Configure builds using CMake

- 06/18/2020
- 6 minutes to read

Azure Sphere uses CMake to configure builds for applications with Visual Studio, Visual Studio Code, and the Windows and Linux command lines. CMake is an open-source, cross-platform make system. For general information about CMake, see the CMake Wiki.

The following sources provide information about using CMake with Visual Studio or Visual Studio Code:

- CMake projects in Visual Studio
- CMake Tools extension for Visual Studio Code

CMake builds use the following files:

Name	Purpose
azsphere_configure_tools	Specify Azure Sphere SDK tools revision.
azsphere_configure_api	Specify target API set.
azsphere_target_hardware_definition	Specify target hardware.
azsphere_target_add_image_package	Create an image package.

CMake parameters are separated by spaces. The line continuation character "^" for the Windows command line or " \ " for the Linux command line can be used for readability but is not required. The specific character is determined by the Windows or Linux terminal configuration.

CMake functions for Azure Sphere

The CMakeLists.txt file provides the general configuration settings that CMake uses to build an application. Azure Sphere supports the use of the following functions in CMakeLists.txt:

Name	Purpose
azsphere_configure_tools	Specify Azure Sphere SDK tools revision.
azsphere_configure_api	Specify target API set.
azsphere_target_hardware_definition	Specify target hardware.
azsphere_target_add_image_package	Create an image package.

If you have an existing application that was created with an SDK earlier than 20.04, see Convert an existing app to use the CMake functions.

The CMakeLists.txt file must call the project command before any of the **azsphere_** functions.

SDK tools version

You must specify the Azure Sphere SDK version for your build by calling the **azsphere_configure_tools** function to store the value in CMakeLists.txt.This call must must precede all other function calls. For example:

Copy

```
`azsphere_configure_tools(TOOLS_REVISION "20.04")`
```

Different versions of the Azure Sphere tools may offer different CMake features. For example, the 20.04 release introduced four new CMake functions that were not present in the 20.01 release. The **azsphere_configure_tools** function specifies the minimum SDK version that is required to build your application.

Target API set

You must specify the target API set for the build by using the **azsphere_configure_api** function to store the value in CMakeLists.txt. For example:

Copy

```
`azsphere_configure_api(TARGET_API_SET "5")`
```

If you change the target API set during development, you need to delete the CMake cache before you run CMake again.

Target hardware definition

You can specify the hardware you are targeting by calling the **azsphere_target_hardware_definition** function to store the value in CMakeLists.txt. For example:

Copy

```
`azsphere_target_hardware_definition(${PROJECT_NAME} TARGET_DIRECTORY
"../../../Hardware/mt3620_rdb" TARGET_DEFINITION "sample_hardware.json")`
```

Image package creation

Specify the image package file and any resource files to include when building by calling the **azsphere_target_add_image_package** function to store the value in CMakeLists.txt. The **azsphere_target_add_image_package** function and the project to build are required; the resource files are optional.

The following function call creates an image package that contains only the Azure Sphere application:

```
azsphere_target_add_image_package(${PROJECT_NAME})
```

The next example creates an image package that contains a certificate in addition to an application:

```
azsphere_target_add_image_package(${PROJECT_NAME} RESOURCE_FILES "certs/bundle.pem")
```

How to delete the CMake cache when changing configuration files

If you change one of the configuration files, you may need to delete the CMake cache to ensure that subsequent builds do not fail. Follow this procedure before attempting another build:

- For Visual Studio Code builds, run the Delete Cache and Reconfigure command.
- For command-line (CLI) builds, delete the build directory that you created in an earlier step.

Visual Studio detects changes to the CMake configuration file and auto-deletes the cache.

Convert an existing app to use the CMake functions

If you already have an Azure Sphere application that was built with CMake prior to the 20.04 SDK, you should convert it to use these new functions. You can still build such applications unchanged for now, but support for them is deprecated and may be removed in a future release.

For an example of the changes you should make, look at how the CMakeLists.txt and *.json configuration files were changed for the External MCU Update high-level app for the 20.04 release.

Note

In addition to updates to use the functions, these files have been updated in the Azure Sphere samples to use lowercase function names, thus aligning with CMake conventions.

CMakeLists.txt configuration changes

The following examples show the changes needed to update the CMakeLists.txt file from 20.01 or earlier to use the new functions.

Example 20.01 SDK CMakeLists.txt file

makefileCopy

```
CMAKE_MINIMUM_REQUIRED(VERSION 3.8)
PROJECT(ExternalMcuUpdateNrf52 C)
```

```
ADD_EXECUTABLE(${PROJECT_NAME} main.c file_view.c mem_buf.c epoll_timerfd_utilities.c
nordic/slip.c nordic/crc.c nordic/dfu_uart_protocol.c)
TARGET_LINK_LIBRARIES(${PROJECT_NAME} applibs pthread gcc_s c)

SET(ADDITIONAL_APPROOT_INCLUDES
"ExternalNRF52Firmware/blinkyV1.bin;ExternalNRF52Firmware/blinkyV1.dat;ExternalNRF52F
irmware/s132_nrf52_6.1.0_softdevice.bin;ExternalNRF52Firmware/s132_nrf52_6.1.0_softde
vice.dat")
INCLUDE("${AZURE_SPHERE_MAKE_IMAGE_FILE}")
```

Updated CMakeLists.txt file

The updated CMakeLists.txt file calls the **azsphere_configure_tools**, **azsphere_configure_api**, and **azsphere_target_hardware_definition** functions to set the SDK tools version, API set, and target hardware, respectively. It also calls **azsphere_target_add_image_package** to build the image package and optionally specify the files to include in it.

makefileCopy

```
cmake_minimum_required(VERSION 3.10)

project(ExternalMcuUpdateNrf52 C)

azsphere_configure_tools(TOOLS_REVISION "20.04")
azsphere_configure_api(TARGET_API_SET "5")

add_executable(${PROJECT_NAME} main.c file_view.c mem_buf.c epoll_timerfd_utilities.c
nordic/slip.c nordic/crc.c nordic/dfu_uart_protocol.c)
target_link_libraries(${PROJECT_NAME} applibs pthread gcc_s c)

azsphere_target_hardware_definition(${PROJECT_NAME} TARGET_DIRECTORY
"../../../Hardware/mt3620_rdb" TARGET_DEFINITION "sample_hardware.json")

azsphere_target_add_image_package(
    ${PROJECT_NAME}
    RESOURCE_FILES
        "ExternalNRF52Firmware/blinkyV1.bin"
        "ExternalNRF52Firmware/blinkyV1.dat"
        "ExternalNRF52Firmware/s132_nrf52_6.1.0_softdevice.bin"
        "ExternalNRF52Firmware/s132_nrf52_6.1.0_softdevice.dat")
```

Visual Studio CMakeSettings.json configuration changes

The following examples show the changes needed to update the CMakeSettings.json file in Visual Studio from 20.01 or earlier to use the new functions.

Example 20.01 SDK CMakeSettings.json file

JSONCopy

```
{
  "environments": [
    {
      "environment": "AzureSphere",
      "AzureSphereTargetApiSet": "4",
      "AzureSphereTargetHardwareDefinitionDirectory":
"${projectDir}\\..\\..\\..\\Hardware\\mt3620_rdb",
      "AzureSphereTargetHardwareDefinition": "sample_hardware.json"
    }
  ],
  "configurations": [
    {
      "name": "ARM-Debug",
      "generator": "Ninja",
      "configurationType": "Debug",
      "inheritEnvironments": [
        "AzureSphere"
      ],
      "buildRoot": "${projectDir}\\out\\${name}-${env.AzureSphereTargetApiSet}",
      "installRoot": "${projectDir}\\install\\${name}-
${env.AzureSphereTargetApiSet}",
      "cmakeCommandArgs": "--no-warn-unused-cli",
      "buildCommandArgs": "-v",
      "ctestCommandArgs": "",
      "variables": [
        {
          "name": "CMAKE_TOOLCHAIN_FILE",
          "value":
"${env.AzureSphereDefaultSDKDir}CMakeFiles\\AzureSphereToolchain.cmake"
        },
        {
          "name": "AZURE_SPHERE_TARGET_API_SET",
          "value": "${env.AzureSphereTargetApiSet}"
        },
        {
          "name": "AZURE_SPHERE_TARGET_HARDWARE_DEFINITION_DIRECTORY",
          "value": "${env.AzureSphereTargetHardwareDefinitionDirectory}"
        },
        {
          "name": "AZURE_SPHERE_TARGET_HARDWARE_DEFINITION",
          "value": "${env.AzureSphereTargetHardwareDefinition}"
        }
```

```
      ]
    },
    {
      "name": "ARM-Release",
      "generator": "Ninja",
      "configurationType": "Release",
      "inheritEnvironments": [
        "AzureSphere"
      ],
      "buildRoot": "${projectDir}\\out\\${name}-${env.AzureSphereTargetApiSet}",
      "installRoot": "${projectDir}\\install\\${name}-
${env.AzureSphereTargetApiSet}",
      "cmakeCommandArgs": "--no-warn-unused-cli",
      "buildCommandArgs": "-v",
      "ctestCommandArgs": "",
      "variables": [
        {
          "name": "CMAKE_TOOLCHAIN_FILE",
          "value":
"${env.AzureSphereDefaultSDKDir}CMakeFiles\\AzureSphereToolchain.cmake"
        },
        {
          "name": "AZURE_SPHERE_TARGET_API_SET",
          "value": "${env.AzureSphereTargetApiSet}"
        },
        {
          "name": "AZURE_SPHERE_TARGET_HARDWARE_DEFINITION_DIRECTORY",
          "value": "${env.AzureSphereTargetHardwareDefinitionDirectory}"
        },
        {
          "name": "AZURE_SPHERE_TARGET_HARDWARE_DEFINITION",
          "value": "${env.AzureSphereTargetHardwareDefinition}"
        }
      ]
    }
  ]
}
```

Updated SDK CMakeSettings.json file

The updated CMakeSettings.json file includes the following changes:

- In the "environments" field, only "Azure Sphere" is required.
- In the "configurations" field for both the Debug and Release builds:

- The "buildRoot" and "installRoot" values no longer require the AzureSphereTargetApiSet setting.
- The CMake toolchain is now defined in "cmakeToolChain", instead of in "variables".
- The "variables" field now specifies only the target API set and uses the new "latest-lts" value to indicate that the project should build with the most recent long-term-stable (LTS) sysroot. The AZURE_SPHERE_TARGET_HARDWARE_DEFINITION_DIRECTORY and AZURE_SPHERE_TARGET_HARDWARE_DEFINITION settings are no longer required, because these values are now set in the CMakeLists.txt file.

JSONCopy

```
{
  "environments": [
    {
      "environment": "AzureSphere"
    }
  ],
  "configurations": [
    {
      "name": "ARM-Debug",
      "generator": "Ninja",
      "configurationType": "Debug",
      "inheritEnvironments": [
        "AzureSphere"
      ],
      "buildRoot": "${projectDir}\\out\\${name}",
      "installRoot": "${projectDir}\\install\\${name}",
      "cmakeToolchain":
"${env.AzureSphereDefaultSDKDir}CMakeFiles\\AzureSphereToolchain.cmake",
      "buildCommandArgs": "-v",
      "ctestCommandArgs": "",
      "variables": [
        {
          "name": "AZURE_SPHERE_TARGET_API_SET",
          "value": "latest-lts"
        }
      ]
    },
    {
      "name": "ARM-Release",
      "generator": "Ninja",
      "configurationType": "Release",
      "inheritEnvironments": [
        "AzureSphere"
      ],
      "buildRoot": "${projectDir}\\out\\${name}",
```

```
      "installRoot": "${projectDir}\\install\\${name}",
      "cmakeToolchain":
"${env.AzureSphereDefaultSDKDir}CMakeFiles\\AzureSphereToolchain.cmake",
      "buildCommandArgs": "-v",
      "ctestCommandArgs": "",
      "variables": [
        {
          "name": "AZURE_SPHERE_TARGET_API_SET",
          "value": "latest-lts"
        }
      ]
    }
  ]
}
```

Visual Studio Code .vs/settings.json configuration changes

The following examples show the changes needed to update the .vs/settings.json file for Visual Studio Code from 20.01 or earlier to use the new functions.

Example 20.01 SDK .vs/settings.json file

JSONCopy

```
{
    "cmake.generator": "Ninja",
    "cmake.buildDirectory": "${workspaceRoot}/out/${buildType}-
${command:azuresphere.AzureSphereTargetApiSet}",
    "cmake.buildToolArgs": [ "-v" ],
    "cmake.configureArgs": [ "--no-warn-unused-cli" ],
    "cmake.configureSettings": {
        "CMAKE_TOOLCHAIN_FILE":
"${command:azuresphere.AzureSphereSdkDir}/CMakeFiles/AzureSphereToolchain.cmake",
        "AZURE_SPHERE_TARGET_HARDWARE_DEFINITION_DIRECTORY":
"${workspaceRoot}/../../../Hardware/mt3620_rdb",
        "AZURE_SPHERE_TARGET_HARDWARE_DEFINITION": "sample_hardware.json",
        "AZURE_SPHERE_TARGET_API_SET": "4"
    },
    "cmake.configureOnOpen": true,
    "C_Cpp.default.configurationProvider": "vector-of-bool.cmake-tools"
}
```

Updated .vs/settings.json file

The updated CMakeSettings.json file includes the following changes to the "cmake.configureSettings" field:

- The AZURE_SPHERE_TARGET_HARDWARE_DEFINITION_DIRECTORY and AZURE_SPHERE_TARGET_HARDWARE_DEFINITION settings are no longer required, because these values are now set in the CMakeLists.txt file.
- The AZURE_SPHERE_TARGET_API_SET value is now "latest-lts", which indicates that the project should build with the most recent long-term-stable (LTS) sysroot.

Note that the "cmake.configureArgs" field has also been deleted for reasons unrelated to CMake. (The field is no longer required because the --no-warn-unused-cli parameter is not needed for this build.)

JSONCopy

```
{
    "cmake.generator": "Ninja",
    "cmake.buildDirectory": "${workspaceRoot}/out/${buildType}-
${command:azuresphere.AzureSphereTargetApiSet}",
    "cmake.buildToolArgs": [ "-v" ],
    "cmake.configureSettings": {
        "CMAKE_TOOLCHAIN_FILE":
"${command:azuresphere.AzureSphereSdkDir}/CMakeFiles/AzureSphereToolchain.cmake",
        "AZURE_SPHERE_TARGET_API_SET": "latest-lts"
    },
    "cmake.configureOnOpen": true,
    "C_Cpp.default.configurationProvider": "vector-of-bool.cmake-tools"
}
```