

# repolmport

## Command Reference

### Table of contents

1 Description.....	2
2 Usage.....	2

## 1. Description

*Import packages into the repository.*

The repoImport command recursively scans directories under `-targetdir` looking for files that match the regexes: `(filebase)(?:buildstamp)?.(extension)` loading them into the repository if the necessary metadata is provided.

There are two setup modes for repoImport:

1. Single package import: If only one build artifact is generated by the Builder, then ensure the following: `-min` and `-max` both use a value of "1", `-type` refers to an existing Package type, `-installroot` refers to a valid file path.
2. Multiple package import: If multiple build artifacts are generated by the Builder, then it is possible to use a `-propfile` as a metadata template for each kind of package to import.

### Note:

The repoImport command assumes that the Package subtype already exists and will fail if the type is not defined in the project model.

The command will fail if the `-min` and `-max` criteria are not met.

*static:* This command can be run outside of an object context.

## 2. Usage

```
ctl -m Builder -c repoImport [-buildstamp <>] [-extension <>]
[-filebase <>] [-installrank <>] [-installroot <>] [-max <>]
[-min <>] [-packageBuildtimePattern <>] [-propfile <>]
[-repouri <${framework.pkgRepo.uri}>] [-requirebuildstamp <>]
[-separator <->] [-targetdir <>] [-type <>] [-vendor <>]
[-version <>] [-versionMajor <>] [-versionMinor <>]
[-versionRelease <>]
```

### 2.1. Options

Option	Description	Type	Default
buildstamp	<i>build identifier</i>	string	<code>\${entity.attribute.buildstamp}</code>
extension	<i>package file extension</i>	string	<code>\${entity.attribute.packageExt}</code>
filebase	<i>package file base</i>	string	<code>\${entity.attribute.packageFile}</code>

installrank	<i>package install-rank</i>	string	<code>\${entity.attribute.packageInst</code>
installroot	<i>package install-root</i>	string	<code>\${entity.attribute.packageInst</code>
max	<i>maximum number of packages to import after build</i>	string	<code>\${entity.attribute.importMax}</code>
min	<i>minimum number of packages to import after build</i>	string	<code>\${entity.attribute.importMin}</code>
packageBuildtimePa	<i>timestamp pattern</i>	string	<code>\${entity.attribute.packageBuil</code>
propfile	<i>metadata template file used during package registration</i>	string	
repouri	<i>repository URI</i>	string	<code>\${framework.pkgRepo.uri}</code> / <code>\${entity.attribute.packageRepo</code>
requirebuildstamp	<i>whether or not to require that files be matched by buildstamp</i>	string	
separator	<i>package name separator character</i>	string	- / <code>\${entity.attribute.packageSepa</code>
targetdir	<i>top level directory containing build artifacts</i>	string	<code>\${entity.attribute.targetdir}</code>
type	<i>package type</i> This must be a pre-existing type in the project model.	string	<code>\${entity.attribute.packageType</code>
vendor	<i>package vendor</i>	string	/ <code>\${entity.attribute.packageVend</code>
version	<i>package version</i>	string	
versionMajor	<i>build major version number</i>	string	<code>\${entity.attribute.versionMajo</code>
versionMinor	<i>build minor version number</i>	string	<code>\${entity.attribute.versionMino</code>
versionRelease	<i>build release version</i>	string	<code>\${entity.attribute.versionRele</code>

	<i>number</i>		
--	---------------	--	--