

**Appian**

# Appian Configuration Manager

Appian 7.8

The information on this page is provided by the Appian Center of Excellence

Toggle Na



This page is *not* supported by Appian Technical Support. For additional assistance, please contact your Appian Account Executive to engage with Appian Professional Services.

## Purpose

The Appian Configuration Manager represents a standard deployment process for Appian platform configurations and customizations. It is a standardize tool that prepares, packages and deploys configuration files, plugins and customizations from a standard structure into an Appian instance. The Configuration Manager provides these main benefits:

- Provides a standard way to manage configuration, plugins and additional customizations
- Eliminates the possibility of human error when merging/renaming files
- Speeds up the deployment process
- Enforces agreed upon deployment best practices and standardized techniques
- Allows for source control of environment configurations and customizations
- Works across operating systems and environments

**Note:** The Appian Configuration Manager is meant to ease deployment and is provided on an as-is basis with no warranty.

## Requirements

- Appian 7.7+ (the latest version is NOT compatible with Appian 7.6)
- Apache Ant 1.8.x+ (included with Appian since 7.5 in /thirdpartyapps/apache-ant)

## Technology

### Scripting Language

The Appian Configuration Manager is implemented using the [Apache Ant™](#) scripting language. The minimum Ant version that is required is 1.8.x and is included with Appian 7.5+ in the /thirdpartyapps/apache-ant folder. Additional libraries used by the Appian Configuration Manager are embedded in the package to ensure that all features work properly for all project implementations.

### Version Control System

In order to reap the full benefits of the Configuration Manager, a VCS is highly recommended for all project implementations. The tool can be readily committed and managed in a version controlled repository (see details below).

## Initial Setup

In order to use the Manager, you need to download it from Appian Forum and then extract it into your project's version controlled repository:

1. Navigate to the [Appian Software](#) record that matches your version of Appian.
2. Select Appian Version.
3. Locate the Add-Ons section.
4. Click the appropriate Appian Configuration Manager link to download the archive file.
  - Make sure to download the version of the Appian Configuration Manager that matches your version of Appian
  - Updated versions of the Appian Configuration Manager will be released periodically. Check back often!
5. Extract the archive file into your version controlled repository and commit.

## Recent Changes

### 7.7 Enhancement - 2014-10-24

- Fixed building on Linux

### 7.7 Initial Release - 2014-10-08

- Added support for Jboss 6.3, removed support for JBoss 6.2
- Increased Java memory settings for JBoss
- Note: this version is NOT compatible with 7.6.

### 7.6 Enhancement - 2014-09-26

- Added improved properties validation. The tool will now display "ERROR HERE" anytime there is a missing token. Previously, this analysis was only

performed on custom.properties

#### 7.5 Initial Release - 2014-04-30

- Initial release for Appian 7.5.
- Note: this version is NOT compatible with 7.4.

#### 7.4 Initial Release - 2014-01-15

- Initial release for Appian 7.4.

#### 7.3 Initial Release - 2013-10-07

- Initial release for Appian 7.3

#### 5.5 - 7.2 PS Build Releases

- The Manager is a tool that has enjoyed many releases and years of support as PS Build.

## Features

The Appian Configuration Manager contains script that will:

- Install and configure JBoss EAP for use with Appian (JBoss installer file needs to be downloaded and provided separately)
- Deploy custom.properties file with environment specific values
- Deploy Spring Security configuration files
- Compile and deploy plugins

## Process Overview

The Appian Configuration Manager prepares a new enterprise application deployment package based on the base product files and the custom configurations:

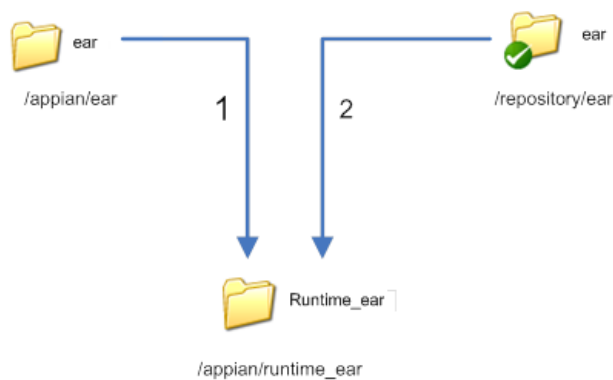


Diagram Steps:

1. Creates a new folder runtime\_ear in the Appian install folder
2. Copies contents of the folder <appian>/ear into the folder runtime\_ear
  - Prepare and package the contents of the Manager's folders
3. Copies the packaged contents in the folder runtime\_ear

The benefits of this procedure is that the base web application files are never modified.

The application server must be reconfigured to use the files located into the folder runtime\_ear and this is done automatically by the Manager when using JBoss application server.

The Manager should be executed on the server running Appian. If using a dedicated build machine, additional steps are necessary, for example changing the location of ear in application server configuration, or deploying data source configuration files.

## How to use the Manager

### Prerequisites

#### Configure ANT\_HOME

1. Create an environment variable named `ANT_HOME` and point to the `/thirdpartyapps/apache-ant` directory in your Appian install.
2. Add
  - [Windows] `%ANT_HOME%\bin` to the `Path` environment variable
  - [Unix based OS] `$ANT_HOME/bin` to the `PATH` environment variable

#### Configure environment specific files

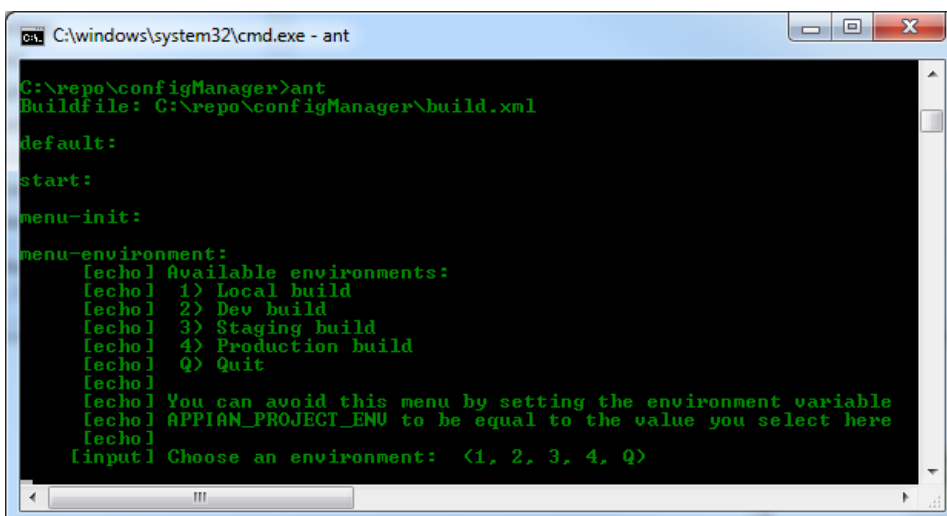
1. Use `<project.home>/build/local.properties.example` file as a guiding tool to create a properties file specific to your environment.
  - Four environments are available for selection by default - local, dev, staging and prod (for example, dev.properties needs to be created for the Development environment)

2. Fill out the properties the same way you would be filling out custom.properties according to [Post-Install\\_Configurations](#).
  - EXAMPLE:
    - All custom properties must be defined in the [project.home]/ear/suite.ear/conf/custom.properties file using tokens
      - custom.properties: conf.suite.SERVER\_AND\_PORT=@conf.suite.SERVER\_AND\_PORT@
    - The values of the tokens must be configured in the `<env>`.properties and are applied automatically when building the application
      - local.properties: conf.suite.SERVER\_AND\_PORT=localhost:8080
      - staging.properties: conf.suite.SERVER\_AND\_PORT=staging-applan.mydomain.com
      - prod.properties: conf.suite.SERVER\_AND\_PORT=applan.mydomain.com
  - NOTE: Any additional files that are required for your install need to be added to the Manager's folder structure to be correctly deployed (for example Spring Security configuration, incoming email setup, etc.). Any environment specific values can be tokenized (@my.tokenized.property@) and the value set in your [environment].properties file.
  - NOTE: Any additional properties not covered under the example file can be added to your [environment].properties file. Corresponding keys need to be added to custom.properties so that they get deployed correctly to /runtime\_ear
  - NOTE: The Manager does not generate encrypted data source passwords. This step needs to be completed manually and values provided in [environment].properties file.
  - NOTE: The Manager does not deploy or configure cleanup scripts. These need to be configured according to documentation ([Required\\_Configurations#Required\\_Configurations](#))
  - NOTE: The Manager does not generate a unique security token ([Applan\\_Engine\\_Connection\\_Restrictions#Generating\\_a\\_Custom\\_Security-Token](#)). This file should exist outside of the Manager folder structure and needs to be manually created.
3. If you are installing JBoss EAP, provide the path to the JBoss install archive and the installation path as well as the credentials for management use
4. [Custom Topology] Create a topology file for your environment in <project.home>/ear/suite.ear/conf (for example applan-topology.xml.dev)
5. Make sure to define properties for your data sources in [environment].properties files
6. See example below ([Configuring\\_Custom\\_Email\\_Senders](#))
  - Create custom file in [project.home]/ear/suite.ear/web.war/WEB-INF/conf/process/email-address-config-custom.xml
  - Add custom property using tokens
  - Tokens must be configured in the `<env>`.properties and are applied automatically when building the application
    - local.properties: custom.site.admin.email=admin-local@mydomain.com
    - prod.properties: custom.site.admin.email=admin-prod@mydomain.com

```
<emailConfig>
  <addresses>
    <address name="admin">
      <emailAddress>@custom.site.admin.email@</emailAddress>
    </address>
  </addresses>
</emailConfig>
```

## Executing the Manager

1. On Windows, open Command Prompt. On a Unix based OS, open Terminal.
2. Navigate to your <project.home> folder
3. Execute the `ant` command



```
C:\windows\system32\cmd.exe - ant
C:\repo\configManager>ant
Buildfile: C:\repo\configManager\build.xml

default:
start:
menu-init:
menu-environment:
[echo] Available environments:
[echo] 1) Local build
[echo] 2) Dev build
[echo] 3) Staging build
[echo] 4) Production build
[echo] Q) Quit
[echo] You can avoid this menu by setting the environment variable
[echo] APPLAN_PROJECT_ENV to be equal to the value you select here
[echo]
[input] Choose an environment: <1, 2, 3, 4, Q>
```

1. Choose your environment.

```

C:\windows\system32\cmd.exe - ant
[echo] You can avoid this menu by setting the environment variable
[echo] APPIAN_PROJECT_ENVU to be equal to the value you select here
[echo]
[input] Choose an environment: <1, 2, 3, 4, Q>
1
menu:
[echo] menu environment: 1
[echo] Choose an action:
[echo] 0> Install JBoss and Deploy all
[echo] 1> Deploy all
[echo] 2> Deploy modifications
[echo] 3> Deploy engine configuration files only
[echo] 4> Deploy all <static>
[echo] 5> Deploy modifications <static>
[echo] 6> Deploy plugins <compile/jar>
[echo] Q> Quit
[echo] You can avoid this menu by setting the environment variable
[echo] APPIAN_PROJECT_ACTION to be equal to the value you select here
[echo]
[input] Choose an action: <0, 1, 2, 3, 4, 5, 6, Q>

```

1. Choose appropriate menu action (see section below for detailed description of each action). Depending on the speed of your system and the action chosen, the execution can take a few minutes.

```

C:\windows\system32\cmd.exe
update:
BUILD SUCCESSFUL
Total time: 2 minutes 26 seconds

```

1. If a problem with the deployment was detected the build will show an error message

```

C:\windows\system32\cmd.exe
install jboss:
BUILD FAILED
C:\repo\ps-build73\build.xml:3: The following error occurred while executing this line:
C:\repo\ps-build73\build\build.xml:466: The following error occurred while executing this line:
C:\repo\ps-build73\build\menu.xml:116: The following error occurred while executing this line:
C:\repo\ps-build73\build\build.xml:440: ERROR: A JBoss installation already exists at this location.
re-execute.

```

## Menu Actions

The Standard Build Process includes seven actions ready to be used. The objective is to ensure a safe and repeatable deployment from a shared development environment to a final stage production setting. Some of these actions include doing a full deployment or just deploy the latest modifications.

```

C:\windows\system32\cmd.exe - ant
[echo] You can avoid this menu by setting the environment variable
[echo] APPIAN_PROJECT_ENVU to be equal to the value you select here
[echo]
[input] Choose an environment: <1, 2, 3, 4, Q>
1
menu:
[echo] menu environment: 1
[echo] Choose an action:
[echo] 0> Install JBoss and Deploy all
[echo] 1> Deploy all
[echo] 2> Deploy modifications
[echo] 3> Deploy engine configuration files only
[echo] 4> Deploy all <static>
[echo] 5> Deploy modifications <static>
[echo] 6> Deploy plugins <compile/jar>
[echo] Q> Quit
[echo] You can avoid this menu by setting the environment variable
[echo] APPIAN_PROJECT_ACTION to be equal to the value you select here
[echo]
[input] Choose an action: <0, 1, 2, 3, 4, 5, 6, Q>

```

The Standard Build Process is environment aware,

### Install JBoss and Deploy all (0)

This option needs to be used first when using JBoss as application server. This action performs the following tasks:

1. Unzips the provided JBoss application server archive in the specified location.
2. Creates JBoss management user with the credentials provided.
3. Executes "Deploy all" action (see below).

## Deploy all (1)

This option is used most frequently as it performs a complete cleanup and deployment. Deploy all performs the following tasks:

1. Deletes the folder runtime\_ear and all its contents.
2. Clears application server cache.
3. Creates a new runtime\_ear folder.
4. [JBoss] Deploys basic JBoss configuration files. See [Configuring JBoss](#) on deployed files.
5. [JBoss] Modifies JBoss configuration to run the Appian application from the folder /runtime\_ear instead of base product /ear.
6. Copies the content from /ear to /runtime\_ear.
7. Copies and filters all the customized files with the parameterized values from each environment from <project.home>/ear to /runtime\_ear, <project.home>/\_admin to <appian.home>/\_admin, <project.home>/server <appian.home>/server and [JBoss] <project.home>/<project.appserver.home> to <appserver.home>.
8. Creates a jar from all custom java files and copies it to <appian.home>/lib.
9. Copies plugins from <project.home>/\_admin/plugins to <appian.home>/\_admin/plugins.
10. Copies the static content from <project.webapp> to <appserver.webapp>.

## Deploy modifications (2)

This menu action provides a faster execution time as it skips steps 1-5 of "Deploy all". This is used when a minor configuration change needs to be deployed to the server. NOTES:

1. Deploy all needs to be executed before Deploy modifications can be run.
2. Since this option does not delete any files from <appian.home> any files removed from configuration in <project.home> will not be removed from your deployment.

## Deploy engines configuration files only (3)

This option is used in distributed environments only on the server running Appian engines only. Tasks performed:

1. Copies and filters three customized files used by Appian engines
  - <appian.home>/ear/suite.ear/conf/custom.properties
  - <appian.home>/ear/suite.ear/conf/appian-topology.xml
  - <appian.home>/server/\_conf/gateway-config-\*.xml
  - [Windows] <appian.home>/server/\_conf/partitions-\*.xml
  - [Windows] <appian.home>/server/\_conf/forums-config-\*.xml
  - NOTE: Since there are no application server assets on the engine server, the files are copied into <appian.home>/ear

## Deploy all (static) (4)

This target is intended to deploy static web server assets such as images, html and css files into the folder <appian.static.webapp>. This folder should have been configured to be the root directory of the Appian application on the web server. ### 5) Deploy modifications (static) ### Similar action to the Deploy All (Static) but only updates the target files based on timestamp. ### 6) Deploy plugins ### If you have custom plugin projects within <project.home>/plugin\_projects, this option will:

1. Clean <appian.home>/\_admin/plugins directory
2. Compile and jar plugins defined in <project.home>/plugin\_projects
3. Copy new jars to <appian.home>/\_admin/plugins
4. Copy existing plugin jars from <project.home>/\_admin/plugins to <appian.home>/\_admin/plugins

# Extending the Build Process

## Adding Environments

One of the most important features of the Standard Build Process is its environment aware platform. Configuring and adding environments to the build process ensures that the risk associated with introducing changes between environments is minimized. To be able to add a new environment, there are three simple steps to follow.

1. Define a new environment, i.e. production-apollo
2. Add the option in <project.home>/build/menu.xml (see bold in example below)

```
<target name="menu-environment" description="Entry point for build script" unless="menu.environment">
<!-- display the build choices -->
<echo message="Available environments:"/>
<echo message=" 1) Local build"/>
<echo message=" 2) Dev build"/>
<echo message=" 3) Staging build"/>
<echo message=" 4) Production build"/>
<echo message=" 5) Production-apollo build"/>
<echo message=" Q) Quit"/>
<input message="Choose an environment: " validargs="1,2,3,4,'5',Q" addproperty="menu.environment"/>
...
""<condition property="environment" value="production-apollo"> ""
""<equals arg1="5" arg2="{menu.environment}"/> ""
```

</condition>

Create the property file based on local.properties <project.home>/build/production-apollo.properties [Distributed Environment] Create the configuratio file based on appian-topology.xml.local <project.home>/ear/suite/ear/conf/appian-topology.xml.production-apollo

Adding Menu Actions

The Standard Build Process ships with 7 configured menu actions. Following three simple steps, it is possible to add more specific tasks related to your project. To be able to add a new menu action, there are three simple steps to follow. Define a new action, i.e. JUnit Add the option in <project.home>/build/menu.xml (see bold in example below)

```
<target name="menu" description="Entry point for build script" depends="menu-init,menu-environment">
<echo message="menu environment: ${menu.environment}" />
<!-- display available actions -->
<echo message="Choose an action:"/>
<echo message=" 0) Install JBoss and Deploy all"/>
...
<b><echo message=" 7) JUnit"/></b>
<echo message=" Q) Quit"/>
<echo message=" "/>
<echo message="You can avoid this menu by setting the environment variable"/>
<echo message="APIAN_PROJECT_ACTION to be equal to the value you select here"/>
<echo message=" "/>
<input message="Choose an action: "validargs="0,1,2,3,4,5,6,""7","Q" addproperty="menu.action"/>
...
<condition property="action" value="plugins">
<equals arg1="6" arg2="${menu.action}"/>
</condition>
<b><condition property="action" value="JUnit"></b>
<b><equals arg1="7" arg2="${menu.action}"/></b>
</condition>
<condition property="overwrite" value="false" else="true">
<or>
...

```

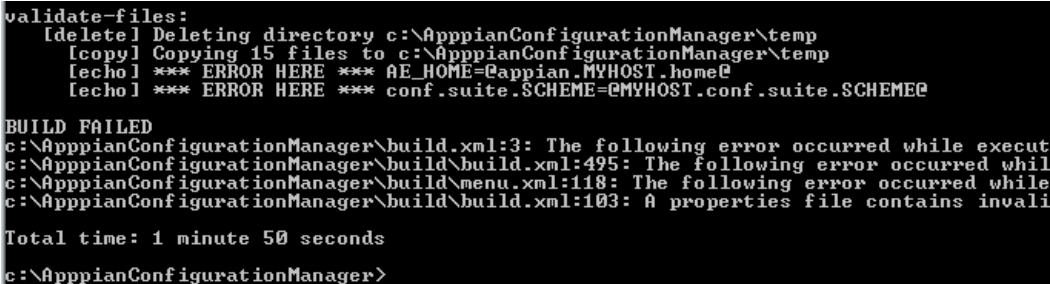
1. Add the new target to <project.home>/build/build.xml:

```
<target name="JUnit">
...
<!-- Do JUnit -->
...
</target>

```

Troubleshooting

"Error Here" appears in the build output



This means that a token defined in your Appian Configuration Manager files is not defined in your environment.properties. In this example, @appian.MYHOST.home@ and @MYHOST.conf.suite.SCHEME@ did not have corresponding values in local.properties. This will happen if you decide to modify the out of the box Appian Configuration Manager files and forget to add the corresponding token to the .properties file.

User Guides by Role

|                                |
|--------------------------------|
| Designer                       |
| Developer                      |
| Web Admin                      |
| Server Admin (On-Premise Only) |

## Tutorials

[Records](#)[Interfaces](#)[Process](#)

## Release Information

[Release Notes](#)[Installation](#)[Migration](#)[System Requirements](#)[Hotfixes](#)[Release History](#)

## Other

[STAR Methodology](#)[Best Practices](#)[Glossary](#)[APIs](#)