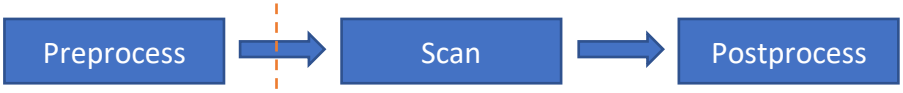
Specification for data exchange from make wrapper to scanner

Overview

Conceptually, the whole scanning process can be divided into 3 phases:



This document defines the specification for data transferred from preprocess phase to scan phase. In the picture above, preprocess phase runs in user own development environment. The scan and postprocess phases either run in docker container on the same machine, or are SaaS hosted by private/public cloud.

In general, all data generated by preprocess and passed to scan phase is packaged in to a single file. Later on, the file is copied into docker container or uploaded to SaaS by HTTP/HTTPS protocol. This document describes the directory structure of the package, list of files, format and content of each files in the package.

Directory structure

Name	Type	Description
xcalbyte.properties	File	Properties of the project been scanned
src	Directory	Directory for user source code. If the source code can be fetched from VCS, this directory does not exist or is empty. The structure of this directory must keep the same as user source code inside the src_root_dir specified in xcalibyte.properties. So that report generator can pick up the right file in this directory.
src.diff	File	Only valid if the source code can be fetched from VCS. This file is generated in src_root_dir by 'git diff'/'svn diff'/etc and can be applied to source by 'patch -pN …'

checksum.sha1	File	Checksum for files to be verified. The extension name of this file specifies the digest algorithm used in calculating the checksum.
<target>	Directory	Directory for targets been built in this project

For the sub directory for a target, the structure is listed below:

Name	Type	Description
xcalbyte.properties	File	Optional. Properties of this target. The values here will override values in top xcalibyte.properties.
src	Directory	Optional. Directory for user source code for this target. For Java program, the source code to be scanned is placed here. For C/C++ code, this directory is only for reference.
src.diff	File	Optional. Code changes for this target. Only available if the source code is managed by VCS.
preprocess	Directory	Optional. Preprocessed files for C/C++ program. Either preprocessed files or whirl ir files should be present for scan. No sub directory.
jarfiles	Directory	Optional. Java jar files for Java program. No sub directory.
irfiles	Directory	Optional. Whirl IR files. No sub directory.
objfiles	Directory	Optional. Native object files. No sub directory.

Format of xcalbyte.properties

xcalibyte.properties is a pure text file made up by line of KEY=VALUE. There is NO space in KEY and NO space around "=". Comment line always starts with "#". Only whole line comment is allowed. KEY is in lower-case.

KEYS:

Key name	Key desc	Value desc	Example
version	Version of the specification	The version number of the specification been used.	version=1.0
project	Project name	The name of the project. The scan result of the projects with the same name will be merged and compared.	project=mastiff wopt

vcs_tool	The version control tool	Value can be empty “”, “git”, “svn”, etc. If no VCS, the value is empty.	vcs_tool=git
vcs_url	The URL of the repository in VCS	Value is the URL of the repository in VCS. If no VCS, the value is empty.	vcs_url=https://git.intranet.net/projects/mastiff
vcs_token	The token to access the repository in VCS	Value is the token string to access the VCS. If no VCS or no token needed to access VCS, the value is empty.	vcs_token= c5f9d8cb674cb70fe3fbeae01594bce41ac1164b
vcs_branch	The branch name if the source is under the control of VCS	Value is the name of the branch in VCS. If no VCS, the value is empty.	vcs_branch=develop
vcs_revision	The revision to be scanned in VCS	Value is the revision number or commit id. If no VCS, the value is empty.	vcs_revision=c575e244
src_root_dir	The local source root directory	Value is the full path name of the source root directory. If the source is fetched from VCS, the value is empty.	src_root_dir=/home/test/mastiff
build_root_dir	The local build root directory	Value is the full path name of the build root directory. If there is no local build, the value is empty.	build_root_dir=/home/test/mastiff/build
configure_command	The command to configure the build	Value is the full command to configure the build. For CMAKE managed project, “cmake <path_to_CMakeLists.txt>” is treated as the configure command. If there is no configure step in the full build process, the value is empty.	configure_command=/home/test/mastiff/configure --prefix=/home/test/mastiff/test --enable-debug

build_command	The command to build the project	Value if the full command to build the project.	build_command=make
c_compiler	The C compiler used in the build	Value is the name of the C compiler if it's found in $PATH, or the full path name of the C compiler. If there is no C source in the project, the value is empty.	c_compiler=/usr/crosstools/bin/arm-none-eabi-gcc
c_extra_flags	Extra C compiler flags passed to build command line options	Value is extra C compiler flags passed to build command line options. If the flags is specified inside the Makefile, it's no need to set here. If there is no extra C compiler flags on build command line, the value is empty.	c_extra_flags=-D_XCALIBYTE_ON=1
cxx_compiler	The C++ compiler used in the build	Value is the name of the C++ compiler if it's found in $PATH, or the full path name of the C++ compiler. If there is no C++ source in the project, the value is empty.	cxx_compiler=/usr/crosstools/bin/arm-none-eabi-g++
cxx_extra_flags	Extra C++ compiler flags passed to build command line options	Value is extra C++ compiler flags passed to build command line options. If the flag is specified inside the Makefile, it's no need to set here. If there is no extra C++ compiler flags on build command line, the value is empty.	cxx_extra_flags=-D_XCALIBYTE_ON=1
assembler	The assembler used in the build	Value is the name of the assembler if it's found in $PATH, or the full path name of the assembler. If there is no assembly code in the project, the value is empty.	assembler=/usr/crosstools/bin/arm-none-eabi-as

as_extra_flags	Extra assembler flags passed to build command line options	Value is extra assembler flags passed to build command line options. If the flag is specified inside the Makefile, it's no need to set here. If there is no extra assembler flags on build command line, the value is empty.	as_extra_files=-mthumb
archiver	The archiver used in the build	Value is the name of the archiver if it's found in $PATH, or the full path name of the archiver. If there is no static library built in the project, the value is empty.	archiver=/usr/crosstools/bin/arm-none-eabi-ar
ar_extra_flags	Extra archiver flags passed to build command line options	Value is extra archiver flags passed to build command line options. If the flag is specified inside the Makefile, it's no need to set here. If there is no extra archiver flags on build command line, the value is empty.	ar_extra_flags=
linker	The linker used in the build	Value is the name of the linker if it's found in $PATH, or the full path name of the linker. If there is no linker used in the project, the value is empty.	linker=/usr/crosstools/bin/arm-none-eabi-ld
ld_extra_flags	Extra linker flags passed to build command line options	Value is extra linker flags passed to build command line options. If the flag is specified inside the Makefile, it's no need to set here. If there is no extra linker flags on build command line, the value is empty.	ld_extra_files=

compile_only	Only compile the files but do not scan issues.	Value must be one of true and false. By default the value is false. This option is used for libraries.	compile_only=false
dependencies	Libraries used to build the project or target	The list of libraries or targets used to build the project or target. The item in the list must match with the sub directory name of the target.	dependencies=libelf libdwarf common
c_scan_options	Extra C scan options passed to xvsa	Value is extra C scan options passed to xvsa. If there is no extra C scan options, the value is empty.	c_scan_options=-std=gnu99 -m32
cxx_scan_options	Extra C++ scan options passed to xvsa	Value is extra C++ scan options passed to xvsa. If there is no extra C scan options, the value is empty.	cxx_scan_options=-std=gnu++14 -m32

Format of checksum file

Checksum file is a pure text file. Each line is the checksum for 1 file. This file can be generated from command line like:
$ find -type f -exec sha1sum {} \; >checksum.sha1

Example for mastiff WOPT

Top level files and directories:
xcalibyte.properties
checksum.sha1
src
libiberty
wopt

Top level xcalibyte.properties:
<TBD>

Top level checksum.sha1:

Sub directory for libiberty:
xcalibyte.properties
compile_only=true
…

Sub directory for wopt:
Xcalibyte.properties
compile_only=false
dependencies=libiberty
…