

# Jetty Remote Deployer

This document explains how to configure Cargo remote deployment support on a Jetty container.

## Overview

By default, Jetty does not come with possibilities for remote deployment. In order to add such a support to Jetty, Cargo uses a "Jetty remote deployer" Web application.

This application is a simple servlet-based application which exposes methods such as deploy or undeploy on standard HTTP POST URLs. When these HTTP methods are called, the servlet implementing these methods connects to the Jetty Server implementation and does deployment related actions on the server; this Web application can therefore be seen as a kind of remote administration proxy.

Being a standard Web application, the Cargo Jetty remote deployer application can be secured using Jetty users and roles.

## Downloading the remote deployer

Two versions of the Jetty remote deployer WAR are available on the [Cargo downloads](#) page (scroll down to the **Tools** section):

- `cargo-jetty-7-and-onwards-deployer`: The Deployer Web application for the Jetty remote containers, which must have been deployed to Jetty before using the CARGO remote deployer. Designed to work with Jetty 7.x and later (Jetty from Eclipse.org)
- `cargo-jetty-6-and-earlier-deployer`: The Deployer Web application for the Jetty remote containers, which must have been deployed to Jetty before using the CARGO remote deployer. Designed to work with Jetty 6.x and earlier (Jetty from Mortbay.org)

Please make sure to download and install the correct flavour for your Jetty version.

## Security

By default, the Cargo Jetty remote deployer comes with no security.

In order to activate security, follow these steps:

1. Open the `WEB-INF/web.xml` file of the Cargo Jetty remote deployer WAR
2. Uncomment the part that says `Uncomment` in order to activate security. By default, that configuration is as follows:
  - Authentication is done using standard HTTP headers: `login-config` set to `BASIC`.
  - Authorization is done using Jetty role: `security-constraint` has a `auth-constraint` with `role-name`.
3. Create a user with the Jetty role manager:
  - a. Open the Jetty `realm.properties` file
  - b. Add, for example, the following definition

```
someusername: somepassword,manager
```

To try the security settings, you can try to visit the `/cargo-jetty-deployer` context on your server, for example

<http://production27:8080/cargo-jetty-deployer>, using any Web browser. If security is configured well, it should:

- Ask for a login and password
- Ask again if the login is not valid
- If the login is valid, show a page saying: `Command / is unknown`

**Note:** Jetty's website has documentation on hashing the password.

## Examples

Here is an example Maven2 plugin configuration that:

- Deploys on a remote Jetty 6.x server
- The server is on `production17`, port 8080
- The Jetty remote deployer WAR is secured using the Jetty role manager
- A user called `someusername` with password `somepassword` is defined as manager

```
<dependencies>
  <dependency>
    <groupId>test.somegroup</groupId>
    <artifactId>somewar</artifactId>
    <version>1.0.0</version>
    <type>war</type>
  </dependency>
</dependencies>

...

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

    <artifactId>cargo-maven2-plugin</artifactId>

    <version>${cargo.plugin.version}</version>
    <configuration>
      <container>
```

```
    <containerId>jetty6x</containerId>
    <type>remote</type>
</container>
```

```
<configuration>
  <type>runtime</type>
  <properties>
```

```
<cargo.hostname>production17</cargo.hostname>
>
```

```
<cargo.servlet.port>8080</cargo.servlet.port>
>
```

```
<cargo.remote.username>someusername</cargo.remote.username>
```

```
<cargo.remote.password>somepassword</cargo.remote.password>
```

```
  </properties>
</configuration>
```

```
<deployer>
  <type>remote</type>
</deployer>
```

```
<deployables>
  <deployable>
    <groupId>test.somegroup</groupId>
    <artifactId>somewar</artifactId>
    <type>war</type>
```

```
<properties>  
  <context>/myAppContext</context>  
</properties>  
</deployable>  
</deployables>
```

```
</configuration>
</plugin>
</plugins>
```

To run the given Maven2 plugin configuration on a simple Maven2 WAR project, simply execute:

```
mvn war:war
mvn cargo:deploy
```

## Known issues

### Unexpected end of file from server and maxIdleTime parameter

By default, Jetty sets the `maxIdleTime` parameter in the `etc/jetty.xml` file 30 seconds. This means that if the actual deployment takes more than 30 seconds, you might get error messages like:

```
[ERROR] Failed to execute goal
org.codehaus.cargo:cargo-maven2-plugin:1.2.2:deploy
(default-cli)
    on project display: Execution default-cli of goal
org.codehaus.cargo:cargo-maven2-plugin:1.2.2:deploy
    failed: Failed to deploy
[/var/lib/jenkins/workspace/Vanessa-cargo/target/display-1.2.2.1
-SNAPSHOT.war]:
    Unexpected end of file from server -> [Help 1]

...

Caused by: java.net.SocketException: Unexpected end of file from
server
    at
sun.net.www.http.HttpClient.parseHTTPHeader(HttpClient.java:770)
    at
sun.net.www.http.HttpClient.parseHTTP(HttpClient.java:633)
    at
sun.net.www.http.HttpClient.parseHTTPHeader(HttpClient.java:767)
    at
sun.net.www.http.HttpClient.parseHTTP(HttpClient.java:633)
    at
sun.net.www.protocol.http.HttpURLConnection.getInputStream(HttpU
RLConnection.java:1162)
    at
java.net.HttpURLConnection.getResponseCode(HttpURLConnection.jav
a:397)
    at
org.codehaus.cargo.container.jetty.JettyRemoteDeployer.getRespon
seMessage(JettyRemoteDeployer.java:262)
    at
org.codehaus.cargo.container.jetty.JettyRemoteDeployer.deploy(Je
ttyRemoteDeployer.java:113)
    ... 25 more
```

In this case, open the target Jetty container's etc/jetty.xml file and set the maxIdleTime parameter to longer. For example:

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Mort Bay Consulting//DTD
Configure//EN" "http://jetty.mortbay.org/configure.dtd">

<Configure id="Server" class="org.mortbay.jetty.Server">
  ...
  <Call name="addConnector">
    <Arg>
      <New class="org.mortbay.jetty.nio.SelectChannelConnector">
        ...
        <!-- Set a longer maxIdleTime (90 seconds in this case)
to allow long remote deployments -->
        <Set name="maxIdleTime">90000</Set>
        ...
      </New>
    </Arg>
  </Call>
  ...
</Configure>
```