

Extension Internals

Table of contents

1 Overview.....	2
2 Packaging.....	2
3 Archive Format.....	2
4 CTL class loader.....	3
5 Examples.....	3

1. Overview

This document describes the format and internals of a CTL extension.

2. Packaging

While it is possible to create an extension archive by hand using the reference information found here, the preferred approach is to use the `build-library` command in ProjectBuilder.

Go to [packaging](#) in the "Developing modules" section for instructions on extension packaging.

3. Archive Format

Extension archives use JAR format with an archive structure and manifest. The archive is divided into roughly four areas of content

- `bins`: contains shell commands that will be copied to `$CTL_HOME/bin`. These should be commands useful to the whole framework. If the extension will be used on Windows platforms be sure to include `.bat` wrappers.
- `jars`: contains Ant typedefs/taskdefs and any other jar file dependencies needed by your Ant code. Be sure to set the taskdef and typedef properties in the `antproject.properties` file mentioned below.
- `modules`: contains CTL command modules that should be made available to all new projects. The `ctl-extension` command copies these to `$CTL_HOME/modules`. New projects created via `ctl-depot` copy from this directory.
- `properties`: Contains several property files to describe setup and extension configuration.

Jar structure:

```

extension-name
|
+-bins                // shell commands
|
+-jars                // java class libraries
|
|   +-extension-name.jar
+-modules             // ctl command modules
|
|   +-m1.jar
+-META-INF           // extension metadata
|
|   +-MANIFEST.MF

```

```
+--properties          // extension config properties
|
|--extension.properties
|--extension.properties.template // optional
|
|--defaults.properties // preference input
|
|--antproject.properties // contains task & type defs
```

The `ctl-extension` command reads the `MANIFEST.MF` file inside the extension archive to access metadata about the extension. If a `MANIFEST.MF` file is not found or does not use the recognized format, the `ctl-extension` will fail with an error. Below is the format:

Extension JAR Manifest Content

```
Manifest-Version: 1.0
X-ANTDEPO-Extension-Author: user
X-ANTDEPO-Extension-Name: name
X-ANTDEPO-Extension-Version: vers
X-ANTDEPO-Includes-Modules: boolean
X-ANTDEPO-Includes-Jars: boolean
X-ANTDEPO-Includes-Bins: boolean
X-ANTDEPO-Archive-Date: yyyy-MM-dd G, H:m:s z
X-ANTDEPO-Archive-Version: 1.0
```

Several attributes (e.g., `X-ANTDEPO-Includes-.*`) use boolean values to denote if the `ctl-extension` command should process that part of the extension. If set true, `ctl-extension` will copy the content to the appropriate area in the framework, otherwise these directories are ignored.

4. CTL class loader

To support loading of Ant types and tasks from extensions, CTL uses its own class loader that looks for installed extensions and reads a file called `antproject.properties`.

The `antproject.properties` can contain two properties: `taskdefs` and `typedefs`. The format of each properties file is based on the Ant [typedef](#) format. An example of an `antproject.properties` file is shown below:

```
taskdefs = /resource/path/to/taskdef.properties
typedefs = /resource/path/to/typedef.properties
```

5. Examples

This section describes two examples of an extension archive.

The first example, is for an extension called *console*. It includes one executable console that will be copied to the `$CTL_HOME/bin` directory. Inside `jars/` is a jar file containing the underlying implementation. The `extension.properties` file includes console specific configuration data.

```
console
|
|--bins
|  |
|  |--console
|--jars           // java class libraries
|  |--console.jar
|--META-INF      // extension metadata
|  |--MANIFEST.MF
|--properties    // extension config properties
|  |--extension.properties
```

The second example shows an extension called *commander* providing a number of ant tasks and CTL modules.

```
commander
|
|--jars           // java classes implementing various tasks and types
|  |--commander.jar
|--modules       // ctl command modules
|  |--Deployment.jar
|  |--Node.jar
|--META-INF      // extension metadata
|  |--MANIFEST.MF
|--properties    // extension config properties
|  |--extension.properties // commander specific configuration properties
|  |--antproject.properties // registers types and tasks
```

The content of the `commander antproject.properties` contains the following:

```
taskdefs = /com/controltier/commander/tasks/taskdef.properties
typedefs = /com/controltier/commander/types/typedef.properties
```

The `ctl-extension` command extracts the `command.jar` under `$CTL_HOME/lib/extensions` like so:

