Configure IBM Urban Code Deploy to manage Workload Automation for z/OS objects

By Maria Elena Massino

In this article I will show you step-by-step how to configure IBM® UrbanCode™ Deploy with Workload Automation plug-in to deploy Workload Automation Job Streams to a z/OS target environment.

Contents

Requirements

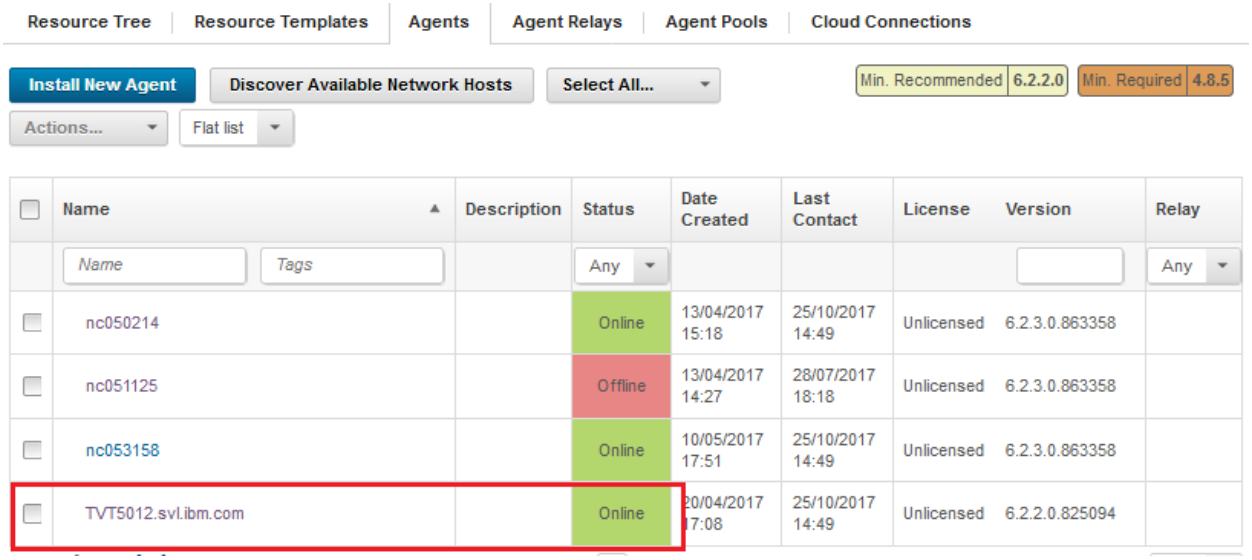
I'm using IBM UrbanCode Deploy 6.2.3 with additional Workload Automation and zOS Utility plug-ins installed, Workload Automation for z/OS 9.3 with SPE9307 (Apar PI79321) and Dynamic Workload Console 9.4.0.1.

(See the IBM UrbanCode Deploy plugins page to download the additional plug-ins and click here for instructions how to install plug-ins in IBM UrbanCode Deploy)

Prerequisites

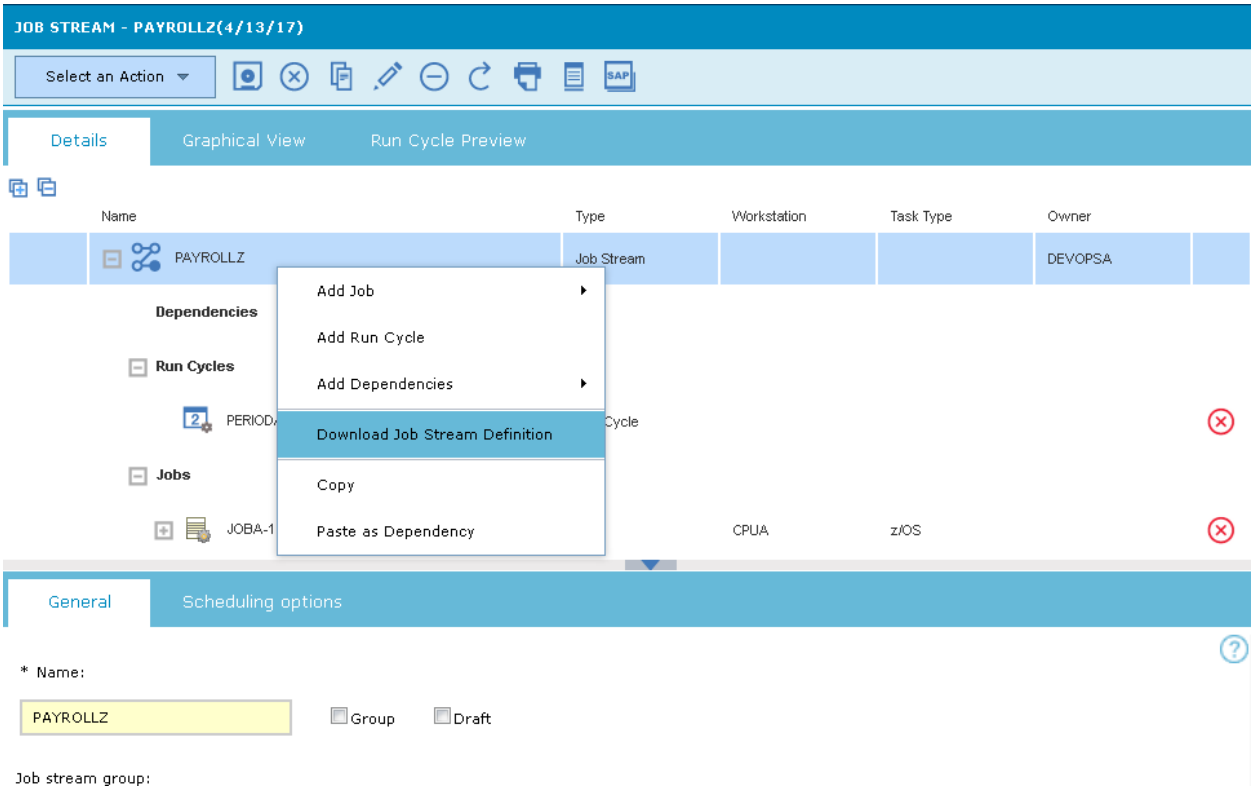
The plugin will create a JCL used as input for Submit Job step of zOS Utility plug-in so you n must install the z/OS deployment tools and configure the z/OS server for communication with IBM UrbanCode Deploy (see IBM UrbanCode documentation for details)

Verify that your IBM UrbanCode Deploy agent on the z/OS environment is installed and running by clicking Resources > Agents. Your agent must be included in the list of agents with a status of **Online**, as shown in the following figure:



Create the input file for your deployment process

A Workload Automation Job Stream can be downloaded from Workload Designer of Dynamic Workload Console. The zip file created from the "Download Job Stream Definition" action will be the input for our deployment process.



Create a component containing your Workload Automation Job Stream Definition

In IBM UrbanCode Deployments a component is a container of *artifacts* and *processes*.

Artifacts include any kind of file that is associated with a software project, in our case it consist in the zip file created from Dynamic Workload Console connected to your source environment .

Processes define the activities that components can perform, in our case it will act on the zip files and generates a JCL for submit.

Artifacts are added to a component by connecting the IBM® UrbanCode™ Deploy server to a computer system that hosts the artifacts. The server can import artifacts from build systems, source-code management systems, and file systems. Imported artifacts are stored in CodeStation, the artifact repository.

For simplicity, the artifact for your component comes from the file system on another machine where you have installed an IBM UrbanCode Deploy agent.

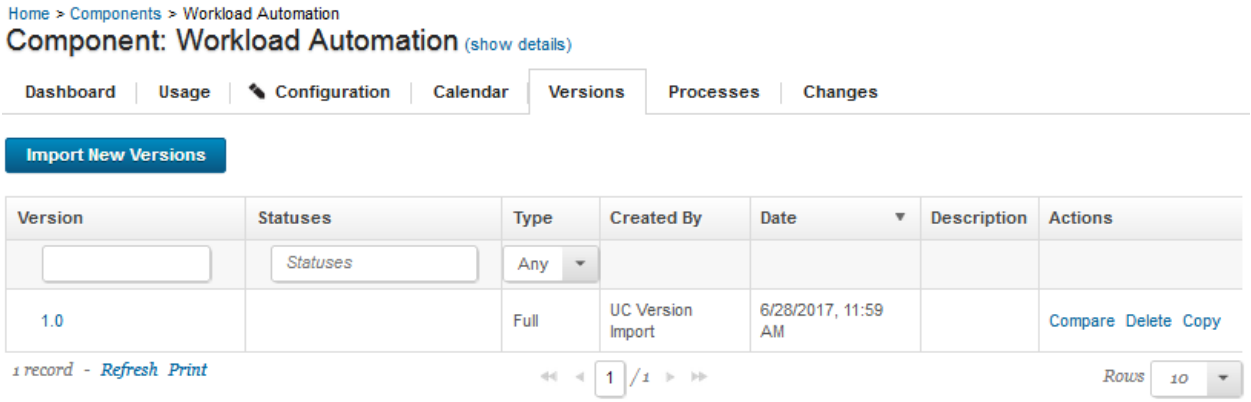
Create Workload Automation Component

1. On the machine where you have installed the IBM UrbanCode Deploy agent create a folder named **WorkloadAutomation** and inside it another folder called **1.0.** This will be the base folder for the versioning of your imports. Copy the zip file with the Workload Automation Job Stream Definition into the 1.0 folder (i.e. C:\WorkloadAutomation).

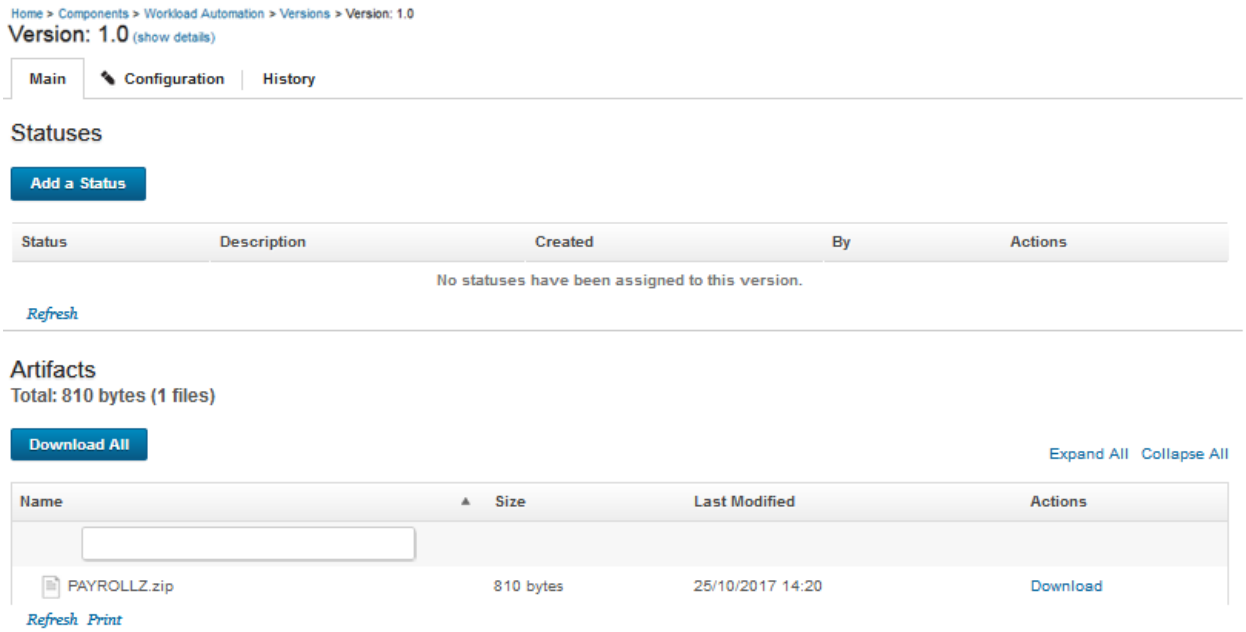
2. Click the Components tab and then click Create Component. In the window that opens, you define the component and specify the location of the artifacts for it.
3. In the Name field, type **Workload Automation**
4. In the Source Configuration Type list, select File System (Versioned). This parameter defines the type of artifacts the component uses. All artifacts in a component share the source type. The File System (Versioned) type looks for the artifacts on the file system.
5. In the Base Path field, specify the location of the **WorkloadAutomation** folder that you created earlier, such as C:\WorkloadAutomation
6. Check the radio for Import new component versions using a single agent and select the IBM UrbanCode Deploy agent where the zip file has been copied
7. Accept the default values for the other fields on the page. The Preserve Execute Permissions and Import Versions Automatically check boxes are cleared and the Copy to CodeStation check box is selected. The Default Version Type is set to Full.
8. Click Save. The Component Workload Automation is created.

Import the component version:

1. Click the Versions tab
2. Click Import New Versions. The server creates a version of the component that is based on the folder in the WorkloadAutomation folder (in our case 1.0), and imports the zip file contained in the WorkloadAutomation/1.0 directory
3. Verify that the list of versions includes version 1.0 of the component, as in the following figure:



5. Click the version name, 1.0.
6. Verify that the list of artifacts includes the zip file in the WorkloaAutomation/1.0 directory, as in the following figure:



Create a component process to deploy your Workload Automation Job Stream Definition

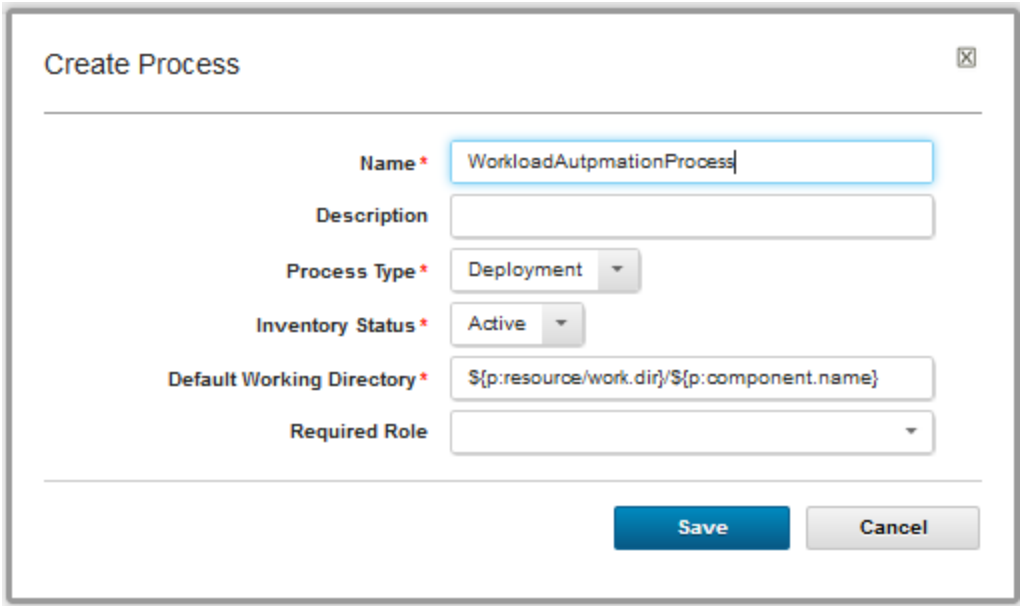
A component process is a succession of commands that are called steps. Steps can manipulate files, run system commands, set properties, pass information to other steps, and run programs. Steps are provided by automation plug-ins.

Processes are designed with the drag-and-drop process editor where you drag plug-in steps onto the design editor and configure them as you go. We will use four plug-ins, some of them come with the product as FileUtils and IBM UrbanCode Deploy and some other must be installed separately like zOS Utility Workload Automation (download them from the plug-in page of IBM UrbanCode)

Create the component process:

1. On the "Component: Workload Automation" page, click the Processes tab and then click Create Process.
2. In the Create Process window, type WorkloadAutomationProcess in the Name field.
3. In the Process Type list, select Deployment. This list has other options for processes that uninstall or configure components.
4. Accept the default values for the other fields and click Save. The Default Working Directory field becomes the folder that the agent uses to do its work, such as downloading artifacts and creating temporary files. In our case the value becomes \<*agent_install*>\var\work\WorkloadAutomation.

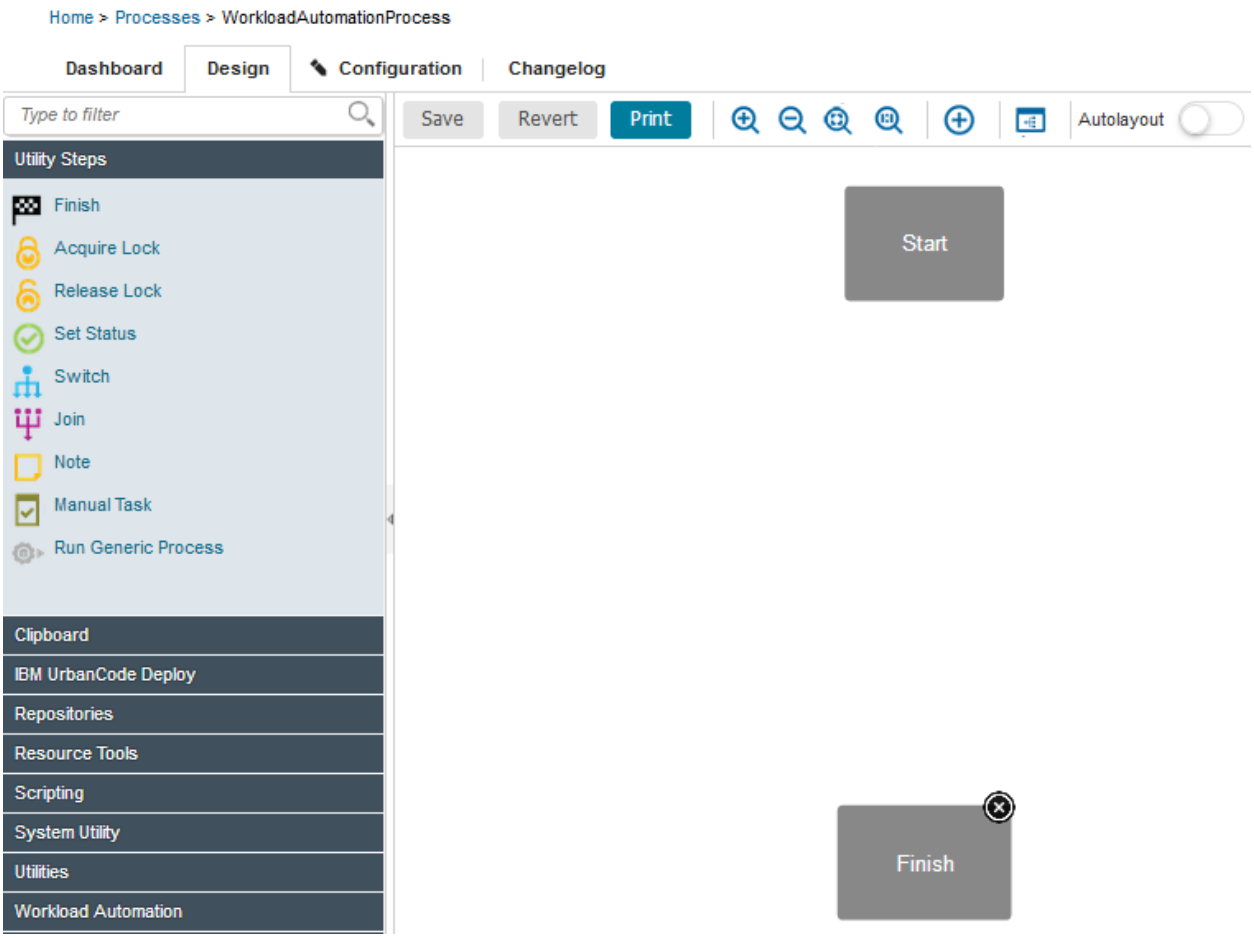
   The Create Process panel looks like:



5. Click Save

The process is listed on the Processes pane.

From the list of processes, select **WorkloadAutomationProcess**. The process opens in the process editor. The process editor lists plug-ins and steps. The required Start and Finish steps represent the beginning and the end of the process and are automatically placed on the design area. You add steps to the process by dragging them onto the design area and arranging them between the Start and Finish steps.



Your deploy process will consist in a sequence of 5 steps:

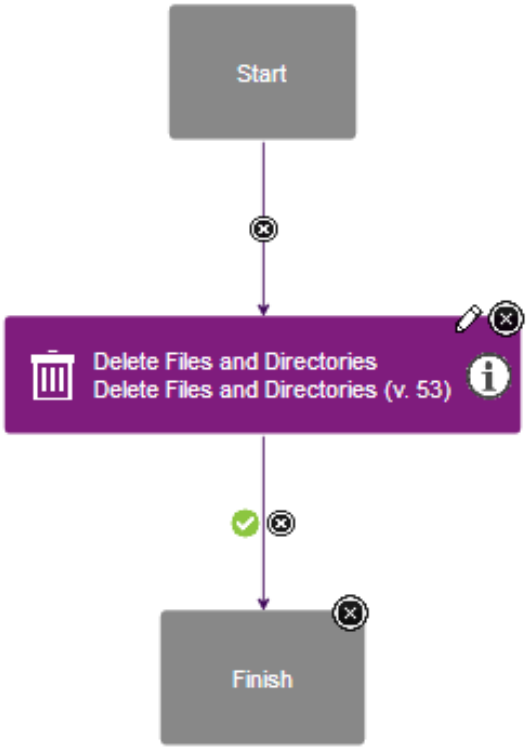
1. Clean the IBM UrbanCode Deploy agent 's working directory. To ensure that the agent works with the most recent files, remove files that remain from previous work
2. Download the artifact from the component into the working directory of the IBM UrbanCode Deploy agent
3. Unzip the file
4. Generate JCL to import Job Stream Definition into Workload Automation target environment
5. Submit the JCL to JES

Step1: clean the IBM UrbanCode Deploy agent 's working directory

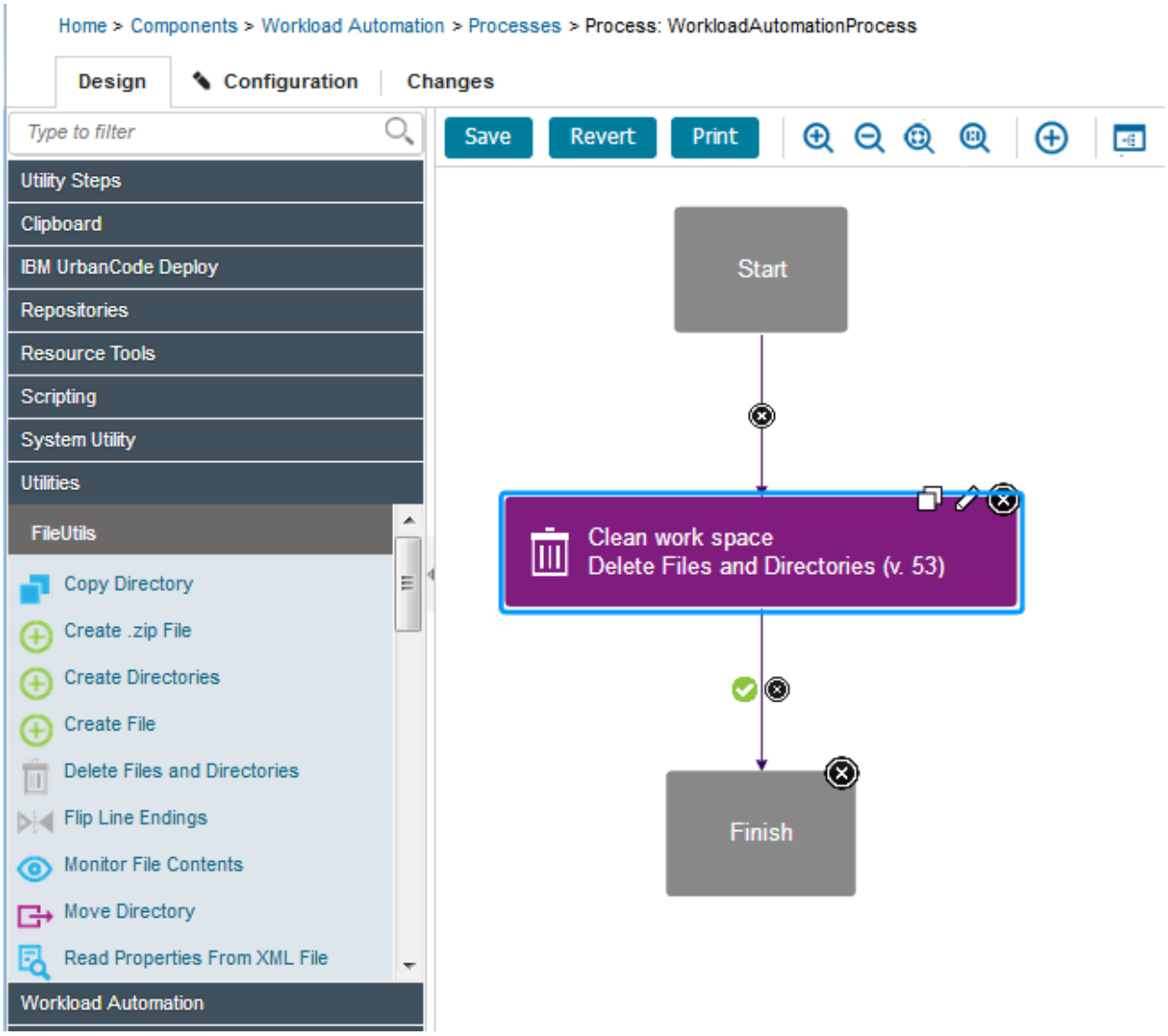
a. In the Step Palette pane on the left, expand Utilities > FileUtils.
b. Click and drag the Delete Files and Directories step to the process editor. Connect with Start and Finish steps if not done automatically as follows:

   Move the mouse over Start. The arrow icon is displayed, as the following figure shows. You use this icon to connect steps to one another.

Click and drag the arrow icon over the Delete Files and Directory step. The connecting arrow links the two steps, as the following figure shows. The direction of the arrow defines the direction of the process flow.



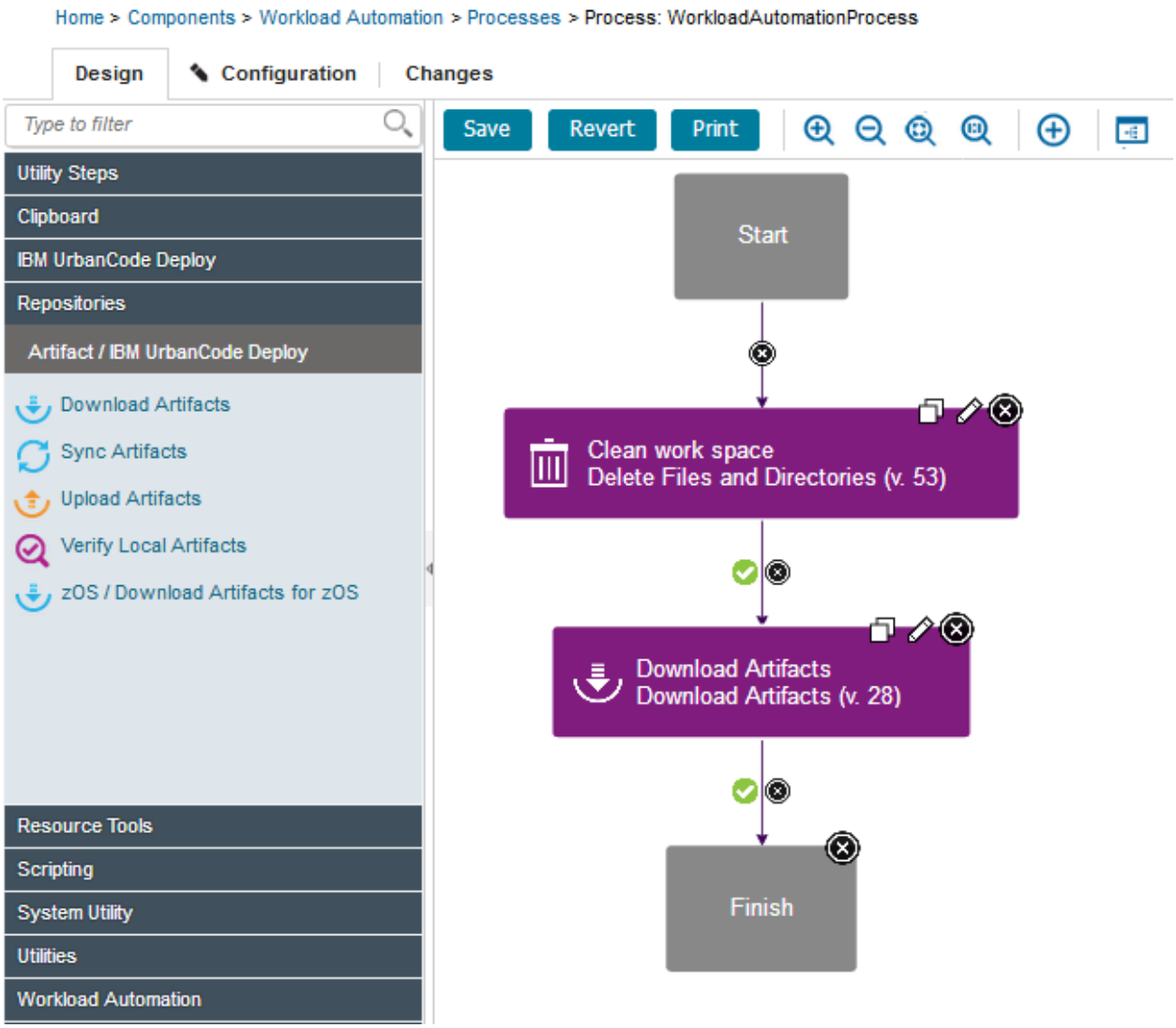
c. Click the Edit icon to open the Edit Properties window where you can modify the step properties.
d. In the Name field, specify a name, for example**, Clean work space**.
e. In the Base Directory field, specify a single period (.). This value resolves to the folder that you specified as the agent working directory.
f. In the Include field, specify an asterisk (*). This parameter instructs the agent to remove all files in the working directory.
g. Accept the default values for the other properties and then click Ok.



Step 2: download the artifact from the component into the working directory of the IBM UrbanCode Deploy agent

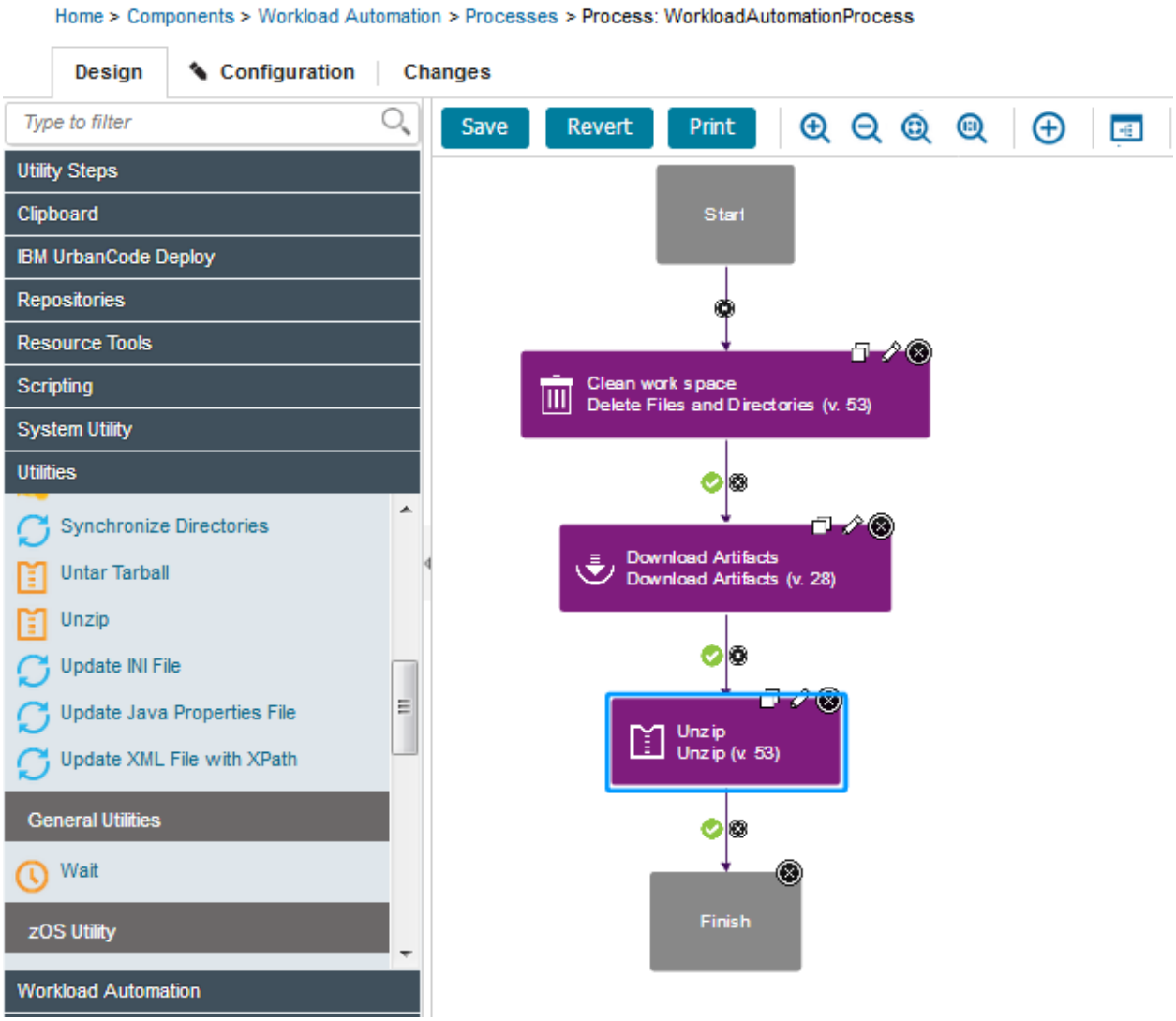
The Download Artifacts step is used in most deployment-type processes. This step downloads the specified version of the component artifacts to the agent's recently cleaned working directory. When you run the process, specify which version of the component artifacts to use.

a. In the Step Palette expand Repositories > Artifact / IBM UrbanCode Deploy and drag the Download Artifacts step to the process editor between the Clean work space step and the Finish step (If the arrows are not automatically created please proceed as described above) .
b. In the Edit Properties window, accept the default values and then click OK.



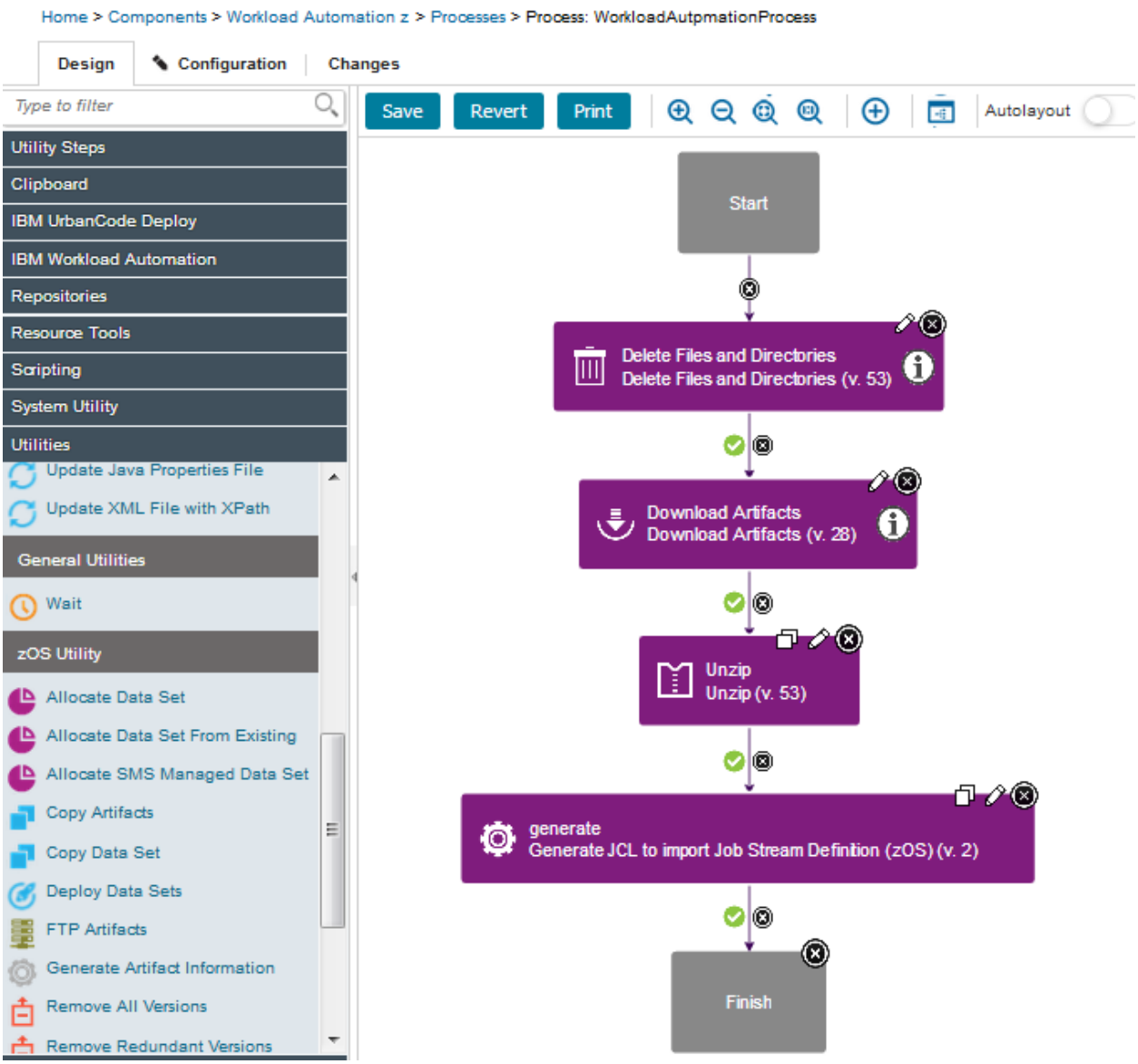
Step 3: unzip the file

a. In the Step Palette expand Utilities and drag the Unzip step to the process editor between the Download Artifacts step and the Finish step (If the arrows are not automatically created please proceed as described above) .
b. In the Edit Properties window insert *.zip in the .zip files field
c. Accept the default values for other fields and then click OK.



Step 4: Generate JCL to import the Job Stream Definition into Workload Automation target environment

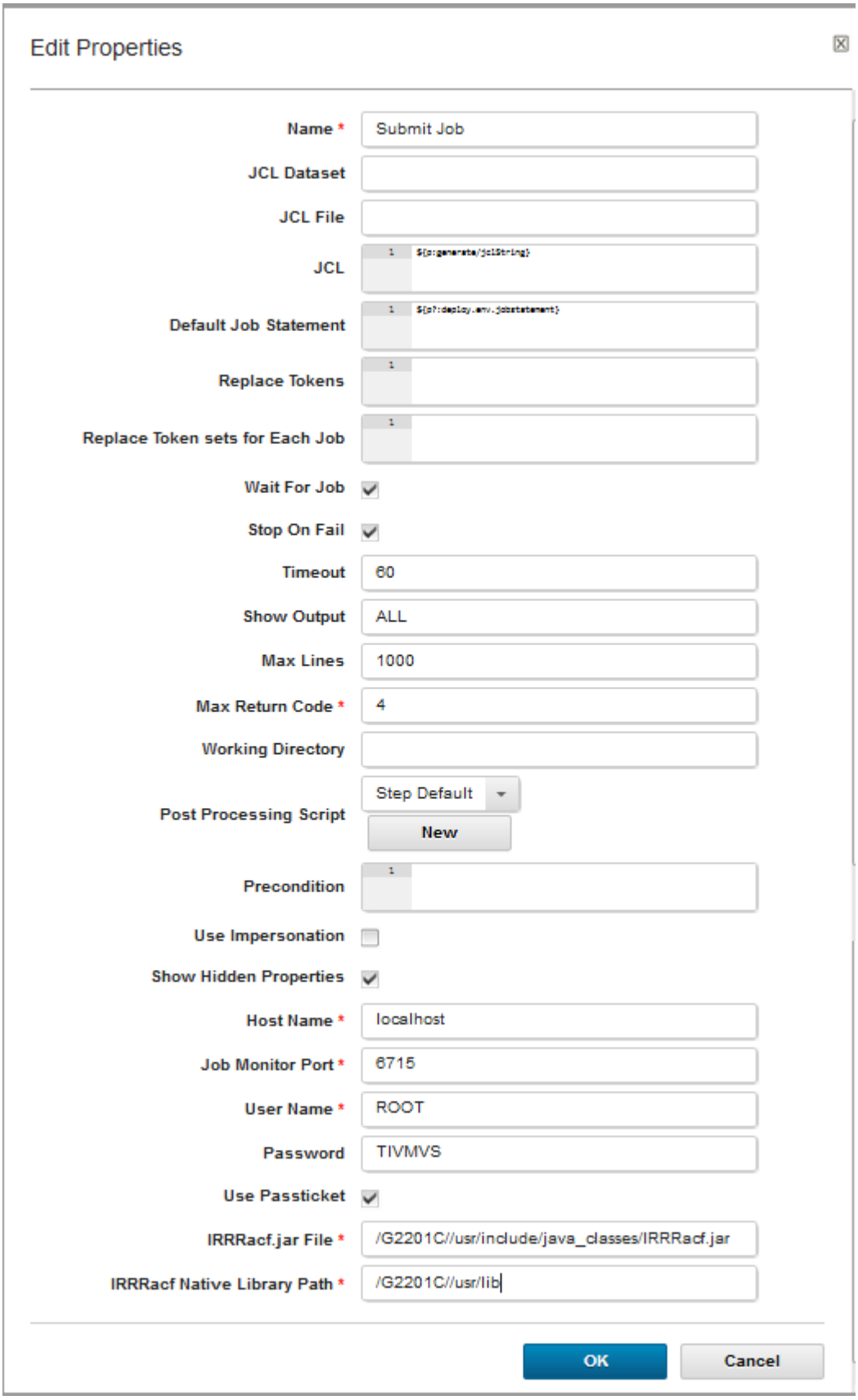
a. In the Step Palette expand Workload Automation and drag the Generate JCL to import Job Stream Definition (zOS) step to the process editor between the Unzip step and the Finish step (If the arrows are not automatically created please proceed as described above) .
b. Open the Edit Properties window
c. Inserta a name for the step (i.e. generate); this name will be used to import the generated JCL as input for the next step
d. Optional: insert the Job Card for your JCL (i.e NOTIFY=&SYSUID,MSGCLASS=9)
e. Insert the name of the library containing the JCL samples of your workload Automation installation (i.e. TWSTST.TWSR.JCL)
f. Insert the name of your subsystem (i.e. TWSR)
g. Accept the default values for other fields and then click Ok.



Step 5: Submit Job

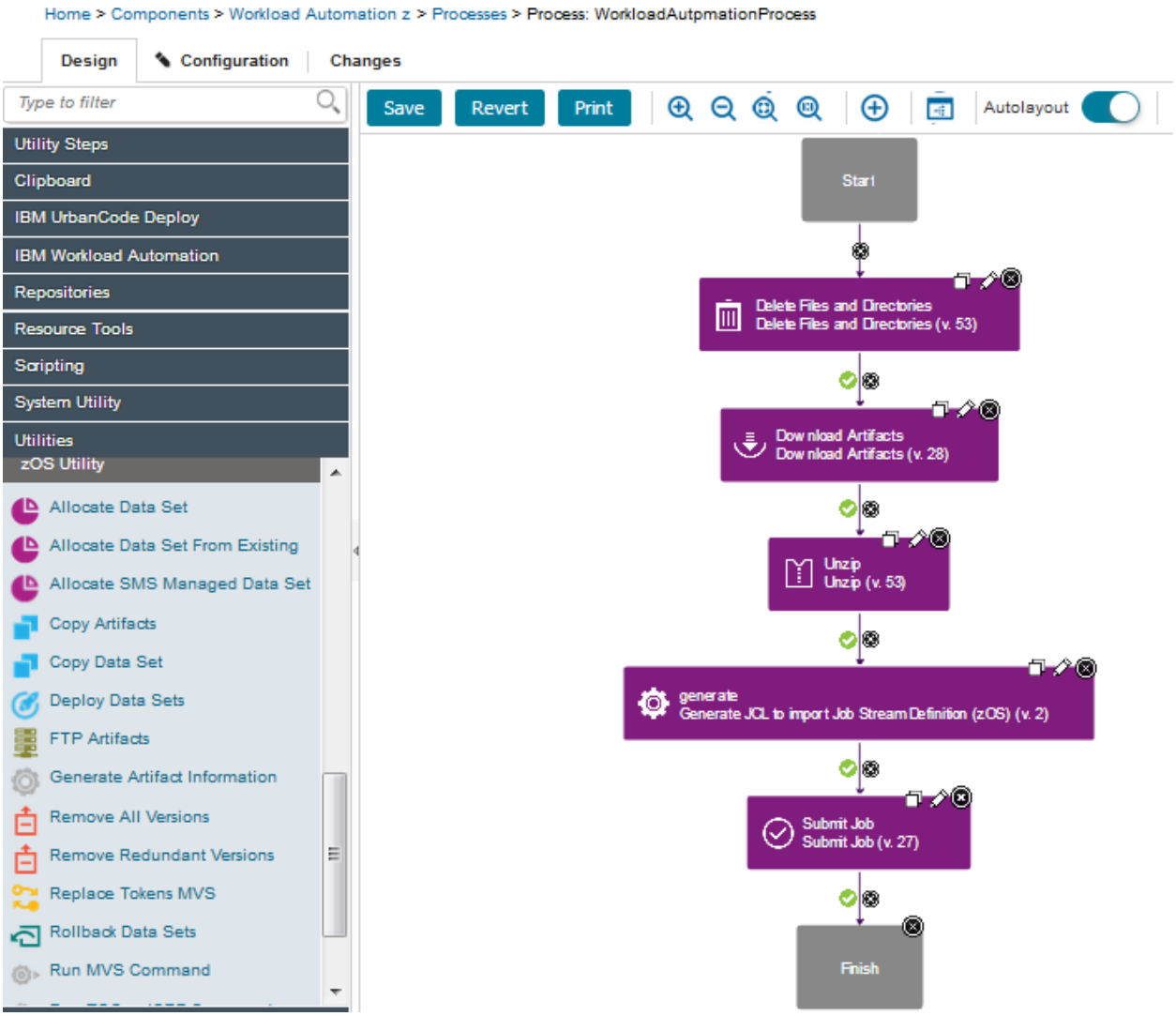
a. In the Step Palette expand Utilities > zOS Utility and drag the Submit Job step to the process editor between the generate step and the Finish step (If the arrows are not automatically created please proceed as described above) .
b. Open the Edit Properties window
c. In the JCL field insert **${p:generate/jclString}**; this syntax tells the step to use the exported variable name jclString from the *generate* step. This variable contains the customized JCL to submit to import the Job stream definition
d. Clean the filed Default Job Statement
e. Click on Show Hidden Properties to show mandatory fields for submit
f. Insert “localhost” in the Host Name
g. Insert the JES Job Monitor port (default is 6715)
h. Insert the User Name and Password

i. Specify the full path to the System Access Facility (SAF) JAR file, which is IRRRacf.jar. The default value is /usr/include/java_classes/IRRRacf.jar.
j. Specify the path to the System Access Facility (SAF) native library, which is libIRRRacf.so. There is one library for 31-bit Java and one for 64-bit Java. You must specify the path of the appropriate library based on the version of Java that you are running. The default value is /usr/lib.
k. Accept the default values for other fields and then click Ok.


Edit Properties
Name *
Submit Job
JCL Dataset
JCL File
JCL
${c:generate/jclString}
Default Job Statement
${c?:deploy.env.jobstatement}
Replace Tokens
Replace Token sets for Each Job
Wait For Job
Stop On Fail
Timeout
60
Show Output
ALL
Max Lines
1000
Max Return Code *
4
Working Directory
Post Processing Script
Step Default
New
Precondition
Use Impersonation
Show Hidden Properties
Host Name *
localhost
Job Monitor Port *
6715
User Name *
ROOT
Password
TIVMVS
Use Passticket
IRRRacf.jar File *
/G2201C//usr/include/java_classes/IRRRacf.jar
IRRRacf Native Library Path *
/G2201C//usr/lib
OK
Cancel

Save the Process

Your process should now like as



Clik on Save button to save your process.

Create an application to manage components process to deploy your Workload Automation Job Stream Definition

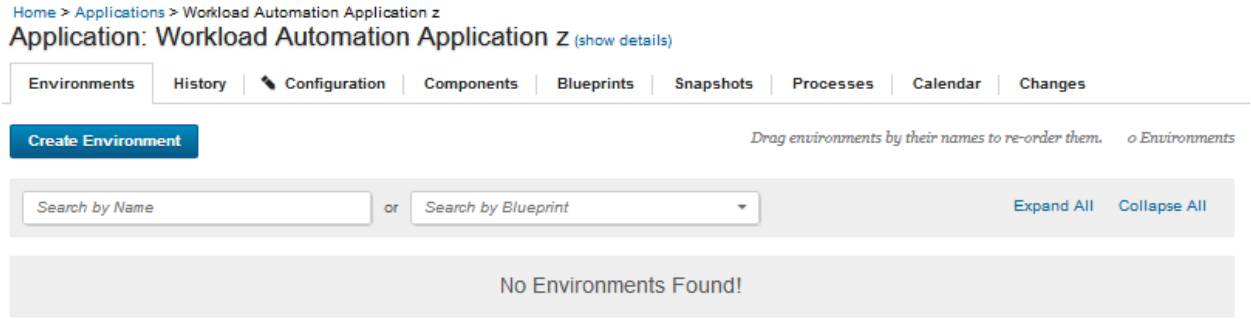
Applications manage components, typically by deploying them into environments.

Your application will manage Workload Automation component; you need to define at least one environment into which the component is deployed; and create a process to do the work. An

environment maps the component to agents and handles inventory, among other things. An application process is similar to but not identical to a component process. Application processes are primarily intended to direct underlying component processes and orchestrate multi-component deployments. In our case you create an application and assign the Workload Automation component to it.

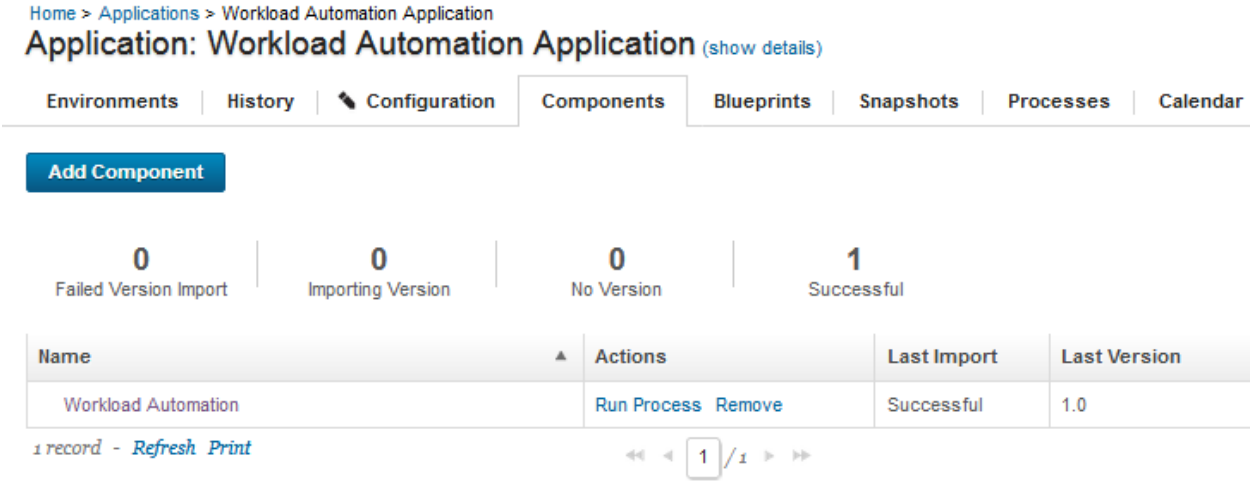
Create the Workload Automation application

1. Click the Applications tab and then click Create Application.
2. Name the new application something like **Workload Automation Application**.
3. Accept the default values for the other fields and click Create.



Add the Workload Automation component to the Workload Automation Application application:

1. Click the Components tab.
   NB: this is the components tab associated with the application, not the Components tab at the top of the page.
2. Click Add Component.
3. In the Add a Component window, select the Workload Automation component and then click Save.

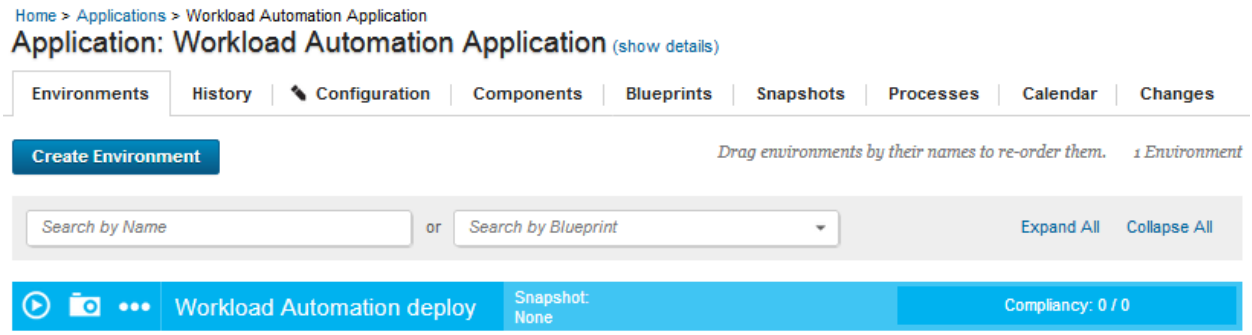


Configure an application environment

An environment is a user-defined collection of resources that identify the components that can be deployed by the parent application, along with the agents that do the work.

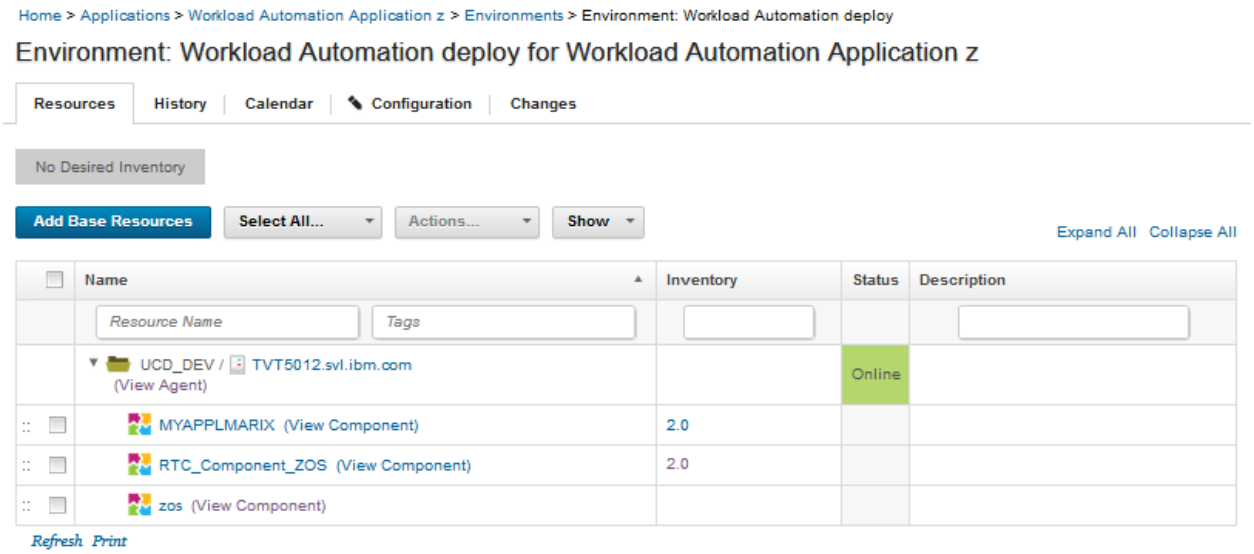
Define the environment

a. In the Application: Workload Automation Application page, click Environments.
b. Click Create Environment.
c. Specify the name to be something like Workload Automation deploy
d. Accept the default values in the other fields in this window and click Save.



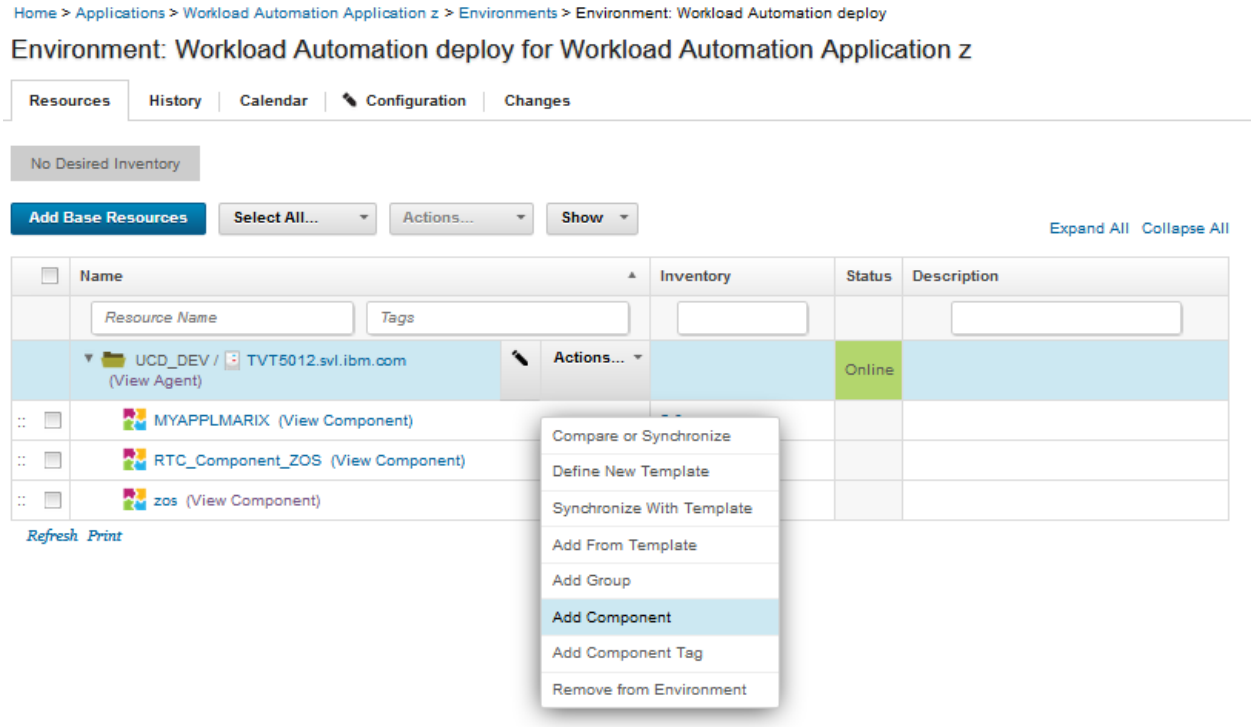
Add resource to the environment

e. Click the environment name (i.e. Workload Automation deploy).
f. Using the Resources tab for the environment, click Add Base Resources. The Add Resource to Environment window shows all available resources.
g. Select the check box next to resource group containing the agent running on the machine where your IBM UrbanCode Deploy agent is installed and then click OK. When you select the resource group, you automatically select the resources associated with it, such as the agent you assigned to the group. Selecting an agent-type resource identifies the location, usually a computer, where deployments can occur.

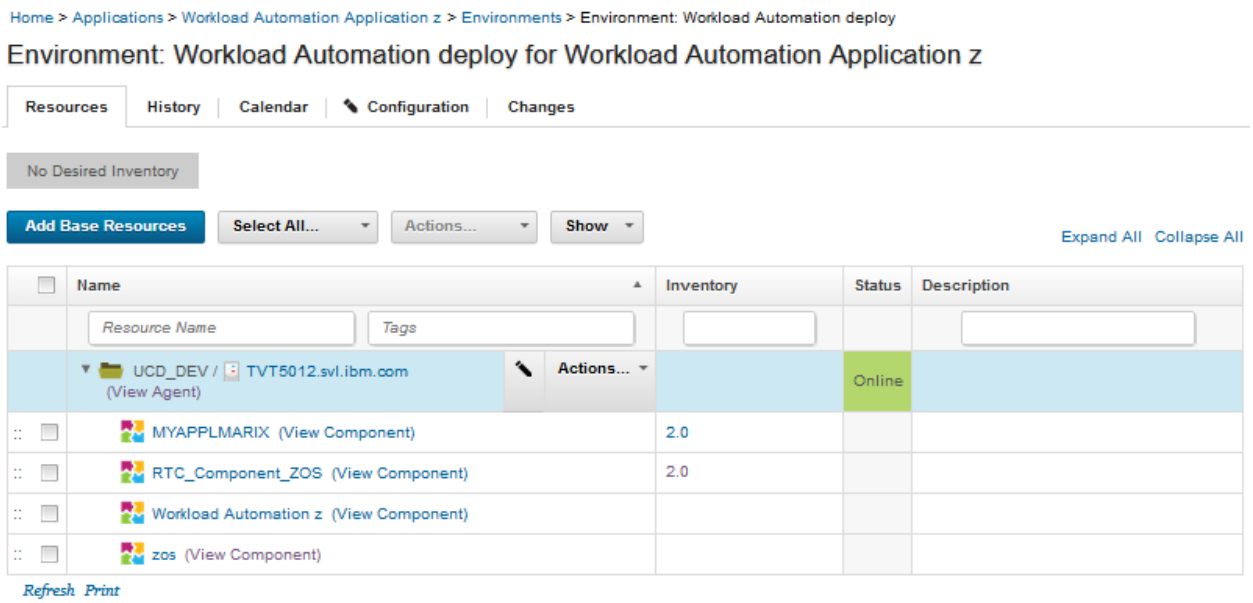


Map the Workload Automation component to this agent resource

h. As you hover the mouse over the row with the agent resource, click Actions > Add Component.



i. Select the Workload Automation component and then click Save



Configure the Environment Properties

Click on Configuration tab in the Workload Automation deploy environment and then click on Environment Properties to configure the environment properties to apply to you environment.

In the Environment properties you can set all the specific for this environment and will be inherited from your component process.

In particular you can set here all the properties used by the Generate JCL to import Job Stream Definitions to generate the TRANSLATE statements in the JCL.

By properly configuring the Environment Properties you can rename objects inside the exported Job Stream definitions to match the target environment objects (i.e. the Workstation Name).

Use the following syntax to define environment properties for the Generate JCL to import Job Stream Definition step:

Name: <prefix><objectType>_<originalName>

Value : <newName>

Where

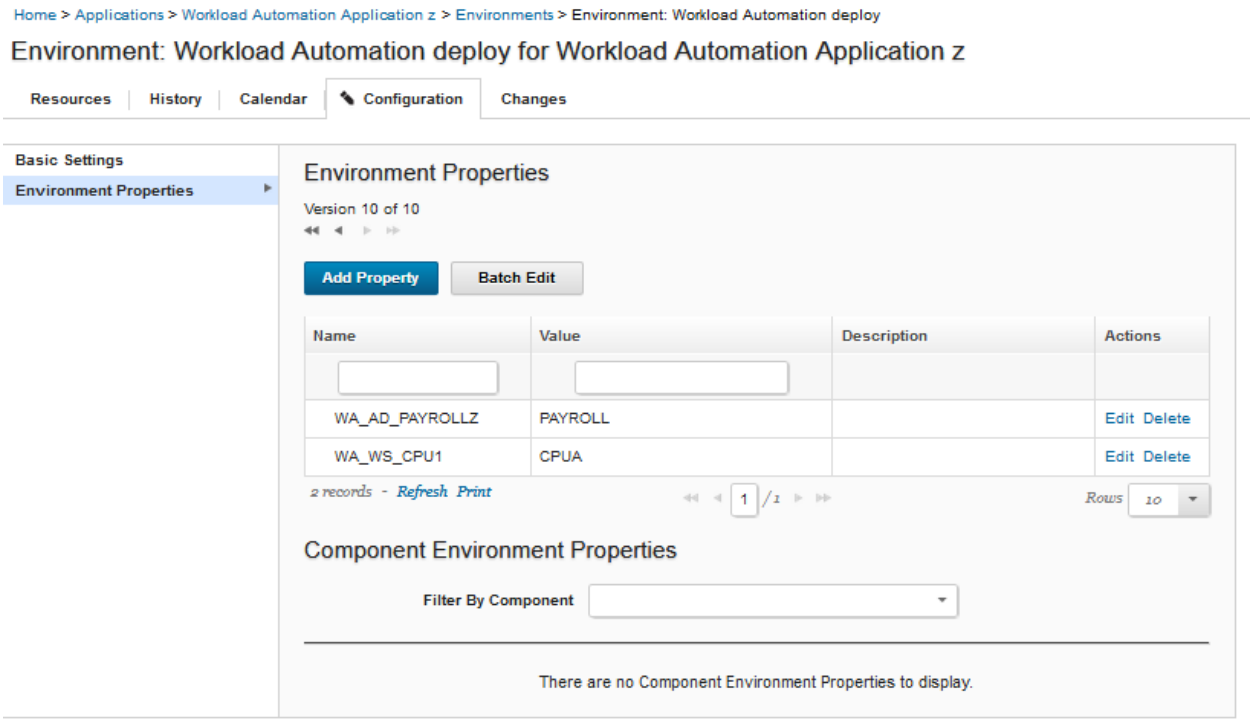
<prefix> is the prefix set in the Workload Automation Prefix in the Generate JCL to import Job Stream Definition step of your Component's process.

<objectType > is one of the following

AD	Job Stream Name
CL	Calendar Name
JS	Job Name
OW	Job Stream Owner
PR	Period Name

	SR	Resource Name
	WS	Workstation Name
<originalName>	Is the name of the corresponding object type to replace	
<neName>	is the new name replacing the originalName	

For example in my Environment I defined the following Environment Properties:



The first property means that the original Job Stream name PAYROLLZ will be renamed to PAYROLL in the deployed Job Stream;

The second property means that all occurrences of Workstation CPU1 will be replaced with CPUA in the deployed Job Stream

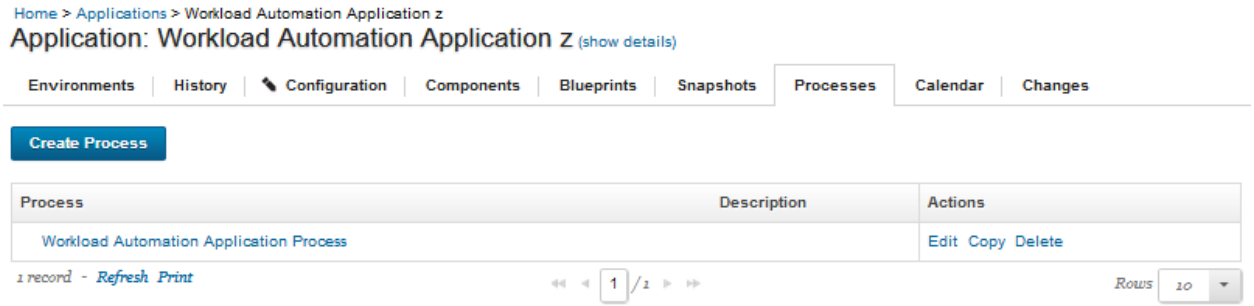
The application process

An application process, like a component process, consists of steps that are configured with the process editor. In our case the application process will deploy the Workload Automation component by calling the component process that you created earlier but it could handle multiple components. You must use application processes to deploy components

Create the application process

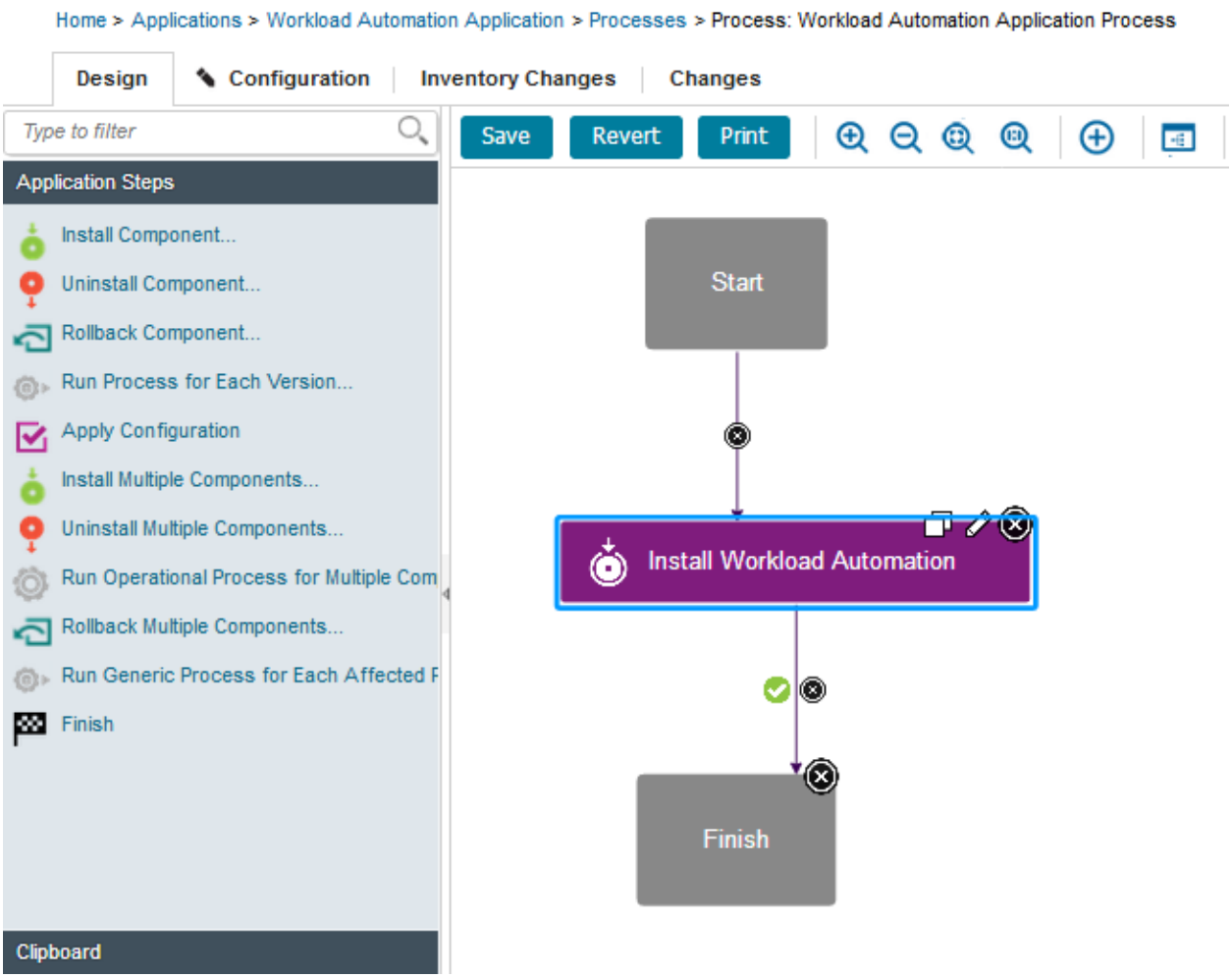
1. Click the Applications tab and then click Workload Automation Application.
2. Click Processes and then click Create Process.
3. In the Create an Application Process window, name the new application process something like **Workload Automation Application Process**.

4. Accept the default values for the other fields and click Save.



Add steps to the application process

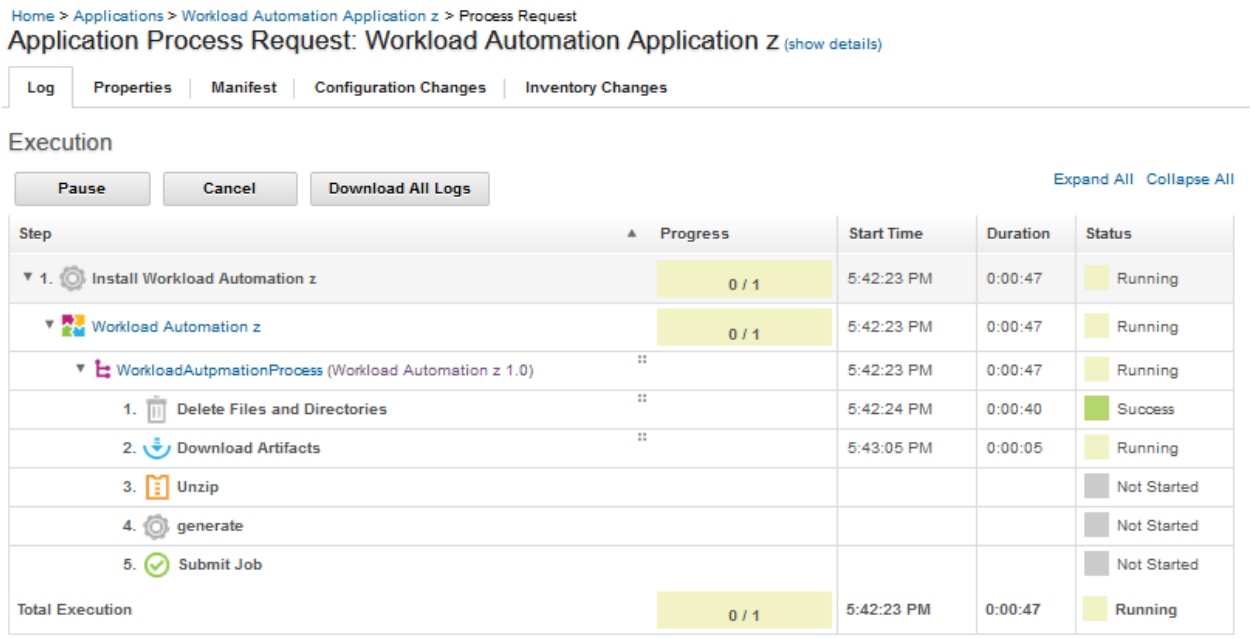
5. Click on **Workload Automation Application Process** to open the new process in the process editor.
6. Add a step that deploys the **Workload Automation** component:
    a. From the list of steps, add an Install Component step to the design area.
    b. In the Edit Properties window, name the step something like **Install Workload Automation.**
    c. All the fields should be correctly prefilled anyway assure that in the Component list you have selected the **Workload Automation component**. All components that are associated with an application are available.
    d. In the Component Process list assure that you have selected the **WorkloadAutomationProcess** component process. If multiple processes were defined for the Workload Automation component, they are listed.
    e. Accept the default values for the other fields and click OK.
7. Connect the Start step to the Install Workload Automation step.
8. Connect the Install Workload Automation step to the Finish step.
9. Save the process by clicking the Save icon



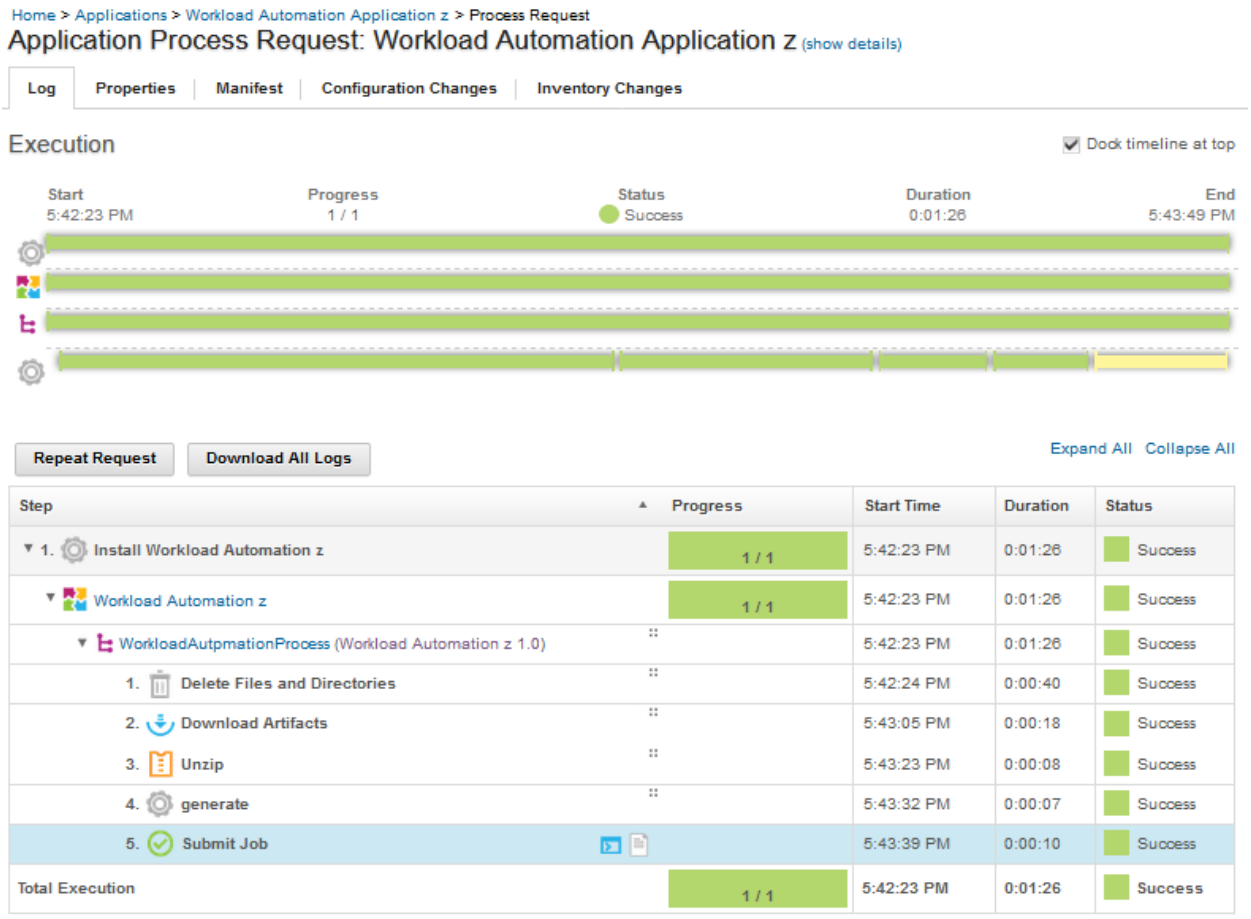
Deploy the Workload Automation component

The final step to have the object defined on the target environment is to run the application process on your environment.

1. Open the application page by clicking Applications and then clicking the application name.
2. In the same row as your environment, click the Request Process icon  .
3. In the Run Process window, accept the default value for the Only Changed Versions parameter. If this check box is selected, no previously deployed component versions are deployed.
4. In the Process list, select the Workload Automation Application Process.
5. Click Choose Versions. The Component Versions window opens.
6. In the Component Versions window, click Select For All, and then select Latest Available. Click OK to return to the Run Process window.
7. Click Submit. The Application Process Request page shows the progress of the request. From here, you can watch as the process runs. The following figure shows the running process if you Expand all the tree:



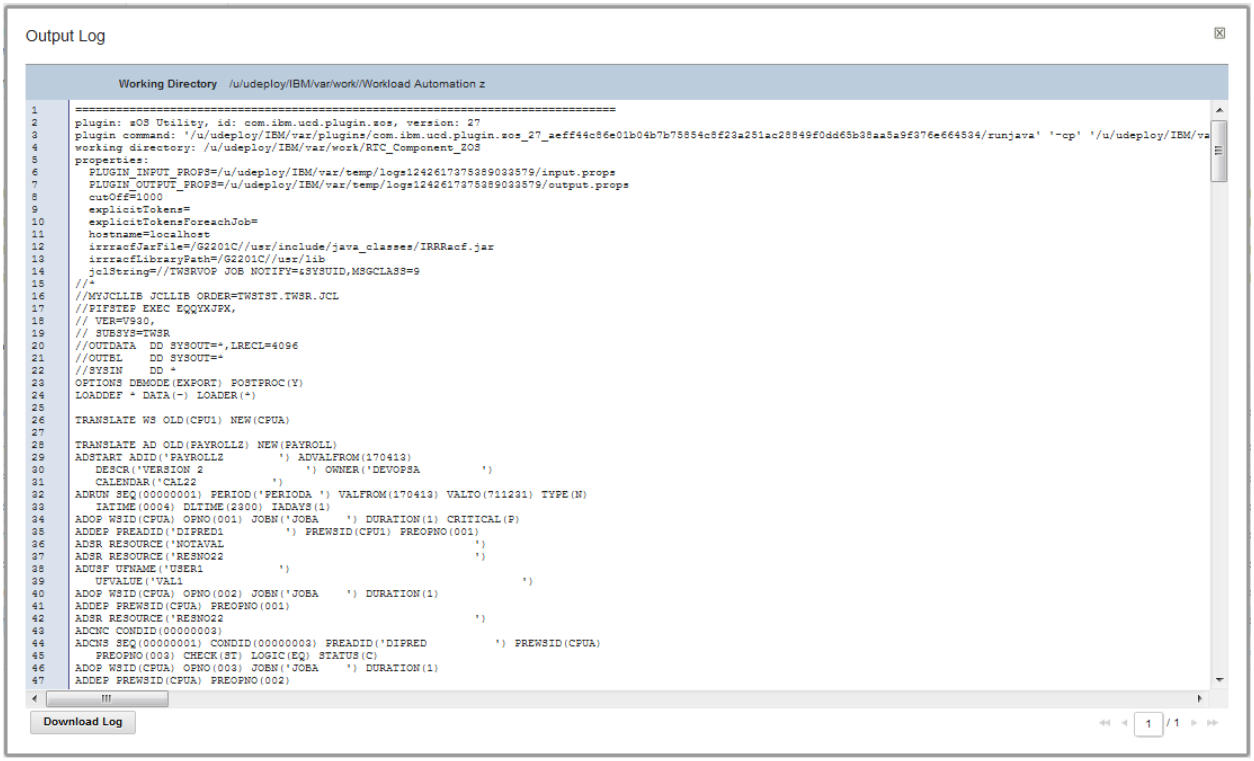
When the process finishes, the Success status is shown, as in the following figure:



To inspect the step result hover the mouse over the step process and the icons  appear for Output Log and Input/Output Properties respectively .

Output Log for Generate JCL to import Jobs Stream Definitions step

The output Log for generate step shows the generated JCL generated using the content of the zip file




```
================================================================================
plugin: zOS Utility, id: com.ibm.ucd.plugin.zos, version: 27
plugin command: '/u/udeploy/IBM/var/plugins/com.ibm.ucd.plugin.zos_27_aeff44c86e01b04b7b75854c8f23a251ac28849f0dd65b38aa5a9f376e664534/runjava' '-cp' '/u/udeploy/IBM/va
working directory: /u/udeploy/IBM/var/work/RTC_Component_ZOS
properties:
  PLUGIN_INPUT_PROPS=/u/udeploy/IBM/var/temp/logs1242617375389033579/input.props
  PLUGIN_OUTPUT_PROPS=/u/udeploy/IBM/var/temp/logs1242617375389033579/output.props
  cutOff=1000
  explicitTokens=
  explicitTokensForeachJob=
  hostname=localhost
  irrracfJarFile=/G2201C//usr/include/java_classes/IRRRacf.jar
  irrracfLibraryPath=/G2201C//usr/lib
  jclString=//TWSRVOP JOB NOTIFY=&SYSUID,MSGCLASS=9
//*
//MYJCLLIB JCLLIB ORDER=TWSTST.TWSR.JCL
//PIFSTEP EXEC EQQYXJPX,
// VER=V930,
// SUBSYS=TWSR
//OUTDATA  DD SYSOUT=*,LRECL=4096
//OUTEL    DD SYSOUT=*
//SYSIN    DD *
OPTIONS DBMODE(EXPORT) POSTPROC(Y)
LOADDEF * DATA(-) LOADER(*)

TRANSLATE WS OLD(CPU1) NEW(CPUA)

TRANSLATE AD OLD(PAYROLLZ) NEW(PAYROLL)
ADSTART ADID('PAYROLLZ        ') ADVALFROM(170413)
   DESCR('VERSION 2             ') OWNER('DEVOPSA         ')
   CALENDAR('CAL22          ')
ADRUN SEQ(00000001) PERIOD('PERIODA ') VALFROM(170413) VALTO(711231) TYPE(N)
   IATIME(0004) DLTIME(2300) IADAYS(1)
ADOP WSID(CPUA) OPNO(001) JOBN('JOBA    ') DURATION(1) CRITICAL(P)
ADDEP PREADID('DIPRED1         ') PREWSID(CPU1) PREOPNO(001)
ADSR RESOURCE('NOTAVAL                                  ')
ADSR RESOURCE('RESNO22                                  ')
ADUSF UFNAME('USER1          ')
   UFVALUE('VAL1                                        ')
ADOP WSID(CPUA) OPNO(002) JOBN('JOBA    ') DURATION(1)
ADDEP PREWSID(CPUA) PREOPNO(001)
ADSR RESOURCE('RESNO22                                  ')
ADCNC CONDID(00000003)
ADCNS SEQ(00000001) CONDID(00000003) PREADID('DIPRED          ') PREWSID(CPUA)
   PREOPNO(003) CHECK(ST) LOGIC(EQ) STATUS(C)
ADOP WSID(CPUA) OPNO(003) JOBN('JOBA    ') DURATION(1)
ADDEP PREWSID(CPUA) PREOPNO(002)
```

Note the TRANSLATE statements generated by the Environment Properties to rename objects

Result

At this point you can open the Dynamic Workload Console on the target engine and verify that your Job Stream is defined with the correct translation applied: