

Maven2 Plugin Reference Guide

Reference Guide

These are the various XML configuration elements that you can use to configure the Cargo Maven2 plugin. Make sure you also check the [use cases](#) which show how to use them.

Top level configuration elements	Description	Mandatory?	Default value
<configuration>	Definition of a Configurati on		Defaults to a standalone configuration if the container is of type local and a runtime one if it's of type remote
<container>	Definition of a Container		Defaults to a Jetty 7.x installed container if not specified
<deployer>	Definition of a Deployer		Defaults to a deployer mat ching the container's type if none is specified (installed local deployer for an installed container, remote deployer for a remote container and embedded local deployer for an embedded container)
<deployables>	A list of deployables that are going to be deployed in the container when it is started or when cargo:deploy / cargo:undeploy is called.		If the project's packaging is war, ear or ejb, the generated artifact is added automatically to the list of deployables to deploy. If you wish the generated artifact not to be added to the deployables list, just add an empty <deployer/> element.
<daemon>	Additional configuration that is used when deploying with the Cargo Daemon .		For more information, please read: Cargo Daemon .
<skip>	Set this to true to bypass cargo execution		Defaults to false

<code><wait></code>	<p>Decides if Cargo should wait after the container is started or not. If using Cargo for integration tests, set it to <code>false</code>, otherwise Cargo will start the container and show the following message: <code>Press Ctrl-C to stop the container...</code></p> <p>Important: This parameter has been deprecated and will be removed soon. If you want to do manual testing, please use the <code>cargo:run</code> MOJO.</p>		<code>false</code>
---------------------------	--	---	--------------------

i The wait parameter

Instead of tweaking your Maven project POM with the different values for the `wait` parameter, please use the `cargo:run` MOJO:

- The `cargo:start` MOJO should be used to start the container and hand over to the next Maven execution; i.e. it should ONLY be used for integration testing.
- The `cargo:run` MOJO will start the container and wait until the user decides to stop it by pressing `CTRL + C`, which implies that it should be used for manual testing.

<code><configuration></code> elements	Description	Mandatory?	Default value
<code><configfiles></code>	List of Configuration files that are to be added to a local container's configuration. Each file is specified using a <code><configfile></code> element. Cargo token replacement is applied to the files and any existing file is overwritten.		No default
<code><files></code>	List of files that are to be added to a local container's configuration. Each file is specified using a <code><file></code> element.		No default

<code><home></code>	For standalone configuration this is the location where Cargo will create the configuration and for existing configuration this is where it is located		<code>\${project.build.directory}/cargo/configurations/\${containerId}</code>
<code><implementation></code>	Full classname of a custom configuration implementation to use. In that case the custom configuration is registered with the <code><containerId></code> and <code><type></code> specified		Defaults to the Cargo-provided implementation if not specified
<u><code><properties></code></u>	<p>Values to use for various Configuration properties.</p> <p>You can also use the <code><propertiesFile></code> element to load configuration properties from a file.</p>		<p>Default configuration properties</p> <p>Note that these configuration properties can also be overridden using Java properties; for example:</p> <pre style="border: 1px dashed blue; padding: 10px;">mvn -Dcargo.s ervlet.po rt=8082 cargo:sta rt</pre> <p>In addition to this, properties can also be set using the Maven2 <code>settings.xml</code> file. See the setting configuration options via the Maven settings.xml section for details.</p>
<code><type></code>	Configuration's type . Valid values are <code>standalone</code> , <code>existing</code> and <code>runtime</code>		<code>standalone</code>

<code><container></code> elements	Description	Mandatory?	Default value
---	-------------	------------	---------------

<append>	If true, then the file specified by <output> will not be erased across different runs		false
<containerId>	Id of the container to use. Valid values can be found in the description page for each container		jetty7x
<dependencies>	List of extra dependencies or shared dependencies that will be added to the container or applications execution classpath.		No default
<home>	Location where the container is installed. If specified in conjunction with the <zipUrlInstaller> or <artifactInstaller> element, it will override the home directory defined by the installer		No default If the user has not defined any home, <zipUrlInstaller> nor <artifactInstaller> element then the plugin will automatically attempt to download the container using the URL used by its tests (see the Tested On section of each container).
<implementation>	Full classname of a custom container implementation to use. In that case, the custom container is registered with the <containerId> and <type> specified		Defaults to the Cargo-provided implementation if not specified
<log>	Path to a file where Cargo logs are saved		Logs to the Maven console if no log file is specified
<output>	Path to a file where container logs are saved		Logs to the file specified by the <log> element or to the Maven console if no such file has been specified

<systemProperties>	<p>List of <key>value</key> pairs to be passed as System properties to the container when it is started.</p> <p>You can also use the <systemPropertiesFile> element to load system properties from a file.</p>		<p>No default</p>
<timeout>	<p>The timeout after which Cargo reports an error if the container is not started or stopped</p>		<p>120000 ms (2 minutes)</p>
<type>	<p>The container's type. Valid values are installed, embedded and remote</p>		<p>Default value is installed unless the <containerId> has not been specified, in which case the default is to use the Jetty 7.x installed container</p>
<zipUrlInstaller>	<p>Defines the location of a container distribution zip that will be downloaded and installed</p>		<p>No default</p> <p>If the user has not defined any home, <zipUrlInstaller> nor <artifactInstaller> element then the plugin will automatically attempt to download the container using the URL used by its tests (see the Tested On section of each container).</p>
<artifactInstaller>	<p>Defines the location of a container Maven artifact that will be downloaded and installed</p>		<p>No default</p> <p>If the user has not defined any home, <zipUrlInstaller> nor <artifactInstaller> element then the plugin will automatically attempt to download the container using the URL used by its tests (see the Tested On section of each container).</p>

<deployer> elements	Description	Mandatory?	Default value
<implementation>	Deployer implementation class. Usage of this option is not recommended, please prefer type instead.		No default
<type>	The deployer's type .		Defaults to a deployer matching the container's type if none is specified (installed local deployer for an installed container, remote deployer for a remote container and embedded local deployer for an embedded container)

<configfile> elements	Description	Mandatory?	Default value
<file>	The configuration file or directory containing configuration files.		No default
<todir>	The target directory, relative to configuration home, where the file should be copied		If not specified, the file will be copied to the configuration's home directory
<tofile>	The target file name to use		The original file name

<file> elements	Description	Mandatory?	Default value
<file>	The file, or directory, to add		No default
<todir>	The target directory, relative to configuration home, where the file should be copied		If not specified, the file will be copied to the configuration's home directory

<tofile>	The target file name to use		The original file name
<configfile>	Indicates if Cargo token replacement should be applied (<code>true</code>) when copying. Do not use this option on a non-ascii file as it will corrupt it!		<code>false</code>
<overwrite>	If any existing file should be overwritten or not		<code>true</code>

<deployable> elements	Description	Mandatory?	Default value
<artifactId>	Maven artifact id for the module. This artifact id must match either the project's artifact id if your project generates a J2EE artifact (WAR, EAR, EJB and RAR) or it must match a specified <project>/<dependencies>/<dependency> artifact id		Defaults to the project's artifact id
<groupId>	Maven group id for the module. This group id must match either the project's group id if your project generates a J2EE artifact (WAR, EAR, EJB and RAR) or it must match a specified <project>/<dependencies>/<dependency> group id		Defaults to the project's group id
<implementation>	Deployable implementation class. Usage of this option is not recommended, please prefer <code>type</code> instead.		No default

<code><location></code>	Path location where the module can be found		Default's to the project's generated artifact location or to the specified <code><project>/<dependencies>/<dependency></code> location
<code><pingURL></code>	URL on which to ping the deployed or undeployed application (to check if deployment or undeployment is successful), that should return an HTTP OK response only after the deployment is complete. If not set, the deployed or undeployed application will not be pinged, hence the deployment considered as complete as soon as the target server's method returns successfully.		No default
<code><pingTimeout></code>	If <code><pingURL></code> is set, the number of milliseconds after which the ping fails the build if still not successful.		20000 (i.e., 20 seconds)
<code><properties></code>	User-defined properties of a deployable.		No default
<code><type></code>	Maven type for the module. This type must match either the project's packaging if your project generates a J2EE artifact (WAR, EAR, EJB and RAR) or it must match a specified <code><project>/<dependencies>/<dependency></code> type		Defaults to the project's packaging

<code><properties></code> elements	Deployable Type	Description	Mandatory?	Default value
--	-----------------	-------------	------------	---------------

<code><context></code>	WAR	The context name to use when deploying the web application.		If not specified, then the default context name is computed from the name of WAR itself (without the file extension) or <code><project>/<build>/<finalName></code>
<code><war></code>	WAR	The path of the WAR being deployed.		Default's to the project's generated artifact location
<code><ear></code>	EAR	The path of the EAR being deployed.		Default's to the project's generated artifact location
<code><name></code>	EAR	The name of EAR deployable (it can be anything, there's no special rule).		If not specified, it is computed from the EAR's file name (removing the filename extension) or <code><project>/<build>/<finalName></code>
<code><ejb></code>	EJB	The path of the EJB being deployed.		Default's to the project's generated artifact location

 About WAR contexts

Many containers have their specific files for redefining context roots (Tomcat has context.xml, JBoss has jboss-web.xml, etc.). If your WAR has such a file, the server will most probably use the context root defined in that file instead of the one you specify using the CARGO deployer.

<code><dependency></code> elements	Description	Mandatory?	Default value
<code><artifactId></code>	Maven's artifact id. This artifact id must match a specified <code><project>/<dependencies>/<dependency></code> artifact id		Defaults to the project's artifact id

<code><groupId></code>	Maven's group id. This group id must match a specified <code><project>/<dependencies>/<dependency></code> group id		Defaults to the project's group id
<code><type></code>	Maven's type. This type must match a specified <code><project>/<dependencies>/<dependency></code> type		Defaults to the project's packaging
<code><classpath></code>	Target classpath, either extra (default) or shared. Shared application classpath deployment is only available for local containers which support shared Application Classpaths .		extra (container classpath)
<code><location></code>	<p>The path of a folder or a jar file you wish to add to deployable classpath. This element can be used to explicitly add entries to the classpath. For example:</p> <pre data-bbox="521 1161 756 1591" style="border: 1px dashed blue; padding: 10px;"> <dependency> <location>src/main/resources/conf</location> </dependency> </pre>		If the groupId and artifactId match those of the project then the deployable is the artifact generated by the project. Otherwise the location is the location of the dependency in your local repository.

<code><zipUrlInstaller></code> elements	Description	Mandatory?	Default value
<code><url></code>	URL from which to download the container's ZIP or TAR.GZ file.		No default

<code><downloadDir></code>	Directory in which the <code>zipUrlInstaller</code> should download the container's ZIP or TAR.GZ file.		<code>\${java.io.tmpdir}/cargo/installs</code>
<code><extractDir></code>	Directory in which the <code>zipUrlInstaller</code> should extract the container's ZIP or TAR.GZ file.		<code>\${project.build.directory}/cargo/installs</code>
<code><proxy></code>	Proxy server settings, if required.		No default

 Automatic proxy settings

Note that CARGO will by default reuse existing Maven2 proxy configuration -so you won't need to type the proxy settings for the `zipUrlInstaller` element.

<code><proxy></code> elements (under the <code>zipUrlInstaller</code> element)	Description	Mandatory?	Default value
<code><cargo.proxy.host></code>	Proxy host name or IP address.		No default
<code><cargo.proxy.port></code>	Proxy port.		Very probably 80
<code><cargo.proxy.user></code>	User name to connect to the proxy server.		No default
<code><cargo.proxy.password></code>	Password to connect to the proxy server.		No default

<code><artifactInstaller></code> elements	Description	Mandatory?	Default value
<code><groupId></code>	Group id.		No default
<code><artifactId></code>	Artifact id.		No default
<code><version></code>	Version.		No default
<code><type></code>	Artifact type.		zip
<code><classifier></code>	Classifier.		No default

<extractDir>	Directory in which the artifactInstaller should extract the container's ZIP or TAR.GZ file.		<code>\${project.build.directory}/cargo/installs</code>
--------------	---	---	---

Daemon configuration

The Cargo Daemon is a Web-based application that uses the Cargo API to configure, start and stop containers on a remote machine. The daemon is meant to be listening 24/7, to allow users to deploy new containers and web applications at their command. For more information, please read: [Cargo Daemon](#).

Container configuration for the Daemon

For the Maven2/Maven3 plugin, the "daemonized server" is actually a [local container](#) with a [host name](#) that points to a remote machine. This implies that:

- You should not set the container type to a [remote container](#) nor add any [remote deployers](#) to the configuration; but instead define the container as a [local container](#) (with either a [standalone](#) or [existing](#) configuration)
- When you define the home paths for the container and the configuration, remember these paths are for the machine where the Daemon is running (and, preferably, use absolute paths)

When you call `cargo:daemon-start`, the Maven2/Maven3 plugin will do the following:

- If an [installer](#) is defined:
 - Download the archive locally
 - Send the archive over to the machine running the Daemon
 - Instruct the Daemon to extract the archive
- If a [standalone local configuration](#) is defined, instruct the Daemon to create it
- In all cases:
 - Send the [configuration files](#) and [deployables](#) to the machine running the Daemon
 - Instruct the Daemon to deploy them
- Finally, instruct the Daemon to start the container

<daemon> elements	Description	Mandatory?	Default value
<classpath>	A list of <classpath>my classpath</classpath> items, that will be added by the JVM launcher when starting a container.		No default
<properties>	A list of properties used to configure the Cargo Daemon .		No default

<properties> elements	Description	Mandatory?	Default value
<code><cargo.daemon.url></code>	URL to connect with the daemon.		No default
<code><cargo.daemon.handleid></code>	The handle id to register this container with.		No default
<code><cargo.daemon.autostart></code>	When set to <code>true</code> , the daemon will automatically restart the container if the daemon notices it is stopped.		<code>false</code>

Setting configuration options via the Maven settings.xml

The Cargo Maven2/Maven3 plugin also allows you to define container configuration properties using the `settings.xml` file. This way, you can for example store properties like usernames and passwords in a centralized location (as opposed to `pom.xml` files).

First, add the configuration properties you would like in your `settings.xml` file's `<servers>` section:

```
<servers>
  <server>
    <id>jonas1</id>
    <configuration>

<cargo.remote.uri>jmx://jonas1</cargo.remote
.uri>

<cargo.jonas.domain.name>jonas</cargo.jonas.
domain.name>

<cargo.remote.username>jonas</cargo.remote.u
sename>

<cargo.remote.password>jonas</cargo.remote.p
assword>
    </configuration>
  </server>
</servers>
```

Then, in the Cargo plugin's configuration, use the `cargo.server.settings` property in order to reference the configuration properties that you have previously defined. For example:

```
<plugin>
  <groupId>org.codehaus.cargo</groupId>

  <artifactId>cargo-maven2-plugin</artifactId>
  <configuration>
    [...]
    <configuration>
      <properties>

<cargo.server.settings>jonas1</cargo.server.
settings>
      </properties>
    </configuration>
  </configuration>
</plugin>
```

In this case, the plugin will internally "expand" the configuration into:

```
<plugin>
  <groupId>org.codehaus.cargo</groupId>

  <artifactId>cargo-maven2-plugin</artifactId>
  <configuration>
    [...]
    <configuration>
      <properties>

<cargo.remote.uri>jmx://jonas1</cargo.remote
.uri>

<cargo.jonas.domain.name>jonas</cargo.jonas.
domain.name>

<cargo.remote.username>jonas</cargo.remote.u
sersname>

<cargo.remote.password>jonas</cargo.remote.p
assword>
      </properties>
    </configuration>
  </configuration>
</plugin>
```

 **Careful with properties set in the POM**

Priority for property values are as follows:

- Highest priority: Properties set using environment variables
- Second priority: Properties that should be loaded using the `cargo.server.settings` option
- Third priority: Properties loaded from a file
- Lowest priority: Properties set in the POM