NAME

cmake-presets − CMakePresets.json

INTRODUCTION

One problem that CMake users often face is sharing settings with other people for common ways to configure a project. This may be done to support CI builds, or for users who frequently use the same build. CMake supports two files, **CMakePresets.json** and **CMakeUserPresets.json**, that allow users to specify common configure options and share them with others.

CMakePresets.json and **CMakeUserPresets.json** live in the project's root directory. They both have exactly the same format, and both are optional (though at least one must be present if **−−preset** is specified.) **CMakePresets.json** is meant to save project−wide builds, while **CMakeUserPresets.json** is meant for developers to save their own local builds. **CMakePresets.json** may be checked into a version control system, and **CMakeUserPresets.json** should NOT be checked in. For example, if a project is using Git, **CMakePresets.json** may be tracked, and **CMakeUserPresets.json** should be added to the **.gitignore**.

FORMAT

The files are a JSON document with an object as the root:

```
{
  "version": 3,
  "cmakeMinimumRequired": {
    "major": 3,
    "minor": 21,
    "patch": 0
  },
  "configurePresets": [
    {
      "name": "default",
      "displayName": "Default Config",
      "description": "Default build using Ninja generator",
      "generator": "Ninja",
      "binaryDir": "${sourceDir}/build/default",
      "cacheVariables": {
        "FIRST_CACHE_VARIABLE": {
          "type": "BOOL",
          "value": "OFF"
        },
        "SECOND_CACHE_VARIABLE": "ON"
      },
      "environment": {
        "MY_ENVIRONMENT_VARIABLE": "Test",
        "PATH": "$env{HOME}/ninja/bin:$penv{PATH}"
      },
      "vendor": {
        "example.com/ExampleIDE/1.0": {
          "autoFormat": true
        }
      }
    },
    {
      "name": "ninja-multi",
      "inherits": "default",
      "displayName": "Ninja Multi-Config",
      "description": "Default build using Ninja Multi-Config generator",
      "generator": "Ninja Multi-Config"
```

```
      },
      {
        "name": "windows-only",
        "inherits": "default",
        "displayName": "Windows-only configuration",
        "description": "This build is only available on Windows",
        "condition": {
          "type": "equals",
          "lhs": "${hostSystemName}",
          "rhs": "Windows"
        }
      }
    ],
    "buildPresets": [
      {
        "name": "default",
        "configurePreset": "default"
      }
    ],
    "testPresets": [
      {
        "name": "default",
        "configurePreset": "default",
        "output": {"outputOnFailure": true},
        "execution": {"noTestsAction": "error", "stopOnFailure": true}
      }
    ],
    "vendor": {
      "example.com/ExampleIDE/1.0": {
        "autoFormat": false
      }
    }
  }
```

The root object recognizes the following fields:

version
A required integer representing the version of the JSON schema. The supported versions are **1**, **2**, and **3**.

cmakeMinimumRequired
An optional object representing the minimum version of CMake needed to build this project. This object consists of the following fields:

major
An optional integer representing the major version.

minor
An optional integer representing the minor version.

patch
An optional integer representing the patch version.

vendor
An optional map containing vendor−specific information. CMake does not interpret the contents of this field except to verify that it is a map if it does exist. However, the keys should be a vendor−specific domain name followed by a /−separated path. For example, the Example IDE 1.0 could use **example.com/ExampleIDE/1.0**. The value of each field can be anything desired by the vendor, though will typically be a map.

configurePresets
An optional array of *Configure Preset* objects. This is allowed in preset files specifying version **1** or above.

buildPresets
An optional array of *Build Preset* objects. This is allowed in preset files specifying version **2** or above.

testPresets
An optional array of *Test Preset* objects. This is allowed in preset files specifying version **2** or above.

Configure Preset
Each entry of the **configurePresets** array is a JSON object that may contain the following fields:

name
A required string representing the machine−friendly name of the preset. This identifier is used in the cmake −−preset option. There must not be two configure presets in the union of **CMakePresets.json** and **CMakeUserPresets.json** in the same directory with the same name. However, a configure preset may have the same name as a build or test preset.

hidden
An optional boolean specifying whether or not a preset should be hidden. If a preset is hidden, it cannot be used in the **−−preset=** argument, will not show up in the **CMake GUI**, and does not have to have a valid **generator** or **binaryDir**, even from inheritance. **hidden** presets are intended to be used as a base for other presets to inherit via the **inherits** field.

inherits
An optional array of strings representing the names of presets to inherit from. The preset will inherit all of the fields from the **inherits** presets by default (except **name**, **hidden**, **inherits**, **description**, and **displayName**), but can override them as desired. If multiple **inherits** presets provide conflicting values for the same field, the earlier preset in the **inherits** list will be preferred. Presets in **CMakePresets.json** may not inherit from presets in **CMakeUserPresets.json**.

This field can also be a string, which is equivalent to an array containing one string.

condition
An optional *Condition* object. This is allowed in preset files specifying version **3** or above.

vendor
An optional map containing vendor−specific information. CMake does not interpret the contents of this field except to verify that it is a map if it does exist. However, it should follow the same conventions as the root−level **vendor** field. If vendors use their own per−preset **vendor** field, they should implement inheritance in a sensible manner when appropriate.

displayName
An optional string with a human−friendly name of the preset.

description
An optional string with a human−friendly description of the preset.

generator
An optional string representing the generator to use for the preset. If **generator** is not specified, it must be inherited from the **inherits** preset (unless this preset is **hidden**). In version **3** or above, this field may be omitted to fall back to regular generator discovery procedure.

Note that for Visual Studio generators, unlike in the command line **-G** argument, you cannot include the platform name in the generator name. Use the **architecture** field instead.

architecture, **toolset**
Optional fields representing the platform and toolset, respectively, for generators that support them. Each may be either a string or an object with the following fields:

value
An optional string representing the value.

strategy
An optional string telling CMake how to handle the **architecture** or **toolset** field. Valid values are:

"set"
Set the respective value. This will result in an error for generators that do not support the respective field.

"external"
Do not set the value, even if the generator supports it. This is useful if, for example, a preset uses the Ninja generator, and an IDE knows how to set up the Visual C++ environment from the **architecture** and **toolset** fields. In that case, CMake will ignore the field, but the IDE can use them to set up the environment before invoking CMake.

toolchainFile
An optional string representing the path to the toolchain file. This field supports *macro expansion*. If a relative path is specified, it is calculated relative to the build directory, and if not found, relative to the source directory. This field takes precedence over any **CMAKE_TOOLCHAIN_FILE** value. It is allowed in preset files specifying version **3** or above.

binaryDir
An optional string representing the path to the output binary directory. This field supports*macr o expansion*. If a relative path is specified, it is calculated relative to the source directory. If **binaryDir** is not specified, it must be inherited from the **inherits** preset (unless this preset is **hidden**). In version **3** or above, this field may be omitted.

installDir
An optional string representing the path to the installation directory. This field supports*macr o expansion*. If a relative path is specified, it is calculated relative to the source directory. This is allowed in preset files specifying version **3** or above.

cmakeExecutable
An optional string representing the path to the CMake executable to use for this preset. This is reserved for use by IDEs, and is not used by CMake itself. IDEs that use this field should expand any macros in it.

cacheVariables
An optional map of cache variables. The key is the variable name (which may not be an empty string), and the value is either **null**, a boolean (which is equivalent to a value of **"TRUE"** or **"FALSE"** and a type of **BOOL**), a string representing the value of the variable (which supports *macro expansion*), or an

object with the following fields:

type
An optional string representing the type of the variable.

value
A required string or boolean representing the value of the variable. A boolean is equivalent to **"TRUE"** or **"FALSE"**. This field supports *macro expansion*.

Cache variables are inherited through the **inherits** field, and the preset's variables will be the union of its own **cacheVariables** and the **cacheVariables** from all its parents. If multiple presets in this union define the same variable, the standard rules of **inherits** are applied. Setting a variable to **null** causes it to not be set, even if a value was inherited from another preset.

environment
An optional map of environment variables. The key is the variable name (which may not be an empty string), and the value is either **null** or a string representing the value of the variable. Each variable is set regardless of whether or not a value was given to it by the process's environment. This field supports *macro expansion*, and environment variables in this map may reference each other, and may be listed in any order, as long as such references do not cause a cycle (for example, if **ENV_1** is **$env{ENV_2}**, **ENV_2** may not be **$env{ENV_1}**.)

Environment variables are inherited through the **inherits** field, and the preset's environment will be the union of its own **environment** and the **environment** from all its parents. If multiple presets in this union define the same variable, the standard rules of **inherits** are applied. Setting a variable to **null** causes it to not be set, even if a value was inherited from another preset.

warnings
An optional object specifying the warnings to enable. The object may contain the following fields:

dev
An optional boolean. Equivalent to passing **-Wdev** or **-Wno-dev** on the command line. This may not be set to **false** if **errors.dev** is set to **true**.

deprecated
An optional boolean. Equivalent to passing **-Wdeprecated** or **-Wno-deprecated** on the command line. This may not be set to **false** if **errors.deprecated** is set to **true**.

uninitialized
An optional boolean. Setting this to **true** is equivalent to passing **--warn-uninitialized** on the command line.

unusedCli
An optional boolean. Setting this to **false** is equivalent to passing **--no-warn-unused-cli** on the command line.

systemVars
An optional boolean. Setting this to **true** is equivalent to passing **--check-system-vars** on the command line.

errors
An optional object specifying the errors to enable. The object may contain the following fields:

dev

An optional boolean. Equivalent to passing **−Werror=dev** or **−Wno−error=dev** on the command line. This may not be set to **true** if **warnings.dev** is set to **false**.

deprecated
An optional boolean. Equivalent to passing **−Werror=deprecated** or **−Wno−error=deprecated** on the command line. This may not be set to **true** if **warnings.deprecated** is set to **false**.

debug
An optional object specifying debug options. The object may contain the following fields:

output
An optional boolean. Setting this to **true** is equivalent to passing **−−debug−output** on the command line.

tryCompile
An optional boolean. Setting this to **true** is equivalent to passing **−−debug−trycompile** on the command line.

find
An optional boolean. Setting this to **true** is equivalent to passing **−−debug−find** on the command line.

Build Preset
Each entry of the **buildPresets** array is a JSON object that may contain the following fields:

name
A required string representing the machine−friendly name of the preset. This identifier is used in the cmake −−build −−preset option. There must not be two build presets in the union of **CMakePresets.json** and **CMakeUserPresets.json** in the same directory with the same name. However, a build preset may have the same name as a configure or test preset.

hidden
An optional boolean specifying whether or not a preset should be hidden. If a preset is hidden, it cannot be used in the **−−preset** argument and does not have to have a valid **configurePreset**, even from inheritance. **hidden** presets are intended to be used as a base for other presets to inherit via the **inherits** field.

inherits
An optional array of strings representing the names of presets to inherit from. The preset will inherit all of the fields from the **inherits** presets by default (except **name**, **hidden**, **inherits**, **description**, and **displayName**), but can override them as desired. If multiple **inherits** presets provide conflicting values for the same field, the earlier preset in the **inherits** list will be preferred. Presets in **CMakePresets.json** may not inherit from presets in **CMakeUserPresets.json**.

This field can also be a string, which is equivalent to an array containing one string.

condition
An optional *Condition* object. This is allowed in preset files specifying version **3** or above.

vendor
An optional map containing vendor−specific information. CMake does not interpret the contents of this field except to verify that it is a map if it does exist. However, it should follow the same conventions as the root−level **vendor** field. If vendors use their own per−preset **vendor** field, they should implement inheritance in a sensible manner when appropriate.

displayName
An optional string with a human−friendly name of the preset.

description
An optional string with a human−friendly description of the preset.

environment
An optional map of environment variables. The key is the variable name (which may not be an empty string), and the value is either **null** or a string representing the value of the variable. Each variable is set regardless of whether or not a value was given to it by the process's environment. This field supports macro expansion, and environment variables in this map may reference each other, and may be listed in any order, as long as such references do not cause a cycle (for example, if **ENV_1** is **$env{ENV_2}**, **ENV_2** may not be **$env{ENV_1}**.)

Environment variables are inherited through the **inherits** field, and the preset's environment will be the union of its own **environment** and the **environment** from all its parents. If multiple presets in this union define the same variable, the standard rules of **inherits** are applied. Setting a variable to **null** causes it to not be set, even if a value was inherited from another preset.

NOTE:
For a CMake project using ExternalProject with a configuration preset having environment variables needed in the ExternalProject, use a build preset that inherits that configuration preset or the ExternalProject will not have the environment variables set in the configuration preset. Example: suppose the host defaults to one compiler (say Clang) and the user wishes to use another compiler (say GCC). Set configuration preset environment variables **CC** and **CXX** and use a build preset that inherits that configuration preset. Otherwise the ExternalProject may use a different (system default) compiler than the top−level CMake project.

configurePreset
An optional string specifying the name of a configure preset to associate with this build preset. If **configurePreset** is not specified, it must be inherited from the inherits preset (unless this preset is hidden). The build directory is inferred from the configure preset, so the build will take place in the same **binaryDir** that the configuration did.

inheritConfigureEnvironment
An optional boolean that defaults to true. If true, the environment variables from the associated configure preset are inherited after all inherited build preset environments, but before environment variables explicitly specified in this build preset.

jobs
An optional integer. Equivalent to passing **−−parallel** or **−j** on the command line.

targets
An optional string or array of strings. Equivalent to passing **−−target** or **−t** on the command line. Vendors may ignore the targets property or hide build presets that explicitly specify targets. This field supports macro expansion.

configuration
An optional string. Equivalent to passing **−−config** on the command line.

cleanFirst
An optional bool. If true, equivalent to passing **−−clean−first** on the command line.

verbose

An optional bool. If true, equivalent to passing **−−verbose** on the command line.

nativeToolOptions
An optional array of strings. Equivalent to passing options after **−−** on the command line. The array values support macro expansion.

Test Preset
Each entry of the **testPresets** array is a JSON object that may contain the following fields:

name
A required string representing the machine−friendly name of the preset. This identifier is used in the ctest **−−preset** option. There must not be two test presets in the union of **CMakePresets.json** and **CMakeUserPresets.json** in the same directory with the same name. However, a test preset may have the same name as a configure or build preset.

hidden
An optional boolean specifying whether or not a preset should be hidden. If a preset is hidden, it cannot be used in the **−−preset** argument and does not have to have a valid **configurePreset**, even from inheritance. **hidden** presets are intended to be used as a base for other presets to inherit via the **inherits** field.

inherits
An optional array of strings representing the names of presets to inherit from. The preset will inherit all of the fields from the **inherits** presets by default (except **name**, **hidden**, **inherits**, **description**, and **displayName**), but can override them as desired. If multiple **inherits** presets provide conflicting values for the same field, the earlier preset in the **inherits** list will be preferred. Presets in **CMakePresets.json** may not inherit from presets in **CMakeUserPresets.json**.

This field can also be a string, which is equivalent to an array containing one string.

condition
An optional *Condition* object. This is allowed in preset files specifying version **3** or above.

vendor
An optional map containing vendor−specific information. CMake does not interpret the contents of this field except to verify that it is a map if it does exist. However, it should follow the same conventions as the root−level **vendor** field. If vendors use their own per−preset **vendor** field, they should implement inheritance in a sensible manner when appropriate.

displayName
An optional string with a human−friendly name of the preset.

description
An optional string with a human−friendly description of the preset.

environment
An optional map of environment variables. The key is the variable name (which may not be an empty string), and the value is either **null** or a string representing the value of the variable. Each variable is set regardless of whether or not a value was given to it by the process's environment. This field supports macro expansion, and environment variables in this map may reference each other, and may be listed in any order, as long as such references do not cause a cycle (for example, if **ENV_1** is **$env{ENV_2}**, **ENV_2** may not be **$env{ENV_1}**.)

Environment variables are inherited through the **inherits** field, and the preset's environment will be the union of its own **environment** and the **environment** from all its parents. If multiple presets in this

union define the same variable, the standard rules of **inherits** are applied. Setting a variable to **null** causes it to not be set, even if a value was inherited from another preset.

configurePreset
An optional string specifying the name of a configure preset to associate with this test preset. If **configurePreset** is not specified, it must be inherited from the inherits preset (unless this preset is hidden). The build directory is inferred from the configure preset, so tests will run in the same **binaryDir** that the configuration did and build did.

inheritConfigureEnvironment
An optional boolean that defaults to true. If true, the environment variables from the associated configure preset are inherited after all inherited test preset environments, but before environment variables explicitly specified in this test preset.

configuration
An optional string. Equivalent to passing **−−build−config** on the command line.

overwriteConfigurationFile
An optional array of configuration options to overwrite options specified in the CTest configuration file. Equivalent to passing **−−overwrite** for each value in the array. The array values support macro expansion.

output
An optional object specifying output options. The object may contain the following fields.

shortProgress
An optional bool. If true, equivalent to passing **−−progress** on the command line.

verbosity
An optional string specifying verbosity level. Must be one of the following:

default
Equivalent to passing no verbosity flags on the command line.

verbose
Equivalent to passing **−−verbose** on the command line.

extra
Equivalent to passing **−−extra−verbose** on the command line.

debug
An optional bool. If true, equivalent to passing **−−debug** on the command line.

outputOnFailure
An optional bool. If true, equivalent to passing **−−output−on−failure** on the command line.

quiet
An optional bool. If true, equivalent to passing **−−quiet** on the command line.

outputLogFile
An optional string specifying a path to a log file. Equivalent to passing **−−output−log** on the command line. This field supports macro expansion.

labelSummary

An optional bool. If false, equivalent to passing **−−no−label−summary** on the command line.

subprojectSummary
An optional bool. If false, equivalent to passing **−−no−subproject−summary** on the command line.

maxPassedTestOutputSize
An optional integer specifying the maximum output for passed tests in bytes. Equivalent to passing **−−test−output−size−passed** on the command line.

maxFailedTestOutputSize
An optional integer specifying the maximum output for failed tests in bytes. Equivalent to passing **−−test−output−size−failed** on the command line.

maxTestNameWidth
An optional integer specifying the maximum width of a test name to output. Equivalent to passing **−−max−width** on the command line.

filter
An optional object specifying how to filter the tests to run. The object may contain the following fields.

include
An optional object specifying which tests to include. The object may contain the following fields.

name
An optional string specifying a regex for test names. Equivalent to passing **−−tests−regex** on the command line. This field supports macro expansion. CMake regex syntax is described under string(REGEX).

label
An optional string specifying a regex for test labels. Equivalent to passing **−−label−regex** on the command line. This field supports macro expansion.

useUnion
An optional bool. Equivalent to passing **−−union** on the command line.

index
An optional object specifying tests to include by test index. The object may contain the following fields. Can also be an optional string specifying a file with the command line syntax for **−−tests−information**. If specified as a string, this field supports macro expansion.

start
An optional integer specifying a test index to start testing at.

end
An optional integer specifying a test index to stop testing at.

stride
An optional integer specifying the increment.

specificTests
An optional array of integers specifying specific test indices to run.

exclude

An optional object specifying which tests to exclude. The object may contain the following fields.

name
An optional string specifying a regex for test names. Equivalent to passing **−−exclude−regex** on the command line. This field supports macro expansion.

label
An optional string specifying a regex for test labels. Equivalent to passing **−−label−exclude** on the command line. This field supports macro expansion.

fixtures
An optional object specifying which fixtures to exclude from adding tests. The object may contain the following fields.

any
An optional string specifying a regex for text fixtures to exclude from adding any tests. Equivalent to **−−fixture−exclude−any** on the command line. This field supports macro expansion.

setup
An optional string specifying a regex for text fixtures to exclude from adding setup tests. Equivalent to **−−fixture−exclude−setup** on the command line. This field supports macro expansion.

cleanup
An optional string specifying a regex for text fixtures to exclude from adding cleanup tests. Equivalent to **−−fixture−exclude−cleanup** on the command line. This field supports macro expansion.

execution
An optional object specifying options for test execution. The object may contain the following fields.

stopOnFailure
An optional bool. If true, equivalent to passing **−−stop−on−failure** on the command line.

enableFailover
An optional bool. If true, equivalent to passing **−F** on the command line.

jobs
An optional integer. Equivalent to passing **−−parallel** on the command line.

resourceSpecFile
An optional string. Equivalent to passing **−−resource−spec−file** on the command line. This field supports macro expansion.

testLoad
An optional integer. Equivalent to passing **−−test−load** on the command line.

showOnly
An optional string. Equivalent to passing **−−show−only** on the command line. The string must be one of the following values:

human

json−v1

repeat
An optional object specifying how to repeat tests. Equivalent to passing **−−repeat** on the command line. The object must have the following fields.

mode
A required string. Must be one of the following values:

until−fail

until−pass

after−timeout

count
A required integer.

interactiveDebugging
An optional bool. If true, equivalent to passing **−−interactive−debug−mode 1** on the command line. If false, equivalent to passing **−−interactive−debug−mode 0** on the command line.

scheduleRandom
An optional bool. If true, equivalent to passing **−−schedule−random** on the command line.

timeout
An optional integer. Equivalent to passing **−−timeout** on the command line.

noTestsAction
An optional string specifying the behavior if no tests are found. Must be one of the following values:

default
Equivalent to not passing any value on the command line.

error
Equivalent to passing **−−no−tests=error** on the command line.

ignore
Equivalent to passing **−−no−tests=ignore** on the command line.

Condition
The **condition** field of a preset, allowed in preset files specifying version **3** or above, is used to determine whether or not the preset is enabled. For example, this can be used to disable a preset on platforms other than Windows. **condition** may be either a boolean, **null**, or an object. If it is a boolean, the boolean indicates whether the preset is enabled or disabled. If it is **null**, the preset is enabled, but the **null** condition is not inherited by any presets that may inherit from the preset. Sub−conditions (for example in a **not**, **anyOf**, or **allOf** condition) may not be **null**. If it is an object, it has the following fields:

type
A required string with one of the following values:

"const"
Indicates that the condition is constant. This is equivalent to using a boolean in place of the object. The condition object will have the following additional fields:

value
A required boolean which provides a constant value for the condition's evaluation.

"equals"

"notEquals"
Indicates that the condition compares two strings to see if they are equal (or not equal). The condition object will have the following additional fields:

lhs
First string to compare. This field supports macro expansion.

rhs
Second string to compare. This field supports macro expansion.

"inList"

"notInList"
Indicates that the condition searches for a string in a list of strings. The condition object will have the following additional fields:

string
A required string to search for. This field supports macro expansion.

list
A required list of strings to search. This field supports macro expansion, and uses short−circuit evaluation.

"matches"

"notMatches"
Indicates that the condition searches for a regular expression in a string. The condition object will have the following additional fields:

string
A required string to search. This field supports macro expansion.

regex
A required regular expression to search for. This field supports macro expansion.

"anyOf"

"allOf"
Indicates that the condition is an aggregation of zero or more nested conditions. The condition object will have the following additional fields:

conditions
A required array of condition objects. These conditions use short−circuit evaluation.

"not"
Indicates that the condition is an inversion of another condition. The condition object will have the following additional fields:

condition

A required condition object.

Macro Expansion

As mentioned above, some fields support macro expansion. Macros are recognized in the form **$<macro−namespace>{<macro−name>}**. All macros are evaluated in the context of the preset being used, even if the macro is in a field that was inherited from another preset. For example, if the **Base** preset sets variable **PRESET_NAME** to **${presetName}**, and the **Derived** preset inherits from **Base**, **PRESET_NAME** will be set to **Derived**.

It is an error to not put a closing brace at the end of a macro name. For example, **${sourceDir** is invalid. A dollar sign (**$**) followed by anything other than a left curly brace (**{**) with a possible namespace is interpreted as a literal dollar sign.

Recognized macros include:

${sourceDir}
Path to the project source directory.

${sourceParentDir}
Path to the project source directory's parent directory.

${sourceDirName}
The last filename component of **${sourceDir}**. For example, if **${sourceDir}** is **/path/to/source**, this would be **source**.

${presetName}
Name specified in the preset's **name** field.

${generator}
Generator specified in the preset's **generator** field. For build and test presets, this will evaluate to the generator specified by **configurePreset**.

${hostSystemName}
The name of the host operating system. Contains the same value as **CMAKE_HOST_SYSTEM_NAME**. This is allowed in preset files specifying version **3** or above.

${dollar}
A literal dollar sign (**$**).

$env{<variable−name>}
Environment variable with name **<variable−name>**. The variable name may not be an empty string. If the variable is defined in the **environment** field, that value is used instead of the value from the parent environment. If the environment variable is not defined, this evaluates as an empty string.

Note that while Windows environment variable names are case−insensitive, variable names within a preset are still case−sensitive. This may lead to unexpected results when using inconsistent casing. For best results, keep the casing of environment variable names consistent.

$penv{<variable−name>}
Similar to **$env{<variable−name>}**, except that the value only comes from the parent environment, and never from the **environment** field. This allows you to prepend or append values to existing environment variables. For example, setting **PATH** to **/path/to/ninja/bin:$penv{PATH}** will prepend **/path/to/ninja/bin** to the **PATH** environment variable. This is needed because **$env{<variable−name>}** does not allow circular references.

$vendor{<macro−name>}
An extension point for vendors to insert their own macros. CMake will not be able to use presets which have a **$vendor{<macro−name>}** macro, and effectively ignores such presets. However, it will still be able to use other presets from the same file.

CMake does not make any attempt to interpret **$vendor{<macro−name>}** macros. However, to avoid name collisions, IDE vendors should prefix **<macro−name>** with a very short (preferably <= 4 characters) vendor identifier prefix, followed by a **.**, followed by the macro name. For example, the Example IDE could have **$vendor{xide.ideInstallDir}**.

SCHEMA

This file provides a machine−readable JSON schema for the **CMakePresets.json** format.

COPYRIGHT