Packaging Libraries for Linux Distributions with CMake

Libraries for message parsing, message routing, logging, and others will need to be packaged for easier inclusion into build/run environments for xAPPs. This page contains notes on various methods for generating packages in Debian (.deb) and RedHat Package Manager (.rpm) formats.

Using CMake

CMake is a build tool generally used for describing and building complex C applications, but is also makes the generation of packages (e.g. .deb or .rpm) very simple. The CPACK extension to CMake is used to generate the necessary rules in a project's Makefile such that the command `make package` will create all packages. At a high level, package creation via CMake is accomplished by defining a set of CPACK related variables, and then including the Cpack module. The following code sample illustrates this:

CMake Sample

```
IF( EXISTS "${CMAKE_ROOT}/Modules/CPack.cmake" )
  include( InstallRequiredSystemLibraries )

  set( CPACK_SET_DESTDIR "on" )
  set( CPACK_PACKAGING_INSTALL_PREFIX "${install_root}" )
  set( CPACK_GENERATOR "DEB;RPM" )

  set( CPACK_PACKAGE_DESCRIPTION "<description text>" )
  set( CPACK_PACKAGE_DESCRIPTION_SUMMARY "<very short description>" )
  set( CPACK_PACKAGE_VENDOR "None" )
  set( CPACK_PACKAGE_CONTACT "None" )
  set( CPACK_PACKAGE_VERSION_MAJOR "${major_version}" )
  set( CPACK_PACKAGE_VERSION_MINOR "${minor_version}" )
  set( CPACK_PACKAGE_VERSION_PATCH "${patch_level}" )
  set( CPACK_PACKAGE_FILE_NAME
    "${CMAKE_PROJECT_NAME}_${major_version}.${minor_version}.${CPACK_PACKAGE_VERSION_PATCH}${spoiled_str}" )
  set( CPACK_SOURCE_PACKAGE_FILE_NAME
    "vric${CMAKE_PROJECT_NAME}_${major_version}.${minor_version}.${CPACK_PACKAGE_VERSION_PATCH}" )

  # omit if not required
  set( CPACK_DEBIAN_PACKAGE_DEPENDS "barpkg ( >= 1.1.4 ), foopkg ( >= 1.1.1 )" )
  set( CPACK_RPM_PACKAGE_REQUIRES "barpkg >= 2.5.0, foopkg >= 2.8" )

  set( CPACK_DEBIAN_PACKAGE_PRIORITY "optional" )
  set( CPACK_DEBIAN_PACKAGE_SECTION "ric" )
  set( CPACK_DEBIAN_ARCHITECTURE ${CMAKE_SYSTEM_PROCESSOR} )
  set( CPACK_RPM_ARCHITECTURE ${CMAKE_SYSTEM_PROCESSOR} )

  INCLUDE( CPack )
ENDIF()
```

When generating .rpm packages on a Ubuntu based system (assumed in the example above) the *alien* package must be installed on the system (in the container) such that the rpmbuild command is available. CMake has a deep set of documentation: CMake Documentation