



Hewlett Packard
Enterprise

Operations Orchestration

Software Version: 10.70

Windows and Linux Operating Systems

OO Shell for Authoring (OOSHA) User Guide

Document Release Date: November 2016

Software Release Date: November 2016

Legal Notices

Warranty

The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from Hewlett Packard Enterprise required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

© 2005-2016 Hewlett Packard Enterprise Development LP

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Documentation Updates

To check for recent updates or to verify that you are using the most recent edition of a document, go to: <https://softwaresupport.hpe.com/>.

This site requires that you register for an HP Passport and to sign in. To register for an HP Passport ID, click **Register** on the HPE Software Support site or click **Create an Account** on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HPE sales representative for details.

Contents

Introduction	4
Installing OOSHA	4
Running HPE OO Authoring Commands from the Command Line	5
Creating Content Packs via OOSHA	9

Introduction

The Operations Orchestration Shell Authoring (OOSHA) utility lets you run HPE OO commands from the command line, instead of from the Studio 10.x interface. This enables you to automate some of your HPE OO authoring processes.

Important: In order to create and write to the oosha log file (`<oosha_installation_folder>/logs/oosha.log`) in the installation folder, you must run OOSHA with write permissions.

Installing OOSHA

There are two ways to install OOSHA:

- As a standalone package
- Within an HPE OO installation (as of version 10.60)

Install a Standalone OOSHA

1. Download the **oosha.zip** file.
2. Unzip the **oosha.zip** file to the desired location.
3. To start OOSHA, navigate to the **bin** folder, and run the **oosha.bat** executable file for the Windows operating system or the **oosha.sh** executable file for the Linux operating system.

Install OOSHA within HPE OO

1. OOSHA is installed and uninstalled with HPE OO Studio.
2. To start OOSHA, navigate to `<oo_installation_folder>/studio/tools`, and run the **oosha.bat** executable file for the Windows operating system or the **oosha.sh** executable file for the Linux operating system.

- `exit`, `quit` - exits OOSHA
 - `cls`, `clear` - clears the console
 - `system properties` - displays the OOSHA system properties
 - `date` - displays the current date
 - `version` - displays the current version of OOSHA
3. You can run commands from a normal prompt/shell using the OOSHA utility by passing arguments to the `oosha.bat` or `oosha.sh` files respectively.

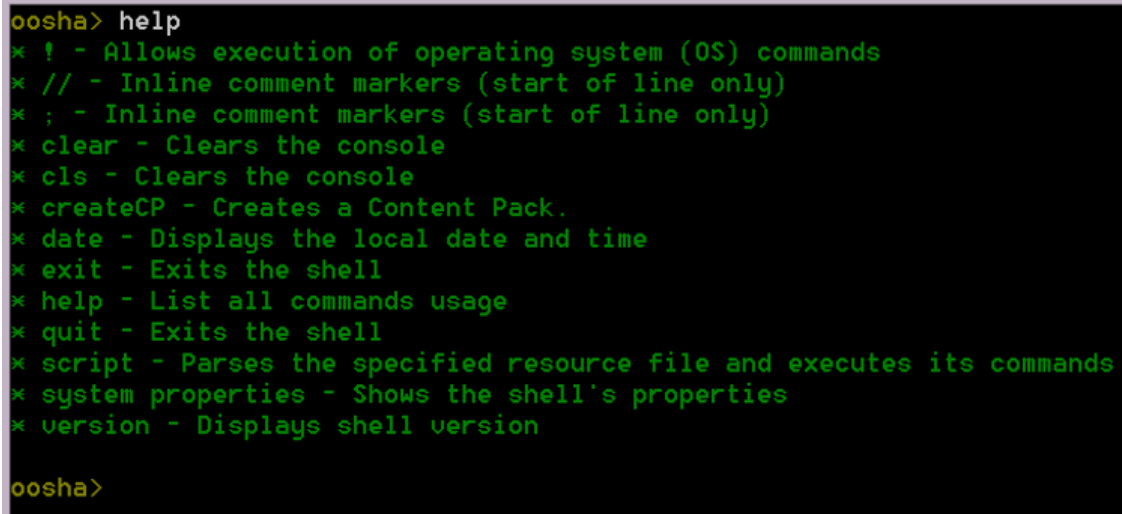
Examples:

```
oosha.bat createCP --projectFolder C:\Users\User1\.oo\Workspace\project1 --
version 7

./oosha.sh createCP --projectFolder '/home/user1/.oo/Workspace/my project' -
-destinationFolder /tmp/cps --description 'test content pack' --publisher
'the publisher' --version 1.9
```

Display Help in OOSHA

- To display a list and description of all available commands in the OOSH utility, type: `help`

A screenshot of a terminal window showing the output of the 'help' command in the OOSHA shell. The prompt is 'oosha>' and the output lists various commands and their functions in green text on a black background. The commands listed are: ! (allows execution of OS commands), // (inline comment markers), ; (inline comment markers), clear (clears console), cls (clears console), createCP (creates a content pack), date (displays local date and time), exit (exits shell), help (list all commands usage), quit (exits shell), script (parses resource file and executes commands), system properties (shows shell's properties), and version (displays shell version). The prompt 'oosha>' is visible at the bottom of the screenshot.

```
oosha> help
* ! - Allows execution of operating system (OS) commands
* // - Inline comment markers (start of line only)
* ; - Inline comment markers (start of line only)
* clear - Clears the console
* cls - Clears the console
* createCP - Creates a Content Pack.
* date - Displays the local date and time
* exit - Exits the shell
* help - List all commands usage
* quit - Exits the shell
* script - Parses the specified resource file and executes its commands
* system properties - Shows the shell's properties
* version - Displays shell version

oosha>
```

- To display help about a particular command, type: `help <command>`

For example: `help createCP`

```

oosha> help createCP
Keyword:                createCP
Description:            Creates a Content Pack.
Keyword:                projectFolder
Help:                   Absolute path to the project folder.
Mandatory:              true
Default if specified:  ''
Default if unspecified: ''

Keyword:                destinationFolder
Help:                   Absolute path to folder where the Content Pack will be created (if doesn't exist, will be created).
Mandatory:              false
Default if specified:  ''
Default if unspecified: ''

Keyword:                version
Help:                   Content Pack version. Default: the version from <projectName>/contentpack.properties.
Mandatory:              false
Default if specified:  ''
Default if unspecified: ''

Keyword:                publisher
Help:                   Content Pack publisher. Default: the publisher from <projectName>/contentpack.properties.
Mandatory:              false
Default if specified:  ''
Default if unspecified: ''

Keyword:                description
Help:                   Content Pack description. Default: the description from <projectName>/contentpack.properties.
Mandatory:              false
Default if specified:  ''
Default if unspecified: ''

Keyword:                includeSystemAccountPasswords
Help:                   Include system account passwords (obfuscated). Default: false. For security reasons, it is not recommended to include passwords in Central.
Mandatory:              false
Default if specified:  ''
Default if unspecified: ''

```

- To display a list of arguments for a valid command that has at least one argument, type: `<command>`

For example: createCP

```

oosha> createCP
You should specify option (--projectFolder, --destinationFolder, --version, --publisher, --description, --includeSystemAccountPasswords, --incrementVersion) for this command
oosha>

```

Navigate the command history

Use the **Up** arrow key or **Down** arrow key on your keyboard to navigate inside the command history for OOSHA.

- The **Up** arrow key history navigates one item back in the history of commands.
- The **Down** arrow key navigates one item forward in the history of commands.

Troubleshoot OOSHA

The OOSHA application logs are stored inside the `<oosha_installation_folder>/logs` folder.

By default, under the `logs` folder, there are two log files: `oosha.log` and `maven.log`.

Change the OOSHA log location (oosha.log)

1. Locate the `log4j.properties` file at `<oosha_installation_folder>/conf/log4j.properties`.
2. Locate the following line inside the `log4j.properties` file:

```
log4j.appender.FILE=org.apache.log4j.RollingFileAppender
```

3. Below that line, add the following line:

```
log4j.appender.FILE.File=<absolute path of the oosha.log file>
```

For example: `log4j.appender.FILE.File=C:/logs/oosha.log`

4. Save the **log4.properties** file and start the OOSHA shell.

Change the Maven log location (maven.log)

1. Locate the **oosha.properties** file at `<oosha_installation_folder>/conf/oosha.properties`.
2. Edit the **oosha.properties** file by setting the **oosha.logs.location.maven** property with the value of the absolute path of the **maven.log** file

For example:

```
oosha.logs.location.maven=C:/logs/maven.log
```

3. Save the **oosha.properties** file and start the OOSHA shell.

Configure the OOSHA encryption

Note: A content pack may contain either CloudSlang content or content that was developed from a Studio project. This section only applies to content that developed from a Studio project.

By default, OOSHA uses the same encryption options as a default Studio installation. To configure OOSHA to use the encryption of another Studio installation, set the value of the **oosha.encryptor.dir** property inside the **oosha.properties** file to point to the desired encryption folder.

1. Locate the **oosha.properties** file `<oosha_installation_folder>/conf/oosha.properties`.
2. Edit the **oosha.properties** file by setting the **oosha.encryptor.dir** property with the value of the absolute path of the Studio installation security folder.

For example:

```
oosha.encryptor.dir=<oo_installation_folder>/studio/var/security
```

3. Save the **oosha.properties** file and start the OOSHA shell.

Creating Content Packs via OOSHA

Using the `createCP` command, you can create a content pack from a project developed in Studio or in CloudSlang.

While a content pack is being created via OOSHA, the project undergoes a series of structural and consistency validation checks.

Note: For information on how to create a new project, see [Create a new project in Studio](#) and [Create a new project from a template](#).

Create a content pack

The `createCP` command packages a project into a content pack. The content pack will contain all the objects in the project.

Note: The invalid flows will also be included.

For a complete list of all the options you can use with the `createCP` command, see the [Reference](#) section or type `help createCP` in the command line.

1. Run the **oosha.bat** executable file for the Windows operating system or the **oosha.sh** executable file for the Linux operating system.
2. In the command line, type the following command, replacing the highlighted values with your own values:

```
createCP --projectFolder <projectFolder> --destinationFolder  
<CPDestinationFolder> --version <CPversion> --publisher <publisher> --  
description <description> --includeSystemAccountPasswords  
<shouldIncludeSAPasswords> --overrideDestination <shouldOverrideDestination> --  
type <contentType>
```

3. Press the **Enter** key.

The content pack is created in the location that was defined in the `--destinationFolder` argument.

The name of the content pack is taken from the **pom.xml** file. If not defined there, it is taken from the **contentpack.properties** file. If not defined there, it is taken from the project folder.

Validate a content pack

When you run the `createCP` command from OOSHA, the structure of the project is validated in the

same way as in Studio.

The createCP command will fail if the project includes corrupt XML or CloudSlang files, or files that do not respect the HPE OO validation schema.

For example:

```
oosha> createCP --projectFolder C:\Users\User1\project_CP
Failed to execute createCP --projectFolder C:\Users\User1\project_CP --destinationFolder --version --publisher --description --includeSystemAccountPasswords false: Structure Check: --type aflValidation (Errors)
#####

Violations                                Count
-----
AFL Validation:                            1
-----
Total (Errors):                             1

1) Structure Violation: AFL Validation
-----
Path:          C:\Users\MUSCA\oo\Temp\project_CP\Content\Library\How Do I flows\fff.xml
Reason:        The operation/flow XML cannot be validated against XML schema. Details: cvc-complex-type.2.4.a: Invalid content was found starting with element 'customElement'. One of '{matchRules}' is expected.
-----

#####

. Please check logs for more information.
oosha>
```

OOSHA validates that the project contains either XML or CloudSlang (.sl) files, but not both. If it contains both, you must specify the type of the content using the --type option.

Possible values:

- **afl** – the project was developed in Studio and contains XML content
- **cs** – the project contains CloudSlang content

If this option is specified, only items of the specified type (XML or CloudSlang) will be included in the resulting content pack.

Note: In this version, OOSHA does not validate the following:

- Start steps
- Connected transitions
- Lanes
- Operation responses
- Folder and file structure
- No missing references to other flows, operations or configuration items inside this project, as well as other projects and content packs

OOSHA does not exclude invalid items (flows, operations, configuration items) from the resulting content pack.

Include hard copies inside the content pack

Note: A content pack may contain either CloudSlang content or content that was developed from a Studio project. This section only applies to content that was developed from a Studio project.

In order for OOSHA to pack operation dependencies (operations' plugins and their dependencies) for hard copy operations, you must configure the **oosha.maven.home** property inside the **oosha.properties** file by following the steps below.

1. Locate the **oosha.properties** file **<oosha_instalation_folder>/conf/oosha.properties**.
2. Edit the **oosha.properties** file by setting the **oosha.maven.home** property with the value of the absolute path of the repository containing the operation dependencies.

The value of this property is typically **<user_home>/oo/data/maven**. This is the absolute path of the Studio maven repository folder that was used to create the project, **<studio_workspace>/oo/data/maven**.

If this property is not specified, the default maven home is **<user_home>/m2/repository**.

3. Save the **oosha.properties** file and start the OOSHA shell.

Include CloudSlang operations inside the content pack

In order for OOSHA to correctly pack CloudSlang operations, it might need to include their Java dependencies transitively. All Java dependencies are packaged as Maven artifacts.

For collecting all necessary artifacts inside the content pack, you must configure some properties inside the **oosha.properties** file by following the steps below.

1. Locate the **oosha.properties** file **<oosha_instalation_folder>/conf/oosha.properties**.
2. Edit the **oosha.properties** file by setting the the following properties:
 - Set the **oosha.maven.home** property with the value of the absolute path of the local repository that OOSHA will use to get dependencies from. If a dependency cannot be found in this repository, OOSHA will try to download it from a remote one.

If this property is not specified, the default maven home is **<user_home>/m2/repository**

- Set the **oosha.maven.remote.repository** property with the value of the URL of the remote repository that OOSHA will use for dependency download.

If this property is not specified, the default remote repository is Maven Central.

- Set the **oosha.maven.settings** property with the value of the absolute path of the Maven settings file to use for configuration.

If no Maven settings file is provided, OOSHA will use the configurations provided using the two properties above. If a file is provided, it will override any configuration specified in the properties above.

3. Save the **oosha.properties** file and start the OOSHA shell.

Note: OOSHA relies on the Maven Dependency plugin and the Maven Install plugin for collecting dependencies. If these do not exist in your local repository, OOSHA will try to download them from the remote one.

Configure System Accounts

Note: A content pack may contain either CloudSlang content or content that was developed from a Studio project. This section only applies to content that developed from a Studio project.

In some cases, you may want to package a content pack from a project that contains system accounts without including the system account passwords.

For example, when you want to run the `createCP` command using a false value for the `--includeSystemAccountPasswords` argument or when the value for the `--includeSystemAccountPasswords` argument is not provided.

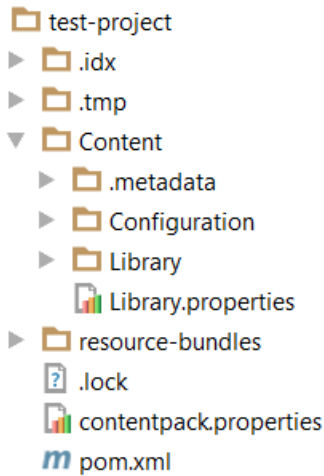
In such a case, you must first configure the **oosha.encryptor.dir** property inside the **oosha.properties** file by following the steps below.

1. Locate the **oosha.properties** file `<oosha_installation_folder>/conf/oosha.properties`.
2. Edit the **oosha.properties** file by setting the **oosha.encryptor.dir** property with the value of the absolute path of the Studio installation security folder `<oo_installation_folder>/studio/var/security`.
3. Save the **oosha.properties** file and start the OOSHA shell.

Create a new project in Studio

The complete process of creating a new project using Studio is documented in the *HPE OO Studio Authoring Guide*.

The project generated by Studio will have the following folder structure:



In the example above, the name of the project is **test-project**.

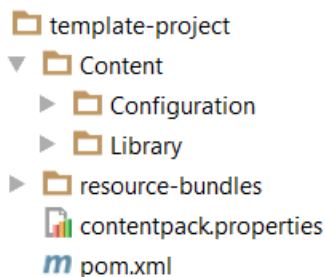
To place CloudSlang or XML files inside this project, you have to create a new subfolder under **Library** and place them inside. Then, you can build the content pack using OOSHA.

Note: Do not place the files directly under the **Library** folder, as this will not work.

Create a new project from a predefined template

To create a new project without Studio:

1. Download **template-project.zip**.
2. Unzip **template-project.zip** to the desired location.
3. Find the directory template-project with the following structure:



4. Rename the **template-project** directory to the name you want to provide for the new project.
 - o For a CloudSlang project, the project naming convention is: **cs- [<vendor>]- <product|technology>**

For example: cs-vmware-vcenter

- For a XML project, the naming convention is: **[<vendor>]-<product|technology>**
 For example: vmware-vcenter
- 5. Locate the **pom.xml** file. Open it, and replace the value inside the **artifactId** tag with the name of the project.
 For example: <artifactId>cs-vmware-vcenter </artifactId>)
- 6. Locate the **contentpack.properties** file. Open it, and provide values for the properties:
 - **content.pack.name** - It is recommended to specify a value for this property. The value should be the name of the project.
 - **content.pack.uuid** - This value is mandatory. You must provide a valid uuid value before running the **createCP** command.
 - **content.pack.version** - It is recommended to specify a value for this property. The value must be the same as the value from the version tag inside the **pom.xml** file. If you want to update the version of the project, you must update it inside both files.
 - **content.pack.publisher**

Reference Material

Available Arguments for the createCP Command

Argument	Description	Default Value/Behavior	Required
--projectFolder	The location of the project to be packaged. This must be the absolute path of the folder that represents a project in the Studio format.		Yes
--destinationFolder	The path to a local folder where the project will be packaged into a content pack. The destination folder and all parent folders up to the destination folder will be created, if they do not exist on the file system.	<ul style="list-style-type: none"> • The default location is where the content pack was created the previous time. • The first time the content pack is created, the default path is: 	No

Argument	Description	Default Value/Behavior	Required
		C:\Users\ <user_name>\.oolcontent packs<="" td=""> <td></td> </user_name>\.oolcontent>	
--version	Lets you specify the version of the content pack.	1.0.0 If the version is not specified, the default version number is taken from the project, with the “-SNAPSHOT” suffix removed. For example, if the project version is 1.7.0-SNAPSHOT , the content pack version will be 1.7.0 .	No
--incrementVersion	The content pack version is taken from the project version with the “-SNAPSHOT” suffix removed, and the project version is increased by 0.0.1. For example, assuming that you have not specified a version number with --version: If the project version is 1.7.0-SNAPSHOT , the content pack version will be 1.7.0 and the project version will be moved up to 1.7.1-SNAPSHOT .	False The default is that the project version is not incremented.	No
--publisher	Lets you specify the publisher of the content pack.	"" (empty)	No
--description	Lets you specify a description of the content pack.	If this is not specified, the description is taken from the contentpack.properties file.	No
--includeSystemAccount	Lets you specify whether	False	No

Argument	Description	Default Value/Behavior	Required
Passwords	<p>the content pack should include system account passwords.</p> <p>See "Exporting a Content Pack" in the <i>HPE OO Studio Authoring Guide</i>.</p> <p>For security reasons, it is recommended to configure the system account passwords in Central.</p>		
-- overrideDestination	<p>Lets you specify whether to override an existing file with the same name as the content pack in the destination folder.</p> <p>If the destination file that the content pack represents exists on the file system, and <code>overrideDestination</code> is set to false, the content pack will not be created.</p> <p>If the destination file that the content pack represents exists on the file system, and <code>overrideDestination</code> is set to true, the content pack will override the original destination file.</p>	False	No
--type	<p>Specify the type of content from the project. Possible values: <code>afl</code>, <code>cs</code></p> <p><code>afl</code> means that the project contains XML files with flows and operations.</p> <p><code>cs</code> means that the project contains CloudSlang files.</p>	<p>The type of the content from the project will be auto-detected.</p> <p>If the project contains both XML and CloudSlang files, OOSHA will not manage to detect the type, and will fail to build the content pack.</p> <p>The solution for this</p>	No

Argument	Description	Default Value/Behavior	Required
	Only files of the specified type will be included in the resulted content pack.	situation is to specify the type of the content. If the project is empty, the detected type will be af1.	

