if mapping.find(targetQ)==-1:

mapping+="+"+targetQ

module.append(mapping)

modules.append(module)

print "XXXXXXXXX Mapping "+deployedApp+" XXXXXXXXXXXX\n"

AdminApp.edit(deployedApp,['-MapModulesToServers',modules])

else:

pass

if len(errList)>0:

for err in errList:

print err+"\n"

else:

#

# If requested a save then save the updated configuration

#

if System.getProperty("was.config.save") == "true":

print "Saving to master configuration...\n"

AdminConfig.save()

print "...OK\n"

print "\n"

else:

print "No save requested, about to reset\n"

AdminConfig.reset()

print "No save performed, set was.config.save property to true, to save to master configuration\n"

print "MAPOK - OK, applications mapped successfully\n"

print "MAPSTATUSEND"

#

# End