Kelverion

INTEGRATION PACK FOR BMC BLADELOGIC SERVER AUTOMATION

For Microsoft System Center Orchestrator

User Guide

Version 3.9

Kelverion Integration Pack for BMC BladeLogic Server Automation

Released: January 2021

Feedback

Send suggestions and comments about this document to support@kelverion.com

Contents

Getting Started

The Kelverion Integration Pack for BMC BladeLogic Server Automation is an add-on for System Center Orchestrator that enables you to automate a number of specific BladeLogic server management actions, including BladeLogic server configuration, server configuration comparison (audit) and software package deployment. In addition, the Integration Pack provides a generic activity for running BLCLI commands, which allows you to perform most BMC BladeLogic Server Automation tasks.

System Requirements

The Kelverion Integration Pack for BMC BladeLogic Server Automation requires the following software to be installed and configured prior to implementing the integration. For more information about installing and configuring Orchestrator and the BMC BladeLogic Server Automation application, refer to the respective product documentation.

- Microsoft System Center Orchestrator *
- Microsoft .NET Framework 4.5.2

As well as:

- BMC BladeLogic Server Automation 8.9 application server
- BMC BladeLogic Server Automation 8.9 web service API (SOAP)

Or:

- BMC TrueSight Server Automation 20.02 application server
- BMC TrueSight Server Automation 20.02 web service API (SOAP)

* Please see Kelverion.com/orchestrator for the latest Orchestrator support information.

Registering and Deploying the Integration Pack

After you download the integration pack file, you must register it with the Orchestrator management server and then deploy it to Runbook Servers and Runbook Designers. For more information about how to install integration packs, see the How to Install an Integration Pack (https://technet.microsoft.com/en-us/library/hh420346.aspx).

To register the integration pack:

1. On the management server, copy the **.OIP** file for the integration pack to a local hard drive or network share.
2. Confirm that the file is not set to **Read Only** to prevent unregistering the integration pack later.
3. Start the **Deployment Manager**.

4. In the navigation pane of the Deployment Manager, expand **Orchestrator Management Server**, right-click **Integration Packs** to select **Register IP with the Orchestrator Management Server**. The **Integration Pack Registration Wizard** opens.
5. Click **Next**.
6. In the **Select Integration Packs or Hotfixes** dialog box, click **Add**.
7. Locate the **.OIP** file that you copied locally from step 1, click **Open** and then click **Next**.
8. In the **Completing the Integration Pack Wizard** dialog box, click **Finish**.
9. On the **End User Agreement** dialog box, read the Kelverion License Terms**,** and then click **Accept**.
10. The **Log Entries** pane displays a confirmation message when the integration pack is successfully registered.

To deploy the integration pack:

1. In the navigation pane of the **Deployment Manager**, right-click **Integration Packs**, click **Deploy IP to Runbook Server or Runbook Designer**.
2. Select the integration pack the you want to deploy, and then click **Next**.
3. Enter the name of the runbook server or computers with the Runbook Designer installed, on which you want to deploy the integration pack, click **Add**, and then click **Next**.
4. Continue to add additional runbook servers and computers running the Runbook Designer, on which you want to deploy the integration pack. Click **Next**.
5. In the **Installation Options** dialog box, configure the following settings.
6. To choose a time to deploy the integration pack, select the **Schedule installation** check box, and then select the time and date from the **Perform installation** list.
7. Click one of the following:
    a. **Stop all running runbooks before installing the integration pack** to stop all running runbooks before deploying the integration pack.
    b. **Install the Integration Packs without stopping the running Runbooks** to install the integration pack without stopping any running runbooks.
8. Click **Next**.
9. In the **Completing Integration Pack Deployment Wizard** dialog box, Click **Finish**.
10. When the integration pack is deployed, the Log Entries pane displays a confirmation message.

Licensing the Integration Pack

After you register and deploy the integration pack you must provide a valid Kelverion license before running any runbooks that contain activities from the integration pack

To deploy the integration pack license file:

1. Copy the .KAL license file to %PROGRAMFILES(X86)%\Kelverion Automation\Licenses
2. Repeat for each Orchestrator Runbook Server and Runbook Designer host system.

Connecting to BMC BladeLogic

A connection establishes a reusable link between Orchestrator and a BMC BladeLogic Server Automation application server. You can create as many connections as you require specifying links to multiple servers running BMC BladeLogic Server Automation. You can also create multiple connections to the same server to allow for differences in security permissions for different user accounts.

To set up a BMC BladeLogic Server Automation connection:

1. In the Runbook Designer, click the **Options** menu, and select BMC BladeLogic Server Automation. The BMC BladeLogic Server Automation dialog box appears.
2. On the **Configurations** tab, click **Add** to begin the connection setup. The **Add Configuration** dialog box appears.
3. In the **Name** box, enter a name for the connection. This could be the name of the BMC BladeLogic application server or a descriptive name to distinguish the type of connection.
4. In the **Type** box select **Connection Configuration**, which is the common configuration required by all activities.
5. In the **BLCLI Command Definition Folder** specify the path of the folder containing XML definition files for your specific version of BLCLI. For more details see BLCLI Command Definition. Default value is:

   C:/Program Files (x86)/Common Files/Microsoft System Center 2012/Orchestrator/ Extensions/Support/Integration Toolkit/4CBD4DD4-4222-49C5-B189-C71C9668D44E/ BLCLICommands_20.02.01.106.

6. In the **BladeLogic Host** box, type the name or IP address of the BMC BladeLogic application server. If you are using the computer name, you can type the NetBIOS name or the fully qualified domain name (FQDN).
7. In the **BladeLogic Port** box, type the port used to access the Web Service on the BMC BladeLogic application server.
8. In the **BladeLogic User Name** and **Password** boxes, type the credentials that Orchestrator will use to connect to the BMC BladeLogic application server.
9. In the **BladeLogic Role** box, type the type the role of the user used to connect to the BMC BladeLogic application server.
10. In the **Authentication Type** box, type the type of authentication used to connect to the BMC BladeLogic application server. The integration pack currently supports only SRP authentication mechanism.
11. In the **Timeout (minutes)** box, type the number of minutes a web service request should be running for, before it times out.
12. In the **Skip Certificate Validation** box, specify if you want the IP to perform server certificate validation or not. When set to **True**, the IP will not perform certificate validation, typically used in secure environments, when working with trusted servers and self-signed certificates. When set to **False**, the IP will validate the server certificate. The server must be configured with a valid certificate signed by a valid certificate authority and the specified **BladeLogic Host** must be listed on the certificate.

13. Add additional connections if applicable.
14. Click **OK** to close the configuration dialog box, and then click **Finish**.

BLCLI Command Definition

The Run BLCLI Command Activity requires command definition information in order to display available namespaces, commands and arguments, for a specific BLCLI version. The integration pack installs command definitions for BLCLI version **8.9.00.136** and **20.02.01.106**, respectively. These are installed in separate folders at the same location where the integration pack assemblies are installed:

C:/Program Files (x86)/Common Files/Microsoft System Center 2012/Orchestrator/ Extensions/Support/Integration Toolkit/4CBD4DD4-4222-49C5-B189-C71C9668D44E

You can configure the integration pack to use a specific BLCLI command definition version by configuring the **BLCLI Command Definition Folder** property in the integration pack configuration to point to the appropriate location. By default the integration pack uses version **20.02.01.106**.

Modifying BLCLI Command Definition

In cases where a particular BLCLI command is not available from any of the installed BLCLI command definition folders you have one of the following options:

- Modify existing command definition XML files and add necessary namespace, command and argument information so that it becomes available in Run BLCLI Command Activity.
- Use the Run Custom BLCLI Command Activity. This activity allows you to configure and run a BLCLI command which is not available in the Run BLCLI Command Activity through existing command definitions.

BLCLI command definition information is stored in XML files which are relatively simple to modify by following examples from existing files.

Make sure to backup original files in case you need to reverse your changes.

Adding a New Namespaces to your Command Definition

Follow these steps to add a new namespace to existing BLCLI command definition:

1. Locate the **Namespaces.xml** file in the BLCLI command definition folder you wish to modify.
2. Add a new **Namespace** element with **Name** attribute specifying the new namespace name. Note that namespace names are case sensitive and they must match exact names as specified in BLCLI documentation.
3. Create a new **<Namespace Name>.xml** file. The top level Namespace element in this file must specify the new namespace name. The hierarchy of a namespace file is as follows:

```
Namespace
    Commands
        Command 1
            Arguments
                Argument 1
                Argument 2
                …
                Argument n
        Command 2
            Arguments
            …
        Command 3
        …
        Command m
```

Adding a New Command to a Namespace

Follow these steps to add a new command to an existing namespace definition:

1. Locate the **<Namespace Name>.xml** file.
2. Add a new **Command** child element under the **Commands** node. Specify the name of the command in the **Name** attribute. Note that command names are case sensitive and they must match exact names as specified BLCLI documentation.
3. If the command has arguments, add a new **Arguments** child element under the Command element.
4. Add **Argument** child elements as required. Argument element attributes:
    - **Name** attribute must contain the argument name.
    - **Index** attribute must specify the argument position in the command, starting at 1. First argument should have index 1, second argument should have index 2, etc.
    - **Type** attribute must specify argument type information.

Working with Activities in Orchestrator

This integration pack adds the BMC BladeLogic Server Automation category to the **Activities** pane in the Runbook Designer. This category contains the following activities:

- Create BLPackage from Audit
- Create Deploy Job
- Run Job
- Get Job Running Status
- Export Audit Results
- Set Server Property
- Run BLCLI Command
- Run Custom BLCLI Command

Common Configuration Instructions for All Activities

The following configuration instructions apply to all activities in this integration pack. Links to this section are included in the configuration instructions for each activity.

Activity Properties

Each activity has a set of required or optional properties that define the configuration of that activity. This includes how it connects to other activity or how the activity performs its actions. You can view or modify activity properties in the Runbook Designer.

To configure the properties for an activity:

1. Double-click the activity. Alternatively, you can right-click the activity, and then click **Properties**.
2. Click **Finish**.

In the activity properties dialog box, several tabs along the left side provide access to general and specific settings for the activity. Although the number of available tabs for object properties differs from activity to activity, all activities will have a **General** tab, a **Details** tab, and a **Run Behavior** tab. Some activities may have additional tabs.

General Tab

This tab contains the **Name** and **Description** properties for the activity. By default, the **Name** of the activity is the same as its activity type, and the **Description** is blank. You can modify these properties to create more descriptive names or provide detailed descriptions of the actions of the activity.

Details Tab

This tab contains details that are specific to the activity.

All activities in this integration pack have the **Connection Name** property at the top of the **Details** tab. This property is used to specify the connection to the BMC BladeLogic application server.

To configure the Connection Name property:

Click the ellipsis **(…)** button next to the **Name** field, and then select the applicable connection name. Connections displayed in the list have been previously configured as described in Configuring Connections for the Kelverion Integration Pack for BMC BladeLogic Server Automation.

Run Behavior Tab

This tab contains the properties that determine how the activity handles multi-value published data and what notifications will be sent if the activity fails or runs for an excessive period of time.

Multi-Value Published Data Behavior

The Get activities retrieve information from another activity or outside source, and can return one or more values in the published data. For example, when you use the Get Collection Member activity, the data output from that activity might be a list of computers that belong to the specified collection.

By default, the data from the Get activity will be passed on as multiple individual outputs. This invokes the next activity as many times as there are items in the output. Alternatively, you can provide a single output for the activity by enabling the **Flatten** option. When you enable this option, you also choose a formatting option:

- **Separate with line breaks**. Each item is on a new line. This format is useful for creating human-readable text files for the output.
- **Separate with _** . Each item is separated by one or more characters of your choice.
- **Use CSV format**. All items are in CSV (comma-separated value) format. This format is useful for importing data into spreadsheets or other applications.

The activity will produce a new set of data every time it runs. The **Flatten** feature does not flatten data across multiple instances of the same activity.

Event Notifications

Some activities are expected to take a limited amount of time to complete. If they do not complete within that time they may be stalled or there may be another issue preventing them from completing. You can define the number of seconds to wait for completion of the action. After this period a platform event will be sent and the issue will be reported. You can also choose whether to generate a platform event if the activity returns a failure.

To be notified when the activity takes longer than a specified time to run or fails to run:

1. In the **Event Notifications** box, enter the **number of seconds** of run time before a notification is generated.
2. Select **Report if activity fails to run** to generate run failure notifications.

For more information about Orchestrator events, see the “Event Notifications " topics in the Runbook Properties (https://technet.microsoft.com/en-us/library/hh489610.aspx#Event Notifications).

Published Data

Returned data is the foundation of a working runbook. It is the data produced as a result of the actions of an activity. This data is published to an internal data bus that is unique for each runbook. Subsequent activities in the runbook can subscribe to this data and use it in their configuration. Link conditions also use this information to add decision-making capabilities to runbooks.

An activity can only subscribe to data from the activities that are linked before it in the runbook. You can use published data to automatically populate the property values needed by activities.

To use published data:

1. Right-click the property value box, click **Subscribe**, and then click **Published Data**.
2. Click the **Activity** drop-down box and select the activity from which you want to obtain the data.
3. To view additional data elements common to all activities, select **Show Common Published Data**.
4. Click the published data element that you want to use, and then click **OK**.

For a list of the data elements published by each activity, see the Published Data tables in the activity topic. For information about the common published data items, see the Published Data (http://technet.microsoft.com/en-us/library/hh403821.aspx).

Create BLPackage from Audit Activity

The Create BLPackage from Audit activity is used in a runbook to create a BLPackage based on the information collected from an Audit job. The BLPackage can then be used to create a Deploy Job in order to remediate any discrepancies detected by the audit.

Required Properties:

You must configure the following properties:

Package Name	Name of the BLPackage to be created.
Package Group ID	ID of a group that should contain the package. Takes precedence over Package Group Name.
Package Group Name	Fully qualified name of the group that should contain the package. Ignored when Package Group ID is specified.
Soft Linked	Indicates whether the package should be created soft linked to the depot object resources.
Collect File ACL	Indicates whether file ACLs should be packaged.
Collect File Attributes	Indicates whether file attributes should be packaged.
Copy File Contents	Indicates whether file contents should be packaged.
Collect Registry ACL	Indicates whether registry ACLs should be packaged.
Audit Job Result Key	Handle to the Audit Job Result used to create the package. Takes precedence over Audit Job Key and Audit Job Name.
Audit Job Key	Handle to the Audit Job used to create the package. Package will be created based on the Job Result of the last Job Run for this Audit Job. Ignored when Audit Job Result Key is specified. Takes precedence over Audit Job Name.
Audit Job Group Name	Fully qualified name of the group where the Audit Job used to create the package is located. BLPackage will be created based on the Job Result of the last Job Run for this Audit Job. Ignored when Audit Job Result Key or Audit Job Key is specified.
Audit Job Name	Name of the Audit Job used to create the package. BLPackage will be created based on the Job Result of the last Job Run for this Audit Job. Ignored when Audit Job Result Key or Audit Job Key is specified.
Template Key	Handle to the template packaged in the audit. Takes precedence over Template Name.
Template Group Name	Fully qualified group where the template packaged in the audit is located. Ignored when Template Key is specified.
Template Name	Name of the template packaged in the audit. Ignored when Template Key is specified.

Optional Properties

You can use the following optional properties to further control the behavior of the activity:

Target Component	Name of the target component. Audits can have several templates specified in them. For each template, the audit can have one master and several targets. The audit target will be picked from an audit based on the specified template, with the matched target component name. If a target component name is not provided, this activity fails if the audit result has multiple targets for the same template.
Match File	NSH path of the matching software file. If there is software in the audit result needed to match to depot software, matching will be attempted based on the mappings in the match file. If the file does not exist, or path was not provided, the activity will try matching to default depot software. If the activity cannot find default depot software, the software is skipped.

Published Data

In addition to publishing all required properties the activity also publishes the following items:

Package Key	Handle to the newly created package which can be used as input to operations which require a package handle

Create Deploy Job Activity

The Create Deploy Job activity is used in a runbook to create a basic deploy job which can be used to deploy a BLPackage or a Software Package to server(s) managed by BMC BladeLogic.

The activity can create the following deploy job types:

- *BLPackage Deploy Job*
- *Software Deploy Job*

Once the Job Type is selected the activity displays input properties specific for that job type.

Required Properties:

You must configure the following properties:

Job Type	Specifies the type of deploy job to be created.
Job Name	Name assigned to this Deploy Job.
Job Group ID	ID of the job group where job is created. Takes precedence over Job Group Name.
Job Group Name	Fully qualified name of the group where job is created. Ignored when Job Group ID is specified.
Package Key	Handle associated with the BLPackage for this Deploy Job. Takes precedence over Package Name.
Package Group Name	Fully qualified name of the group where the BLPackage for this Deploy Job is located. Ignored when Package Key is specified.
Package Name	Fully qualified name of the BLPackage for this Deploy Job. Ignored when Package Key is specified.
BladeLogic Server Name	Server where you want to run this job.
Enable Simulate	True indicates the job should include a Simulate phase, during which a dry run is performed without deploying the package.
Enable Commit	True indicates the job should include a Commit phase, during which the package is applied to the target server.
Staged Indirect	True indicates the package is delivered to a server functioning as a repeater. During the Commit phase, the package is applied to the target server.

Optional Properties

You can use the following optional properties to further control the behavior of the activity:

Job Type	Specifies the type of deploy job to be created.
Job Name	Name assigned to this Deploy Job.

Job Group ID	ID of the job group where job is created. Takes precedence over Job Group Name.
Job Group Name	Fully qualified name of the group where job is created. Ignored when Job Group ID is specified.
Software Object Key	Handle associated with the depot software object for this Deploy Job. Takes precedence over Software Object Group Name.
Software Object Group Name	Fully qualified name of the group where the software object for this Deploy Job is located. Ignored when Software Object Key is specified.
Software Object Name	Name of the Software Object for this Deploy Job.
Software Model Type	Type of software being deployed. • AIX_PATCH_INSTALLABLE: AIX patch • AIX_PACKAGE_INSTALLABLE: AIX package • HP_PRODUCT_INSTALLABLE: HP-UX product • HP_BUNDLE_INSTALLABLE: HP-UX bundle • HP_PATCH_INSTALLABLE: HP-UX patch • RPM_INSTALLABLE: Linux RPM • SOLARIS_PATCH_INSTALLABLE: Solaris patch • SOLARIS_PACKAGE_INSTALLABLE: Solaris package • HOTFIX_WINDOWS_INSTALLABLE: Windows Hot Fix • SERVICEPACK_WINDOWS_INSTALLABLE: Windows Service Pack • MSI_WINDOWS_INSTALLABLE: Windows MSI Package • INSTALLSHIELD_WINDOWS_INSTALLABLE: Windows InstallShield Package
BladeLogic Server Name	Server where you want to run this job.
Enable Simulate	True indicates the job should include a Simulate phase, during which a dry run is performed without deploying the package.
Enable Commit	True indicates the job should include a Commit phase, during which the package is applied to the target server.
Staged Indirect	True indicates the package is delivered to a server functioning as a repeater. During the Commit phase, the package is applied to the target server.

Published Data

In addition to publishing all required properties the activity also publishes the following items:

Job Key	Handle to the newly created deploy job.

Run Job Activity

The Run Job activity is used in a runbook to run one of the supported BladeLogic jobs.

Required Properties:

You must configure the following properties:

Job Type	Specifies the type of job the activity will be running. Options include: • ACL Push Job • Audit Job • Batch Job • BLPackage Job • Compliance Job • Component Discovery Job • File Deploy Job • Software Deploy Job • NSH Script Job • Snapshot Job • Virtual Infrastructure Discovery Job
Wait for Job to Complete	Specifies whether the activity waits for the configured job to complete running.
Job Key	Handle to the Job to be executed. Takes precedence over Job Name.
Job Group Name	Fully qualified group name of the Job to be executed. Ignored when Job Key is specified.
Job Name	Name of the Job to be executed. Ignored when Job Key is specified.

Published Data

In addition to publishing all required properties the activity also publishes the following items:

Job Run Key	Handle to the Job Run for the executed job.

Get Job Running Status

The Get Job Running Status activity is used in to determine if a specific job is in running state.

Required Properties:

You must configure the following properties:

Job Run Key	Specifies the job run key for the job whose running status is to be determined.

Published Data

In addition to publishing all required properties the activity also publishes the following items:

Job Is Running	Indicates whether the configured job is in running state or not.

Export Audit Results Activity

The Export Audit Results activity is used in a runbook to export results of an Audit Job Run to a file, in HTML or CSV format.

Required Properties:

You must configure the following properties:

Export Type	Specifies export file type.
Export Folder	Path of the local folder where the export file will be created.
Export File Name	Name of the export file to be created.
Template Group Name	Fully qualified component template group containing the template that is the basis of the Audit Job.
Template Name	Name of the template that is the basis of the Audit Job.
Job Group Name	Fully qualified job group where the Audit Job is located.
Job Name	Name of the Audit Job whose run results are to be exported.
Job Run ID	ID of the Audit Job Run with results to export. Takes precedence over Job Run Key.
Job Run Key	Key of the Audit Job Run with results to export. Ignored when Job Run ID is specified.

Published Data

This activity publishes all required properties.

Set Server Property Activity

The Set Server Property activity is used in a runbook for setting a property value on a BMC BladeLogic server.

Required Properties:

You must configure the following properties:

BladeLogic Server Name	Name of the server.
BladeLogic Property Name	Name of the property to be set. The property must be defined in the Server property class and it must be editable.
BladeLogic Property Value	The value to be set.
Encrypt Value	Specifies whether property value should first be encrypted before being set.

Published Data

This activity publishes all required properties.

Run BLCLI Command Activity

The Run BLCLI Command activity is used to run a BLCLI command. This activity requires the **BLCLI Command Definition Folder** to be specified in the activity configuration.

Command definition information is used by this activity to display lists of available namespaces, commands, and arguments for a specific BLCLI version. For more details see BLCLI Command Definition.

You must be familiar with capabilities, options, and configuration of specific BLCLI commands to use this activity. For more details consult the BLCLI help.

Required Properties:

You must configure the following properties:

Command Namespace	Specifies the namespace containing the BLCLI command to be executed.
Command Name	Specifies the name of the BLCLI command to be executed. When selecting a command which requires argument inputs, the activity will display additional input properties corresponding to the command arguments.
Attachment Type	Specifies the type of attachment associated with the command. Some commands can have input attachments when binary data is required as input. In such cases an input file must be specified. For example, the bulk commands in the Server namespace. Some commands can have output attachments when the command returns binary data. In such cases an output file path must be specified, where the binary data is to be saved. For example, the export commands from InportExport or Utility namespaces.
Number of Input Attachments	Specifies the number of attachments for the command. This property is only available when **Attachment Type** is: • Input Attachment • Input and Output Attachment Based on the value specified for this property, the IP will display additional input properties for specifying input attachment file path(s) and attachment argument names.
Input File Path	Only available only when the **Number of Input Attachments** is greater than one and the **Attachment Type** is: • Input Attachment • Input and Output Attachment
Input File Path X	Specifies a file to be provided to the command as input binary data. Only available only when the **Number of Input Attachments** is greater than one and the **Attachment Type** is: • Input Attachment • Input and Output Attachment

	The attachment file will be associated with the command argument specified by the corresponding Argument Name X. For example, the **createProvisionJob** BLCLI command in **ProvisionJob** namespace takes two input attachments, specified by the **inputFile** and **targetFile** arguments, respectively. When configuring this command in the IP activity, each input attachment must be associated with its corresponding BLCLI command argument. So, for the input file attachment you must specify: Input File Path 1: C:\MyInputFile.txt Argument Name 1: **inputFile** And for the target file attachment: Input File Path 2: C:\MyTargetFile.csv Argument Name 2: **targetFile**
Argument Name X	Specifies the command argument name that is to be associated with the attachment file specified by the corresponding Input File Path X. Only available only when the **Number of Input Attachments** is greater than one and the **Attachment Type** is: • Input Attachment • Input and Output Attachment
Output Folder Name	Specifies a folder where binary data output from a command is to be saved. This property is visible only when **Attachment Type** is *Output Attachment*.
Output File Name	Specifies the name of the file where binary data output from a command is to be saved. This property is visible only when **Attachment Type** is *Output Attachment*.

Published Data

In addition to publishing all required properties the activity also publishes the following items:

Command Result	Result returned from BLCLI command.

Run Custom BLCLI Command Activity

The Run Custom BLCLI Command activity is used to run a BLCLI command. This activity should be used in cases when the BLCLI command is not available through the Run BLCLI Command activity.

This activity requires additional configuration but offers greater flexibility for running a BLCLI command, compared to the Run BLCLI Command Activity. You must be familiar with capabilities, options and configuration of specific BLCLI commands in order to use this activity. For more details consult the BLCLI help.

Use the Number of Arguments property to specify how many arguments the target BLCLI command has. Based on the specified argument count, the activity will display individual argument fields for specifying argument values. The activity supports up to a maximum of 100 arguments. Argument values in the activity must be entered in the order required by the BLCLI command. For more details consult the BLCLI help for the specific command.

Required Properties:

You must configure the following properties:

Command Namespace	Specifies the namespace containing the BLCLI command to be executed.
Command Name	Specifies the name of the BLCLI command to be executed.
Number of Arguments	Specifies the number of arguments for the configured BLCLI command.
Argument 01	First command argument
Argument 02	Second command argument
Argument 03	Third command argument
…	…
Argument 100	100th command argument
Attachment Type	Specifies the type of attachment associated with the command. Some commands can have input attachments when binary data is required as input. In such cases an input file must be specified. For example, the bulk commands in the Server namespace. Some commands can have output attachments when the command returns binary data. In such cases an output file path must be specified, where the binary data is to be saved. For example, the export commands from InportExport or Utility namespaces. namespaces.
Number of Input Attachments	Specifies the number of attachments for the command. This property is only available when Attachment Type is: • Input Attachment • Input and Output Attachment Based on the value specified for this property, the IP will display additional input properties for specifying input attachment file path(s) and attachment argument names.

	For example, when this property is set to 3, the activity will display 6 additional input properties: Input File Path 1,2,3 Argument Name 1,2,3
Input File Path	Specifies a file to be provided to the command as input binary data. This property is available only when Attachment Type is: • Input Attachment • Input and Output Attachment **This property is available only when Number of Input Attachments is set to 1. In this case the attachment argument name does not need to be specified.**
Input File Path X	Specifies a file to be provided to the command as input binary data. This property is available only when Attachment Type is: • Input Attachment • Input and Output Attachment **This property is available only when Number of Input Attachments is set to 2 or more.** The attachment file will be associated with the command argument specified by the corresponding Argument Name X. For example, the **createProvisionJob** BLCLI command in **ProvisionJob** namespace takes two input attachments, specified by the **inputFile** and **targetFile** arguments, respectively. When configuring this command in the IP activity, each input attachment must be associated with its corresponding BLCLI command argument. So, for the input file attachment you must specify: Input File Path 1: C:\MyInputFile.txt Argument Name 1: **inputFile** And for the target file attachment: Input File Path 2: C:\MyTargetFile.csv Argument Name 2: **targetFile**
Argument Name X	Specifies the command argument name that is to be associated with the attachment file specified by the corresponding Input File Path X. This property is available only when Attachment Type is: • Input Attachment • Input and Output Attachment **This property is available only when Number of Input Attachments is set to 2 or more.**
Output Folder Name	Specifies a folder where binary data output from a command is to be saved. This property is visible only when Attachment Type selection is Output Attachment.

Output File Name	Specifies the name of the file where binary data output from a command is to be saved. This property is visible only when Attachment Type selection is Output Attachment.

Published Data

In addition to publishing all required properties the activity also publishes the following items:

Command Result	Result(s) returned from BLCLI command. When the BLCLI command returns a list of results the activity will publish multiple values of this field.