

IBM® UrbanCode Deploy

IBM Workload Automation Plug-in

Document version 1.1

IBM Workload Automation

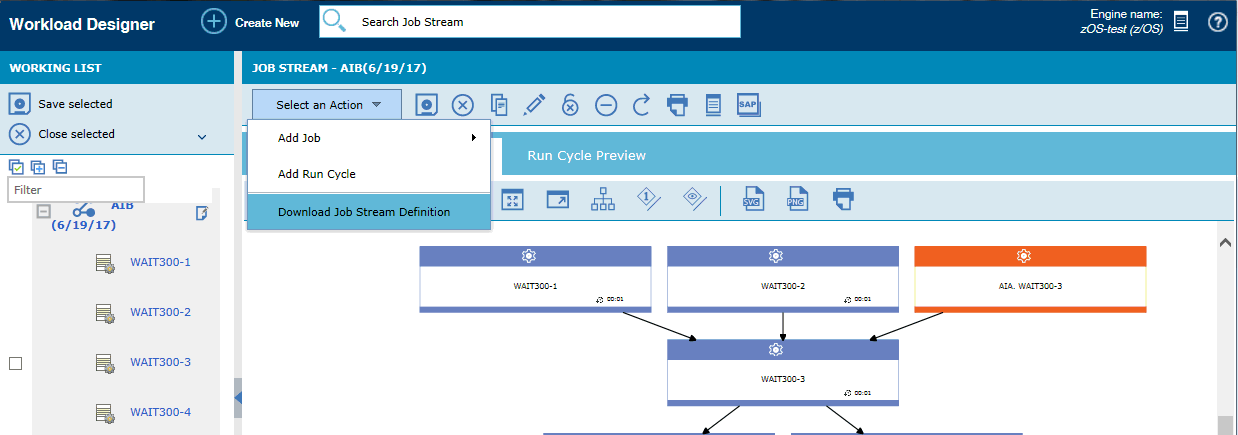
Development team



# Overview

The IBM Workload Automation plug-in is a facility provided to help you moving workload automation definitions from one environment to another, in a distributed or z/OS environment. You can download a job stream definition from the Dynamic Workload Console, then IBM UrbanCode Deploy applies the appropriate environment parameters for the importing process that is performed by the plug-in’s steps.

To download a job stream definition, from the Dynamic Workload Console open the Workload Designer and select the job stream you want. From the Select an Action pop-up menu, select Download Job Stream Definition, as shown in the following example:



The process steps in the IBM Workload Automation plug-in includes:

* [Import Job Stream Definition (Distributed)](#_Import_Job_Stream)  
  This step invokes the wappman command of Workload Automation to import the job stream definition contained in a Workload Automation Template, appropriately configured.
* [Generate JCL to import Job Stream Definition (zOS)](#__RefHeading__16426_1704686280)  
  This step generates a JCL that invokes the EQQYXJPX procedure to import the application definition. This JCL will be then submitted from the Submit Job step of the z/OS Utility plug-in. For details about the z/OS Utility plug-in, see <https://developer.ibm.com/urbancode/plugin/zos-utility/>  
    
  **Note:** The Submit Job step requires the job server component that is included with IBM UrbanCode Deploy, Rational Team Concert, or Rational Developer for System z.

## IBM Workload Automation compatibility

* IBM Workload Scheduler version 9.4, Fix Pack 1
* IBM Workload Scheduler for z/OS version 9.3.0.7 (SPE released in June 2017)
* Dynamic Workload Console version 9.4, Fix Pack 1

## Installation

No special steps are required for installing the IBM Workload Automation plug-in (for details, see [Installing plug-ins in UrbanCode products](http://developer.ibm.com/urbancode/docs/installing-plugins-ucd/))

Because the Generate JCL to import Job Stream Definition step must be used together with the Submit Job step of the z/OS Utility plug-in, ensure that you have installed and configured the z/OS deployment tools before you use the plug-in. To learn how to install and configure the z/OS deployment tools, see [Deploying to the z/OS platform](http://www.ibm.com/support/knowledgecenter/SS4GSP_6.1.1/com.ibm.udeploy.doc/topics/deploying_zos.html). You must also configure the job server component before you run the Submit Job step.

## UrbanCode Deploy requirements

* In the distributed environment, ensure that you installed the UrbanCode Deploy agent on the workstation (either a master domain manager or a dynamic agent) where you run the wappman command.
* In the z/OS environment, ensure that you installed the UrbanCode Deploy agent on the system where you run the EQQYXJPX procedure.

# Steps

The process steps in the IBM Workload Automation plug-in includes:

* [Import Job Stream Definition (Distributed)](#_Import_Job_Stream)
* [Generate JCL to import Job Stream Definition (zOS)](#__RefHeading__16426_1704686280)

## Import Job Stream Definition (Distributed)

The following table shows the input properties for the Import Job Stream Definition (Distributed) step.

| Name | Type | Description | Required |
| --- | --- | --- | --- |
| Custom Encoding | String | Character encoding to be used when accessing the properties file. This field takes precedence over Use System Encoding. | N |
| Directory Offset | String | The directory related to the current working directory containing the files to operate on. | N |
| Environment  Properties | String | Leave the default to import from the environment properties, or insert a list of properties separated by commas. | N |
| Name | String | Name of the step. | Y |
| Replace Existing Definition | Boolean | Select this check box if the objects that were created at first deploy must be updated, if they are still present in the database, created if they are no longer present, and deleted if no longer present in this version of the objects. | N |
| Use System Encoding | Boolean | Select this check box to use the local character encoding of the agent when accessing the properties file. | N |
| Workload Automation Path | String | Workload Automation installation path. | N |
| Workload Automation Prefix | String | Prefix to be used in environment properties for Workload Automation mapping. Default is WA\_ | N |

## Generate JCL to import Job Stream Definition (zOS)

The following table shows the input properties for the Generate JCL to import Job Stream Definition (zOS) step.

| Name | Type | Description | Required |
| --- | --- | --- | --- |
| Custom Encoding | String | A character encoding to be used when accessing the properties file. This field takes precedence over Use System Encoding. | N |
| Directory Offset | String | The directory related to the current working directory containing the files to operate on. | N |
| Environment  Properties | String | Leave the default to import from the environment properties, or insert a list of properties separated by commas. | N |
| JCL Samples Library | String | Name of the library containing the JCL samples. | Y |
| Job Card | String | Additional parameters for job card. | N |
| Name | String | Name of the step. | Y |
| Subsystem Name | String | Name of the subsystem. | Y |
| Use System Encoding | Boolean | Select this check box to use the local character encoding of the agent when accessing the file with the definition. | N |
| Workload Automation Prefix | String | The prefix to be used in the environment properties for Workload Automation translation. Default is WA\_ | N |

# Configuring the environment properties

According to the system you are using, insert the environment properties to set how objects are to be defined in your target environment

For details, see the following sections:

* [Configuring the environment properties (distributed)](#_Configuring_the_environment)
* [Configuring the environment properties (z/OS)](#_Configuring_the_environment_1)

## Configuring the environment properties (distributed)

The import step uses the files contained in the exported zip file, appropriately customized for the target environment.

For each environment of your application, select the Configuration tab and open the Environment Properties frame.

Select Add Property to define how each object is to be mapped into the target environment.

### Customizing the mapping file

The mapping file contains a list of elements, some of which are dependent on the topology of the environment in which the file is used.

The objects contained in the workload application are extracted to the definitions file with the same definition they have in the source environment. The definitions file can contain a complete object definition or, in some cases, only a name or reference to the object that is extracted. Simple references and not a full object definition is extracted for those objects that require to be mapped to an object already present in the target environment. For some objects extracted by reference, the object definition is written to the mapping file, which requires manual customization to map the objects in the Workload Scheduler source environment to the environment where the workload application will be deployed.

The type of object can be one of the following:

* JOBSTREAM
* JOB
* RUNCYCLEGROUP
* VARTABLE
* CALENDAR
* RESOURCE
* PROMPT
* WORKSTATION
* REFJOBSTREAM
* REFJOB
* EVENTRULE
* VARIABLEVALUE

Fill in the following fields according to the following syntax:

**Name:** <prefix><object-type>\_<old-value>  
where:  
 <prefix> is the Workload Automation Prefix that you set in the step (default is WA\_)   
<object-type> is the type of object   
<old-value> is the value of the object in the original environment  
  
**Value:** <new-value> is the value to be taken at the target environment

### Applying regular expressions

As an alternative, you can apply mapping rules defined using advanced regular expressions, according to Java standards:

**Name:** <prefix>**REGEX\_**<object-type>\_<SELECTIONREGEX>  
where:  
 <prefix> is the Workload Automation Prefix that you set in the step (default is WA\_)  
<object-type> is the type of object   
<SELECTIONREGEX> is the regular expression that defines the search pattern.   
  
**Value:** <REPLACEREGEX> is the regular expression that defines the replacing value of each matching element.

The following table shows an example of the resulting mapping file:

| **Name** | **Value** |
| --- | --- |
| WA\_WORKSTATION\_NC0500113 | NC0542147 |
| WA\_ JOBSTREAM\_ORD\_3\_NEW\_VER | ORD\_3\_VER |
| WA\_ JOB\_NOTIFY | NOTIFY |
| WA\_ JOBSTREAM\_DEV\_PAYROLL | PRD\_PAYROLL |
| WA\_EVENTRULE\_ ^(.+?)\_DEV$ | $1\_PRD |
| WA\_RUNCYCLEGROUP\_ ^A24Y(.)$ | B42X$1 |
| WA\_VARTABLE\_ ^(.+?)\_DEV$ | $1\_PRD |

## Configuring the environment properties (z/OS)

The Generate JCL to import Job Stream Definition step uses the file contained in the exported zip file and add that translation statements for the target environment, as defined in the environment properties.

For each environment of your application, select the Configuration tab and open the Environment Properties frame. Select Add Property to define how each object is to be translated into the target environment. Fill in the following fields according to the following syntax:

**Name:** <prefix><object-type>\_<old-value>  
where:  
 <prefix> is the Workload Automation Prefix that you set in the step (default is WA\_)   
<object-type> is the type of object (AD, CL, JS, OW, PR, SR, or WS)  
<old-value> is the value of the object in the original environment  
  
**Value:** <new-value> is the value to be taken at the target environment

The following table shows some examples of environment properties:

| **Name** | **Value** |
| --- | --- |
| WA\_AD\_OLDAPPL | NEWAPPL |
| WA\_CL\_CALENDAR11 | CALENDAR22 |
| WA\_JS\_OLDJOB | NEWJOB |
| WA\_OW\_JOHNNY | JAMES |
| WA\_PR\_PERIOD1 | PERIOD2 |
| WA\_SR\_RES11 | RES22 |
| WA\_WS\_CPU1 | CPU2 |

# Building a process

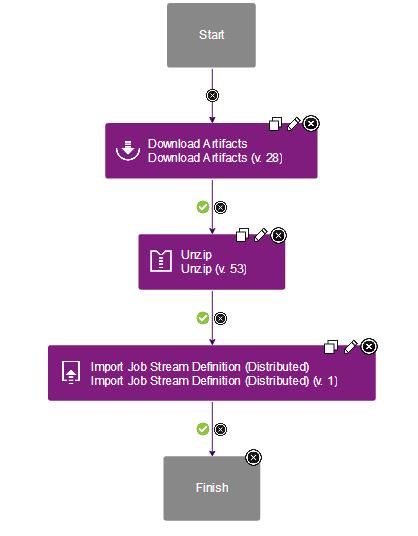
## Distributed environment

The Import Job Stream Definition step takes as input the files contained into the exported zip file.

A simple process is to combine the Import Job Stream Definition (Distributed) with other basic steps from UrbanCode Deploy. For example, if your zip file is imported into the component, you could define a process to:

1. Download Artifacts: it downloads the zip file from UrbanCode Deploy to the UrbanCode Deploy agent working directory (or any other that you set in the step properties).
2. Unzip: extracts all the files into the UrbanCode Deploy agent working directory (or any other that you set in the step properties).
3. Import Job Stream Definition (Distributed): applies the environment properties to the extracted files from the UrbanCode Deploy agent working directory (or any other you set in the step properties), and import the objects into workload automation.

The following figure shows the complete process:



## z/OS environment

The Generate JCL to import Job Stream Definition step takes as input the file contained in the exported zip file.

A simple process is to combine the Generate JCL to import Job Stream Definition with other basic steps from UrbanCode Deploy and with the Submit Job step of the z/OS Utility plug-in. For example, if your zip file is imported into the component, you could define a process to:

1. Download Artifacts: downloads the zip file from UrbanCode Deploy to the UrbanCode Deploy agent working directory (or the directory you set as Offset Directory)
2. Unzip: extracts the file to the UrbanCode Deploy agent working directory (or any other you define in the step)
3. Generate JCL to import Job Stream Definition: reads the object definition from the file extracted into the UrbanCode Deploy agent working directory (or any other you define in the step) and generate the JCL containing the application definition to be passed to the EQQYXJPX procedure together with the environment properties applied.   
   The step also exports the property ***jclString*** that must be used as input for the Submit Job step.
4. The Submit Job step: submits the JCL.  
   Set the JCL parameter of this step as   
   **${p:<generate\_step\_name >/jclString}**where <generate\_step\_name> is the name of the Generate JCL to import Job Stream Definition step (in the following example, **generate**).

The following figure shows the complete process:

