Contents

1. Overview

The **package manifest** is an XML file called **package.xml** that must be included with any catkin-compliant package's root folder. This file defines properties about the package such as the package name, version numbers, authors, maintainers, and dependencies on other catkin packages. Note the concept is similar to the *manifest.xml* file used in the legacy rosbuild (/rosbuild) build system.

Your system package dependencies are declared in `package.xml`. If they are missing or incorrect, you may be able to build from source and run tests on your own machine, but your package will not work correctly when released to the ROS community. Others depend on this information to install the software they need for using your package (excepted from task-oriented doc for catkin at U-Texas (http://farnsworth.csres.utexas.edu/docs/catkin/html/howto/system_library_dependencies.html)).

2. Format 2 (Recommended)

This is the recommended format for new packages. It is also recommended that older format 1 packages be migrated to format 2. For instructions on migrating from format 1 to format 2, see Migrating from Format 1 to Format 2 (https://docs.ros.org/melodic/api/catkin/html/howto/format2/migrating_from_format_1.html#migrating-from-format1-to-format2) in the catkin API docs.

The full documentation for format 2 can be found in the catkin API docs (https://docs.ros.org/melodic/api/catkin/html/howto/format2/index.html). More information can be found in REP 140 -- Package Manifest Format Two Specification (http://www.ros.org/reps/rep-0140.html).

2.1 Basic Structure

Each package.xml file has the `<package>` tag as the root tag in the document.

```
<package format="2">

</package>
```

2.2 Required Tags

There are a minimal set of tags that need to be nested within the `<package>` tag to make the package manifest complete.

- `<name>` - The name of the package
- `<version>` - The version number of the package (required to be 3 dot-separated integers)
- `<description>` - A description of the package contents
- `<maintainer>` - The name of the person(s) that is/are maintaining the package
- `<license>` - The software license(s) (e.g. GPL, BSD, ASL) under which the code is released.

As an example, here is package manifest for a fictional package called *foo_core*.

```
<package format="2">
  <name>foo_core</name>
  <version>1.2.4</version>
  <description>
  This package provides foo capability.
  </description>
  <maintainer email="ivana@osrf.org">Ivana Bildbotz</maintainer>
  <license>BSD</license>
</package>
```

2.3 Dependencies

The package manifest with minimal tags does not specify any dependencies on other packages. Packages can have six types of dependencies:

- **Build Dependencies** specify which packages are needed to build this package. This is the case when any file from these packages is required at build time. This can be including headers from these packages at compilation time, linking against libraries from these packages or requiring any other resource at build time (especially when these packages are *find_package()*-ed in CMake). In a cross-compilation scenario build dependencies are for

the targeted architecture.

- **Build Export Dependencies** specify which packages are needed to build libraries against this package. This is the case when you transitively include their headers in public headers in this package (especially when these packages are declared as (CATKIN_)DEPENDS in *catkin_package()* in CMake).
- **Execution Dependencies** specify which packages are needed to run code in this package. This is the case when you depend on shared libraries in this package (especially when these packages are declared as (CATKIN_)DEPENDS in *catkin_package()* in CMake).
- **Test Dependencies** specify only *additional* dependencies for unit tests. They should never duplicate any dependencies already mentioned as build or run dependencies.
- **Build Tool Dependencies** specify build system tools which this package needs to build itself. Typically the only build tool needed is catkin. In a cross-compilation scenario build tool dependencies are for the architecture on which the compilation is performed.
- **Documentation Tool Dependencies** specify documentation tools which this package needs to generate documentation.

These six types of dependencies are specified using the following respective tags:

- `<depend>` specifies that a dependency is a build, export, and execution dependency. This is the most commonly used dependency tag.
- `<build_depend>`
- `<build_export_depend>`
- `<exec_depend>`
- `<test_depend>`
- `<buildtool_depend>`
- `<doc_depend>`

All packages have at least one dependency, a build tool dependency on catkin as the following example shows.

```
<package format="2">
  <name>foo_core</name>
  <version>1.2.4</version>
  <description>
    This package provides foo capability.
  </description>
  <maintainer email="ivana@osrf.org">Ivana Bildbotz</maintainer>
  <license>BSD</license>

  <buildtool_depend>catkin</buildtool_depend>
</package>
```

A more realistic example that specifies build, exec, test, and doc dependencies could look as follows.

```
<package format="2">
  <name>foo_core</name>
  <version>1.2.4</version>
  <description>
    This package provides foo capability.
  </description>
  <maintainer email="ivana@willowgarage.com">Ivana Bildbotz</maintainer>
  <license>BSD</license>

  <url>http://ros.org/wiki/foo_core</url>
  <author>Ivana Bildbotz</author>

  <buildtool_depend>catkin</buildtool_depend>

  <depend>roscpp</depend>
  <depend>std_msgs</depend>

  <build_depend>message_generation</build_depend>

  <exec_depend>message_runtime</exec_depend>
  <exec_depend>rospy</exec_depend>

  <test_depend>python-mock</test_depend>

  <doc_depend>doxygen</doc_depend>
</package>
```

More details on dependencies can be found in the catkin API docs here (https://docs.ros.org/melodic/api/catkin/html/howto/format2/catkin_library_dependencies.html).

2.4 Metapackages

It is often convenient to group multiple packages as a single logical package. This can be accomplished through **metapackages**. A metapackage is a normal package with the following export tag in the package.xml:

```
<export>
  <metapackage />
</export>
```

Other than a required <buildtool_depends> dependency on catkin, metapackages can only have execution dependencies on packages of which they group.

Additionally a metapackage has a required, boilerplate CMakeLists.txt file:

```
cmake_minimum_required(VERSION 2.8.3)
project(<PACKAGE_NAME>)
find_package(catkin REQUIRED)
catkin_metapackage()
```

Note: replace <PACKAGE_NAME> with the name of the metapackage.

2.5 Additional Tags

- <url> - A URL for information on the package, typically a wiki page on ros.org.
- <author> - The author(s) of the package

3. Format 1 (Legacy)

Older catkin pakages use format 1. If the <package> tag has no format attribute, it is a format 1 package. Use format 2 for new packages.

The format of package.xml is straightforward.

3.1 Basic Structure

Each package.xml file has the <package> tag as the root tag in the document.

```
<package>

</package>
```

3.2 Required Tags

There are a minimal set of tags that need to be nested within the <package> tag to make the package manifest complete.

- <name> - The name of the package
- <version> - The version number of the package (required to be 3 dot-separated integers)
- <description> - A description of the package contents
- <maintainer> - The name of the person(s) that is/are maintaining the package
- <license> - The software license(s) (e.g. GPL, BSD, ASL) under which the code is released.

As an example, here is package manifest for a fictional package called *foo_core*.

```
<package>
  <name>foo_core</name>
  <version>1.2.4</version>
  <description>
  This package provides foo capability.
  </description>
  <maintainer email="ivana@willowgarage.com">Ivana Bildbotz</maintainer>
  <license>BSD</license>
</package>
```

3.3 Build, Run, and Test Dependencies

The package manifest with minimal tags does not specify any dependencies on other packages. Packages can have four types of dependencies:

- **Build Tool Dependencies** specify build system tools which this package needs to build itself. Typically the only build tool needed is catkin. In a cross-compilation scenario build tool dependencies are for the architecture on which the compilation is performed.
- **Build Dependencies** specify which packages are needed to build this package. This is the case when any file from these packages is required at build time. This can be including headers from these packages at compilation time, linking against libraries from these packages or requiring any other resource at build time (especially when these packages are *find_package()*-ed in CMake). In a cross-compilation scenario build dependencies are for the targeted architecture.
- **Run Dependencies** specify which packages are needed to run code in this package, or build libraries against this package. This is the case when you depend on shared libraries or transitively include their headers in public headers in this package (especially when these packages are declared as (CATKIN_)DEPENDS in *catkin_package()* in CMake).
- **Test Dependencies** specify only *additional* dependencies for unit tests. They should never duplicate any dependencies already mentioned as build or run dependencies.

These four types of dependencies are specified using the following respective tags:

- <buildtool_depend>
- <build_depend>
- <run_depend>
- <test_depend>

All packages have at least one dependency, a build tool dependency on catkin as the following example shows.

```
<package>
  <name>foo_core</name>
  <version>1.2.4</version>
  <description>
    This package provides foo capability.
  </description>
  <maintainer email="ivana@willowgarage.com">Ivana Bildbotz</maintainer>
  <license>BSD</license>

  <buildtool_depend>catkin</buildtool_depend>
</package>
```

A more realistic example that specifies build, runtime, and test dependencies could look as follows.

```
<package>
  <name>foo_core</name>
  <version>1.2.4</version>
  <description>
    This package provides foo capability.
  </description>
  <maintainer email="ivana@willowgarage.com">Ivana Bildbotz</maintainer>
  <license>BSD</license>

  <url>http://ros.org/wiki/foo_core</url>
  <author>Ivana Bildbotz</author>

  <buildtool_depend>catkin</buildtool_depend>

  <build_depend>message_generation</build_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>std_msgs</build_depend>

  <run_depend>message_runtime</run_depend>
  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>

  <test_depend>python-mock</test_depend>
</package>
```

More details on dependencies can be found here (/catkin/conceptual_overview#Dependency_Management).

3.4 Metapackages

It is often convenient to group multiple packages as a single logical package. This can be accomplished through **metapackages**. A metapackage is a normal package with the following export tag in the package.xml:

```
<export>
  <metapackage />
</export>
```

Other than a required <buildtool_depends> dependency on catkin, metapackages can only have run dependencies on packages of which they group.

Additionally a metapackage has a required, boilerplate CMakeLists.txt file:

```
cmake_minimum_required(VERSION 2.8.3)
project(<PACKAGE_NAME>)
find_package(catkin REQUIRED)
catkin_metapackage()
```

Note: replace <PACKAGE_NAME> with the name of the metapackage.

Metapackage now have CMakeLists.txt files after a discussion about installing the package.xml files:

https://groups.google.com/forum/#!msg/ros-sig-buildsystem/mn-VCkl2OHk/dUsHBBjyK30J (https://groups.google.com/forum/#!msg/ros-sig-buildsystem/mn-VCkl2OHk/dUsHBBjyK30J)

So now the package.xml files for metapackages are installed, but the requirement that other packages should not depend on metapackages is still enforced by requiring the users do not deviate from the suggested boilerplate CMakeLists.txt code.

3.5 Additional Tags

- `<url>` - A URL for information on the package, typically a wiki page on ros.org.
- `<author>` - The author(s) of the package

For more info see http://ros.org/reps/rep-0127.html (http://ros.org/reps/rep-0127.html)

Except where otherwise noted, the ROS wiki is licensed under the
Creative Commons Attribution 3.0 (http://creativecommons.org/licenses/by/3.0/)

Wiki: catkin/package.xml (last edited 2019-07-24 22:49:57 by MaryaBelanger (/MaryaBelanger))

Brought to you by: Open Robotics

(https://www.openrobotics.org/)