Documentation

Image Compiler

1.0.1.0

Orchestra

Table of Contents

Part I - Image Compiler Documentation

1 Image Compiler Documentation

1.1 Introduction

The image compiler is a command line tool that helps to improve the deployment process of Orchestra packaged applications. A packaged applications is an orchestra container that is enriched with additional process scenarios that form together an application. The image compiler allows the creation of packaged applications.

The following artifacts are needed when creating Orchestra packaged applications (Please see the image-compile command):
1. Process scenario file(s) (psc, psx)
2. Optional - A standard Orchestra Image file, the tool generates only configuration files if not specify input Image
3. Optional - Container template file(s), the tool uses default templates if not specify templates

Note: If there is an encrypted Process Scenario file in psx format, you need to activate an Orchestra license to enable the image compiler to decrypt the psx files, please see the licensing commands createNodeActivation, activateNode, activateProduct, updateNodeActivation and nodeInfo for more information.

The output of image compiler will help to improve the deployment task of a packaged application. For all landscape and credential entries found in the source scenarios, the image compiler generates configuration file that are extended by the additional properties that can be injected by container settings.

In additional, the tool can be used to build Orchestra image file from standard Orchestra artifact (Orchestra-cd, Juno or Standalone). Please see the buildImage command for more details.
Note
If you need the Image Compiler for Orchestra, please contact licensing@soffico.de. Please let us know in your email for which Orchestra version you need it.

1.2 Installation

Image compiler environment

Before the image compiler and other command line tools can be used, the environment of the Image Compiler has to be adjusted. This can be done in the file "<scripts>\Windows\orchestra_env.bat" (or orchestra_env.sh for Linux)

orchestra_env.bat / orchestra_env.sh

```
rem --------  adjust these values
set ORC_JAVA_HOME=@JAVA_HOME
set IMGAGE_COMPILER_HOME=@INSTALL_PATH\imagecompiler
set ORC_HOME=<path-to-your-orchestra-installation>
rem --------

set ORC_JAVA_EXEC=%ORC_JAVA_HOME%\bin\java
set ORC_CLASSPATH=%IMGAGE_COMPILER_HOME%\lib\*
set ORC_CLASSPATH=%ORC_CLASSPATH%;%ORC_HOME%\WEB-INF\lib\*
set ORC_CLASSPATH=%ORC_CLASSPATH%;%ORC_HOME%\WEB-INF\classes
```

Setting	Usage	Default	Possible values
ORC_JAVA_HOME	Overrides the default value if not empty. Defines the path to the Java folder of the Java version for orchestra.	Empty, automatically retrieves Java version from sub-directory "Java" or afterward's from Windows registry	Any Path, relative or absolute on the file system. E. g. "C:\Program Files\Java\jre11" Don't forget the double quotation marks at least when you have spaces in the path.
IMGAGE_COMPILER_HOME	Overrides the default value if not empty. Defines the folder of where	Empty	Any Path, relative or absolute on the file system. E. g. "C:\Program

	Image Compiler is located (sub-folders of this folder are dockerfile, documentation, etc.) .		Files\Orchestra_ImageCompiler\imagecompiler" Don't forget the double quotation marks at least when you have spaces in the path.
ORC_HOME	Overrides the default value if not empty. Defines the folder of where Orchestra is located (sub-folders of this folder are themes, WEB-INF, etc.) .	Empty, uses a relative path "./orchestra"	Any Path, relative or absolute on the file system. E. g. ORC_HOME=C:\Orchestra Designer\4.11.0.0\orchestra

Start scripts

The image compiler can be executed with the delivered start scripts. Depending on your operating system you have to choose one of the following ones:

```
Orchestra/Startscripts/Unix/imagecompiler.sh
Orchestra/Startscripts/Windows/imagecompiler.cmd
```

NOTE: Due to changes in the release structure in Orchestra 4.15, the Image Compiler was changed. With Image Compiler version 1.0.1.0 the backward compatibility was restored. It was tested with orchestra artifacts 4.12 and 4.15. Side effects on older orchestra versions can not be excluded completely.

1.3 Reference Manual

1.3.1 Image Compiler CLI

1.3.1.1 Command: image-compile

Description

analyze environment and credential entries from the given scenarios and generates Docker Compose file in YML format or Kubernetes ConfigMap from the templates.
In addition, generate an Orchestra Image that contains the given scenarios, the finished file will be created in the given target directory.

Required parameters

Parameter	Description
-source <directory|pscpath>	the directory of psc files or specific psc filepath where the psc file(s) should be read
-target <directory>	the directory of output file(s)

Optional parameters

Parameter	Description
-dockerTemplate <file>	the docker configuration template file Predefined template placeholders: {IMAGE_NAME} is replaced with the Image Name {AUTODEPLOYMENT_PATH} is replaced with Auto deployment path in container {SCENARIO_CONFIGURATION} is replaced with the list of scenario properties
-kubernetesTemplate <file>	the kubernetes configuration template file Predefined template placeholders: {AUTODEPLOYMENT_PATH} is replaced with Auto deployment path in container {SCENARIO_CONFIGURATION} is replaced with the list of scenario properties
-mode <mode>	emit_configuration : Only emit the configuration files (default value) compile : Compile a new image and emit the configuration files
-image <file>	the docker base image file path

-type <type>	docker : generate only docker configuration files kubernetes : generate only kubernetes configuration files all : generate both configuration types (default value)
-binaryMode <mode>	base64 : encode credential content as base64 and store it in the configuration files (default value) fileNameOnly : store only file name in configuration file : store file name and generate credential content to files
-autoDeployDir <directory>	the directory of container where configure for auto deployment to put psc file(s) (default directory : /tmp/orchestra/autodeployment/internal)
-binDir <directory>	the directory of container where binary files should be generated (default directory : /tmp/orchestra/security/binary)
-licenseFile <licensepath>	the path to the orchestra license that should be added to the container
-targetImage <name>	the name of the created target image
-pushRegistry <registry>	the registry to push image to.
-pushUsername <username>	the username to login to the registry.
-pushPassword <password>	the password to login to the registry.
-ignoreSslCertificateCheck	use this switch to ignore the SSL certificate validation.
-showStackTrace	use this switch to show stack trace on the console.

Batch mode

The batch mode is started, if the start script is launched with additional arguments.

⚠ Execution will be stopped if any required parameter is missing.

Example
Here we are executing the image compiler in batch mode.

```
./imagecomplier.sh

> image-compile -source .\psc -target .\target -dockerTemplate .
\docker_compose_template.yml -kubernetesTemplate
.\kubernetes_template.properties -image .\image_latest.tar.gz -mode
compile -type all -binaryMode fileNameOnly -licenseFile .\license.xml -
targetImage .\target\image.tar.gz
```

Interactive mode

If the image-compile is executed without any arguments, the image compiler will be started in the interactive mode. In this mode the compiler will stay open and you can interact with the compiler until you enter the command exit.

Example

Here we are executing the image compiler in interactive mode.

```
./imagecomplier.sh
> image-compile
```

PSX File

PSX are encrypted scenarios. In order to enable the Image Compiler to decrypt and read the PSX files, the Image Compiler need to activate an Orchestra license by the licensing commands.

- Command: createNodeActivation
- Command: updateNodeActivation
- Command: activateNode
- Command: activateProduct

⚠ Please ensure that the the licence that is injected into the container also contains the product license for the psx scenarios. If this restriction is not satisfied, the files will not be automatically deployed, since the license check will fail.

Output file

orchestra_docker_compose.yml

```
version: "3.7"
services:
 orchestra:
   image: {IMAGE_NAME}
   restart: always
   volumes:
    - ./data/orc/logs:/tmp/orchestra/logs
    # Uncomment the following line if you want to bind an autodeployment path
    # - ./data/orc/deploypath:{AUTODEPLOYMENT_PATH}
    - ./data/orc/longtimearchive:/tmp/longtimearchive
   environment:
     #containerMode defines if orchestra works in container mode or not.
     #     If set to true, the container mode is active
     #     If set to false, the container mode is inactive
     ORCHESTRA_CONTAINER_MODE_ACTIVE: 'true'
     #The unique name of this node inside a cell. The unique identifier is the combination of cellname and nodeid.
     #Only letters digits and '-' are allowed. The length must not exceed 10 characters.
     orchestra_settings_runtime_nodeid: ORC
     #Defines a comma separated list of tags that are associated with the current node.
     orchestra_settings_runtime_node_label: ORC
     #Name of the owning cell, has to be unique (two cells are not allowed to have the same cellname)
     orchestra_settings_runtime_cellname: ORC

     #Please use one of the following runtime database types: MSSQL,MYSQL,Oracle,POSTGRESQL,MARIADB
     orchestra_settings_database_runtime_db_typ: <dbtype>
     orchestra_settings_database_runtime_url: <url-runtime-database>
```

```
    orchestra_settings_database_runtime_user: <user of the runtime database>
    orchestra_settings_database_runtime_pwd: <password of the runtime database>
    #Please use one of the following archive database types:
MSSQL,MYSQL,Oracle,POSTGRESQL,MARIADB
    # orchestra_settings_database_archive_db_typ: <dbtype>
    # orchestra_settings_database_archive_url: <url-archive-database>
    # orchestra_settings_database_archive_user: <user of the runtime database>
    # orchestra_settings_database_archive_pwd: <password of the runtime database>
    #At this directory orchestra creates its folder-structure and saves the zip-files with
process information and messages
    # orchestra_settings_longtimearchive_file_path: /tmp/longtimearchive
    #Is LTA active or inactive? (is the normal housekeeping active?)
    orchestra_settings_longtimearchive_mode: inactive
    #Autodeployment completion mode:
    #     internal: storage of deploymentFiles in DB; no polling but only initial deployment
at container startup
    #     external: renaming of files on file level, polling after defined time interval
    orchestra_settings_runtime_autodeployment_mode: internal
    #The directory to be searched for scenario, credential and landscape files
    orchestra_settings_runtime_autodeployment_dir: {AUTODEPLOYMENT_PATH}
    #if true, also subdirectories are scanned for files to deploy/upload
    orchestra_settings_runtime_autodeployment_recursive: 'true'
    #if true, scenarios are activated after successful deployment of the files
    orchestra_settings_runtime_autodeployment_activate: 'true'
    #Orchestra will be accessible through this port for HTTP-connections. If -1 is used, the
http standard port is disabled.
    #If 0 is used, a proper port will be chosen automatically.
    orchestra_settings_ExtendedWebApp_servlet_port_http: 8090
    {SCENARIO_CONFIGURATION}


  ports:
```

```
    - "8091:8091"
    - "8443:8443"
    - "8444:8444"
    # Uncomment the following line if you want allow access to the embedded
applications (e.g. monitor) over http
    # - "8080:8080"
    # Uncomment the following line if you want allow http access for rest and soap services
    # - "8019:8019"
    # Uncomment the following line if you want allow cell connections to the orchestra
    #- "8894:8894"
```

juno_docker_compose.yml

```
version: "3.7"
services:
 orchestra:
   image: {IMAGE_NAME}
   restart: always
   volumes:
    - ./data/orc/logs:/home/juno/logs
    # Uncomment the following line if you want to bind an autodeployment path
    # - ./data/orc/deploypath:{AUTODEPLOYMENT_PATH}
    # Uncomment the following line if you want to bind a longtimearchive path
    # - ./data/orc/longtimearchive:/tmp/orchestra/longtimearchive
   environment:
     #containerMode defines if orchestra works in container mode or not.
     #     If set to true, the container mode is active
     #     If set to false, the container mode is inactive
     ORCHESTRA_CONTAINER_MODE_ACTIVE: 'true'
     #The unique name of this node inside a cell. The unique identifier is the combination
of cellname and nodeid.
```

```
    #Only letters digits and '-' are allowed. The length must not exceed 10 characters.
    orchestra_settings_runtime_nodeid: ORC
    #Defines a comma separated list of tags that are associated with the current node.
    orchestra_settings_runtime_node_label: ORC
    #Name of the owning cell, has to be unique (two cells are not allowed to have the
same cellname)
    orchestra_settings_runtime_cellname: ORC
    #Autodeployment completion mode:
    #  internal: storage of deploymentFiles in DB; no polling but only initial deployment at
container startup
    #  external: renaming of files on file level, polling after defined time interval
    orchestra_settings_runtime_autodeployment_mode: internal
    #The directory to be searched for scenario, credential and landscape files
    orchestra_settings_runtime_autodeployment_dir: {AUTODEPLOYMENT_PATH}
    #if true, also subdirectories are scanned for files to deploy/upload
    orchestra_settings_runtime_autodeployment_recursive: 'true'
    #if true, scenarios are activated after successful deployment of the files
    orchestra_settings_runtime_autodeployment_activate: 'true'
    #Orchestra will be accessible through this port for HTTP-connections. If -1 is used, the
http standard port is disabled.
    #If 0 is used, a proper port will be chosen automatically.
    orchestra_settings_ExtendedWebApp_servlet_port_http: 8090
    #Is LTA active or inactive? (is the normal housekeeping active?)
    orchestra_settings_longtimearchive_mode: inactive
    {SCENARIO_CONFIGURATION}

  ports:
    # HTTPS-Port for Orchestra monitor
    - "8091:8091"
    # HTTPS-Port for HTTP-Traffic (REST/SOAP)
    - "8443:8443"
```

```
    # HTTPS-Port with certification authentication (REST/SOAP)
    - "8444:8444"
    # Uncomment the following line if you want allow access to the embedded
applications (e.g. monitor) over http
    # - "8090:8090"
    # Uncomment the following line if you want allow http access for rest and soap services
    # - "8019:8019"
    # Uncomment the following line if you want allow cell connections to the orchestra
    #- "8894:8894"
```

- {IMAGE_NAME} will be replaced with a name of target image.
- {AUTODEPLOYMENT_PATH} will be replaced with the directory of container where configure for auto deployment .
- {SCENARIO_CONFIGURATION} will be replaced with scenario container configuration if scenario is exists.

1.3.1.2 *Command: createBaseImage*

Description

Create base image from docker file.

If -dockerFile is not used, the default dockerfile will be used.

The docker engine must be running.

Required parameters

Parameter	Description
-targetPath <directory>	the directory of output image
-targetImage <image>	the name of output image the name of output image can be input in the format **<image name>[:<tag>]**

	• **<image name>** the name of image. • **<tag>** the image tag

Optional parameters

Parameter	Description
-dockerFile <file>	the path of docker file
-architecture [arm|x86]	defines the architecture of dockerfile that is used to create base image. The default is x86
-showStackTrace	use this switch to show stack trace on the console.

Batch mode

The batch mode is started, if the start script is launched with additional arguments.

⚠ Execution will be stopped if any required parameter is missing.

Example

Here we are executing the update image in batch mode.

```
./imagecomplier.sh

> createBaseImage
-dockerFile <docker file>
-targetPath <output directory>
-targetImage <name>[:<tag>]
```

1.3.1.3 *Command: buildImage*

Description

create orchestra image from standard Orchestra artifact (Orchestra-cd, Juno or Standalone).

This feature requires a base image for the build process.

The base image can either be specified by its name or by using an image file stored on your local machine. If you do not specify the base image, the default image that will be used is

registry.access.redhat.com/ubi8/ubi:latest. Docker engine is required to pull the specified image.

The compressed file with suffix .zip, .tar and .gz is supported.

If -javaPath is not used, make sure that the base image contain supported java for orchestra.

If using base image by name, the docker engine must be running.

Required parameters

Parameter	Description
-orchestraPath <file or directory>	the compressed file or directory of the base orchestra artifact (orchestra-cd or juno-cd) that is used to create the container
-targetPath <directory>	the directory of output image
-targetImage <image>	the name of output image the name of output image can be input in the format **<image name>[:<tag>]** • **<image name>** the name of image. • **<tag>** the image tag
-baseImage <image or file>	the name of base image or local image file that has to be used. the name of base image can be input in the format **<image name>:<tag>**. The docker engine is required if use this option: • **<image name>** the name of image. • **<tag>** the image tag

Optional parameters

Parameter	Description
-javaPath <file or directory>	the compressed file or directory of java for Linux
-copyLib <files or directories>	the files or files in directories to be added to the library directory of orchestra (use | to separate multiple path). This switch can be used **multiple** times.

-copy <file or directory>|<target path>	the file or files in directory to be added to the specific directory of image. This switch can be used **multiple** times. • **<file or directory>** the file or directory to be copied • **<target path>** the target file or directory inside image
-remove <files or directories>	the relative path of files or directories of base orchestra directory to be excluded from output image (use | to separate multiple path). This switch can be used multiple times. The base orchestra directory of **orchestra-cd** is <orchestra cd artifact>/Application/orchestra The base orchestra directory of **Juno/Standalone** is the same directory of orchestra Juno/Standalone artifact
-tomcatPath <file or directory>	the compressed file or directory of tomcat for Linux (required if orchestra artifact is **orchestra-cd**)
-loggingPropertiesPath <file>	the file of external logging.properties
-showStackTrace	use this switch to show stack trace on the console.

Batch mode

The batch mode is started, if the start script is launched with additional arguments.

⚠ Execution will be stopped if any required parameter is missing.

Example

Here we are executing the build image in batch mode.

```
./imagecomplier.sh

> buildImage
-orchestraPath <orchestra or juno directory>
-javaPath <java directory>
-tomcatPath <tomcat directory>
-targetPath <output directory>
-targetImage <name>[:<tag>]
-copy <source file or directory>|<target>
-copy <source file or directory>|<target>
-copyLib <source file or directory>
-copyLib <source file or directory>|<source file or directory>
-remove <relative path of file or directory>
-remove <relative path of file or directory>|<relative path of file or
directory>
-baseImage <image or file>
-loggingPropertiesPath <logging.properties file>
```

1.3.1.4 Command: updateImage

Description

Add files or directory to an orchestra image.

Required parameters

Parameter	Description
-image <file>	the file of orchestra image
-type [JUNO|STANDALONE|ORCHESTRA]	defines the type of the orchestra-artifact
-targetImage <image>	the name of output image the name of output image can be input in the format **<image name>[:<tag>]** • **<image name>** the name of image. • **<tag>** the image tag

Optional parameters

Parameter	Description
-targetPath <directory>	the directory of output image (default is the same as directory of orchestra image

-copyLib <files or directories>	the files or files in directories to be added to the library directory of orchestra (use | to separate multiple path). This switch can be used multiple times
-copy <file or directory>|<target path>>	the file or files in directory to be added to the specific directory of image. This switch can be used **multiple** times. • **<file or directory>** the file or directory to be copied • **<target path>** the target file or directory inside image
-showStackTrace	use this switch to show stack trace on the console.

Batch mode

The batch mode is started, if the start script is launched with additional arguments.

⚠ Execution will be stopped if any required parameter is missing.

Example

Here we are executing the update image in batch mode.

```
./imagecomplier.sh

> updateImage
-image <file>
-targetPath <output directory>
-targetImage <name>[:<tag>]
-type <orchestra or juno>
-copy <source file or directory>|<target>
-copy <source file or directory>|<target>
-copyLib <source file or directory>
-copyLib <source file or directory>|<source file or directory>
```

1.3.1.5 *Command: createNodeActivation*

Description

generates license activation request file for image compiler.

Required parameters

Parameter	Description
-customerKey <customerKey>	the customer key
-nodeType <type>	the type of node, e.g. Production, Integration, Development
-nodeName <nodeName>	the name of node
-target <directory>	the directory of licensing request file

Optional parameters

Parameter	Description
-showStackTrace	use this switch to show stack trace on the console.

1.3.1.6 Command: updateNodeActivation

Description

generates updated license activation request file for image compiler.

Required parameters

Parameter	Description
-target <directory>	the directory of licensing request file

Optional parameters

Parameter	Description
-showStackTrace	use this switch to show stack trace on the console.

1.3.1.7 Command: activateNode

Description

activate image compiler license.

Required parameters

Parameter	Description

-response <file>	the license response file for image compiler

Optional parameters

Parameter	Description
-showStackTrace	use this switch to show stack trace on the console.

1.3.1.8 Command: activateProduct

Description

activates product license that needed to open PSX file.

Required parameters

Parameter	Description
-licenseFile <file>	the path of the product license file

Optional parameters

Parameter	Description
-productKey <productKey>	the product key
-showStackTrace	use this switch to show stack trace on the console.

1.3.1.9 Command: license-key

Description

prepare license key for container.

Required parameters

Parameter	Description
-licenseFile <licensepath>	the path to the orchestra license that should be added to the container file

Optional parameters

Parameter	Description
-showStackTrace	use this switch to show stack trace on the console.

1.3.1.10 Command: pushImage

Description

push image file to registry.

Required parameters

Parameter	Description
-image <file>	the image file path.
-registry <registry>	the registry to push image to.
-targetImage <name>[:<reference>]	the target image repository of registry to push image to. • **<name>** the repository of image. • **<reference>** the image tag or image digest. the default is latest.

Optional parameters

Parameter	Description
-username <username>	the username to login to the registry.
-password <password>	the password to login to the registry.
-ignoreSslCertificateCheck	use this switch to ignore the SSL certificate validation.
-showStackTrace	use this switch to show stack trace on the console.

Add public certificate to trusted store (cacerts)

if the register server uses self-signed certificate. You need to add the public certificate to trusted store (cacerts) with the keytool

To import certificates into cacerts:

- Go to the location of the JDK/JRE you're using, for example C:\Program Files\Java\jdk1.8.0_101\jre\lib\security
- Open a command prompt and type: keytool -import -alias <alias> -keystore <cacerts_file> -trustcacerts -file <certificate_filename>
- When prompted Enter keystore password:, enter "changeit". By default keystores have a password of "changeit"
- When prompted Trust this certificate? [no]:, enter "yes".
- This imports the certificate into the keystore and display the message: "Certificate was added to keystore".

1.3.1.11 Command: nodeInfo

Description

displays node information.

1.3.1.12 Command: help

Description

prints a help screen which contains a list of all available commands and their detailed description.

Example
`./imagecomplier.sh` `> help`