

# IMXWGU

## i.MX Windows 10 IoT User's Guide

Rev. W1.2.0 — 20 September 2022

User guide

### Document information

Information	Content
Keywords	i.MX, Windows 10 IoT
Abstract	i.MX Windows 10 IoT User's Guide describes the process of building and installing Windows 10 IoT OS BSP (Board Support Package) for the i.MX platform. It also covers special i.MX features and how to use them.



## 1 Overview

The User's Guide describes the process of building and installing Windows 10 IoT OS BSP (Board Support Package) for the i.MX platform. It also covers special i.MX features and how to use them. The guide also lists the steps to run the i.MX platform, including board DIP switch settings and instructions on the usage and configuration of U-Boot bootloader. Features covered in this guide may be specific to particular boards or SOC's. For the capabilities of a particular board or SOC, see the [i.MX Windows 10 IoT Release Notes \(IMXWNR\)](#).

### 1.1 Audience

This chapter is intended for software, hardware, and system engineers planning to use the product and anyone who wants to know more about the product.

### 1.2 Conventions

This chapter uses the following conventions:

- Courier New font: This font is used to identify commands, explicit command parameters, code examples, expressions, data types, and directives.

### 1.3 How to start

The i.MX Windows 10 IoT BSP is a collection of binary files, source code, and support files you can use to create a bootable Windows 10 IoT image for i.MX development systems.

Before you start, see the [Feature List per Board chapter](#). This section lists all the i.MX boards covered by this BSP and contains a list of possible features.

### 1.4 Using Prebuilt Binaries to create an image

The Prebuild Binary package contains prebuilt release-signed binaries of the drivers and firmware required for Windows 10 IoT Enterprise to run on the NXP i.MX EVK boards. It is the fastest way to get started running on physical hardware.

If you have downloaded the BSP with the Prebuilt Binaries, see i.MX Windows 10 IoT Quick Start Guide, which will guide you through creating a Windows IoT image that includes the BSP binaries and deploying it to an i.MX EVK board.

### 1.5 Using Source Files to create image

The BSP Source Files package contains the source files of the drivers and firmware required for Windows 10 IoT Enterprise to run on NXP i.MX EVK boards. It is intended to be used as a reference for partners that have created their own hardware designs that use the i.MX 8 family of SoCs and must customize the drivers and firmware for their own design.

If you have downloaded an archive with BSP sources, you must first build the Windows drivers and Boot firmware from the source before you can create a Windows IoT image and deploy it to your device. Start from [Building Windows 10 IoT for NXP i.MX Processors](#), which will guide you through the process of building the Windows drivers and boot firmware from the source. Once you have successfully built the driver and

firmware binaries, you can go back to the chapter in i.MX Windows 10 IoT Quick Start Guide that describes how to Deploy Windows IoT image to development board.

## 1.6 References

For more information about Windows 10 IoT Enterprise, see Microsoft online documentation.

- <http://windowsondevices.com>

The following Quick Start Guides contain basic information on the board and setting it up.

- i.MX 8M Quad Evaluation Kit Quick Start Guide
- i.MX 8M Mini Evaluation Kit Quick Start Guide
- i.MX 8M Nano Evaluation Kit Quick Start Guide
- i.MX 8M Plus Evaluation Kit Quick Start Guide
- i.MX 8 information is available at <http://www.nxp.com/imx8>

## 2 Building Windows 10 IoT for NXP i.MX Processors

### 2.1 Obtaining BSP sources

#### 2.1.1 Required tools

The following tools are required to follow this chapter:

- git
- git-lfs
- software to unpack zip, gzip, and tar archives

#### 2.1.2 The NXP i.MX Windows IoT BSP source package

The Windows IoT BSP for NXP i.MX Processors consists of multiple parts, namely:

1. The NXP i.MX BSP sources package available at [www.nxp.com](http://www.nxp.com). The package contains sources for both the boot firmware and Windows drivers.
2. The i.MX firmware and NXP Code Signing Tool (CST) available at [www.nxp.com](http://www.nxp.com).
3. Open-source parts of boot firmware and tools that are not distributed as part of the NXP BSP package.
4. The Windows IoT Enterprise operating system provided by Microsoft.

To prepare sources for building BSP, follow these steps:

**Note:** Only steps 1-3 are required for building Windows drivers from the source. All steps are required for building firmware from the source.

1. Download the NXP i.MX BSP provided as `W<os_version>-imx-windows-bsp-<build_date>.zip` archive from [www.nxp.com](http://www.nxp.com).
2. Create an empty directory, further referred as `<BSP_DIR>`, and extract the downloaded archive there. The path to this directory must be as short as possible containing only letters and underscores. Braces and other special characters can cause build errors.

Shell command `unzip W<os_version>-imx-windows-bsp-<build_date>.zip -d win10-iot-bsp` can be used to extract the package. The command creates the `win10-iot-bsp` directory containing `.git` repository with the BSP release.

3. Populate the directory by running `Init.bat`, to build drivers on the Windows host, `Init.sh`, to build firmware on the Linux host. Both scripts check out sources from the repository by `git \reset --hard`. The `Init.sh` shall also check out submodules that are required to build i.MX boot firmware by `git submodule update --init --recursive`. During prerelease testing, the `Init.sh` executed inside Ubuntu environment has run into "server certificate verification failed. CAfile: /etc/ssl/certs/ca-certificates.crt CRLfile: none" error. The problem could be solved by installing `apt-transport-https ca-certificates` and `certificate update`.

```
sudo apt update ; sudo apt-get install apt-transport-https
ca-certificates -y ; sudo
\update-ca-certificates
```

4. At this point, it is possible to build the Windows drivers. Further steps are required only to build i.MX boot firmware (ATF, U-boot, and UEFI) from sources.
5. Download and extract the [Code Signing Tools \(CST\)](http://www.NXP.com) from <http://www.NXP.com>. To access this tool, create an account on the NXP website. Extract the tool inside the `bsp` repository and rename the newly created folder to `cst` to get the `<BSP_DIR>/cst` folder:

```
tar xf cst-3.1.0.tgz
mv release cst
rm cst-3.1.0.tgz
```

6. Download and extract the [i.MX firmware](#) from NXP's website and place it to `firmware-imx`. Extract the tool inside the `bsp` repository and rename the newly created folder to `firmware-imx` to get `<BSP_DIR>/firmware-imx/firmware/ddr/` in directory tree:

```
chmod +x firmware-imx-8.14.bin
./firmware-imx-8.14.bin
mv firmware-imx-8.14 firmware-imx
rm firmware-imx-8.14.bin
```

7. Your directory structure must contain the following folders.

```
- <BSP_DIR>
|- cst (manually downloaded)
|- firmware-imx (manually downloaded)
|- Documentation
|- MSRSec
|- RIoT
|- imx-atf
|- imx-mkimage
|- imx-optee-os
|- imx-windows-iot
|- mu_platform_nxp
|- patches
|- uboot-imx
```

## 2.2 Building the drivers in the BSP

Before you start building the BSP, you must have an archive with the latest BSP sources from NXP sites downloaded and extracted as described in the [Obtaining BSP sources](#) chapter.

### 2.2.1 Required tools

The following tools are required to build the drivers:

- Visual Studio 2019
- Windows Kits (ADK/SDK/WDK)

#### 2.2.1.1 Visual Studio 2019

- Make sure that you **install Visual Studio 2019 before the WDK** so that the WDK can install a required plugin.
- Download [Visual Studio 2019](#).
- During installation, select **Desktop development with C++**.
- During installation, select the following in the Individual components tab. If these options are not available, try updating VS2019 to the *Latest* release:
  - **MSVC v142 - VS 2019 C++ ARM64 Spectre-mitigated libs (16.11)**
  - **MSVC v142 - VS 2019 C++ ARM64 build tools (16.11)**
  - **Windows 10 SDK (10.0.19041.0)**

#### 2.2.1.2 Windows Kits from Windows 10, version 2004 (10.0.19041.685)

**Warning:** *IMPORTANT: Make sure that any previous versions of the ADK and WDK have been uninstalled!*

- Install [ADK 2004](#)
- Install [WDK 2004](#)
  - Scroll down and select Windows 10, version 2004.
  - Make sure that you allow the Visual Studio Extension to install after the WDK install completes.
- If the WDK installer says it could not find the correct SDK version, install [SDK 2004](#)
  - Scroll down and select Windows 10 SDK, version 2004 (10.0.19041.0).
- After installing all Windows Kits, restart the computer and check if you have the correct versions installed in the Control panel.

### 2.2.2 Structure of Windows driver sources

The *imx-windows-iot* - sources of Windows drivers have the following structure:

<b>BSP</b>	Contains boot firmware, driver binaries (generated at build time), and scripts needed to deploy BSP to the development board.
<b>build</b>	Contains build scripts, and the VS2019 solution file. components - Contains third-party binaries and utility projects. driver - Contains driver sources.
<b>hals</b>	Contains hal extension sources.

### 2.2.3 One-time environment setup

To generate driver packages on a development machine, test certificates must be installed.

1. Open an Administrator Command Prompt.
2. Navigate to your BSP and into the folder `imx-windows-iot\build\tools`.
3. Launch `StartBuildEnv.bat`.
4. Run `SetupCertificate.bat` to install the test certificates.

Some tools may not work correctly if LongPath is not enabled, therefore run the following command in the console:

Execute `reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem /v LongPathsEnabled /t REG_DWORD /d 1` command.

### 2.2.4 Building the drivers

1. Open the solution `imx-windows-iot\build\solution\iMXPlatform\iMXPlatform.sln` located in the path where you have extracted BSP archive.  
**Note:** *Microsoft pkggen requires launching Visual Studio 2019 as Administrator. Although driver projects are configured not to generate packages BSP might contain miscellaneous projects that require launching Visual Studio as Administrator to build successfully.*
2. Choose Debug or Release build type.
3. If the secure boot feature is enabled, it is required to use signed drivers.
4. To build, press Ctrl-Shift-B or choose Build -> Build Solution from the menu. It compiles all driver packages then `imx-windows-iot\BSP\IoTEntOnNXP\drivers` for deployment.
5. The updated drivers could now be injected into the installation image or manually installed to the running development board.
  - To manually install drivers, copy them to the development board via USB drive, network share, scp, remote desktop. The drivers can be installed either by clicking `install` in right-click menu of the 'inf' file or by the devcon command-line utility.
  - For debug, use the `.kdfiles` of WinDBG. To initiate driver reload, use devcon or reset the board.
  - To create an installation SD card, see i.MX Windows 10 IoT Quick Start Guide.

## 2.3 Building ARM64 Firmware

This chapter describes the process of setting up a build environment to build the latest firmware and update the firmware on the development board.

### 2.3.1 Setting up your build environment

1. Start Linux environment such as:
  - Dedicated Linux system
  - Linux Virtual Machine
  - Windows Subsystem for Linux ([WSL setup instructions](#))

**Note:** *W-imx-windows-bsp-.zip was validated with Ubuntu 20.04 in WSL and also standalone Ubuntu.*

2. Obtain and prepare the BSP sources by following all steps described in [Obtaining BSP sources](#). Use `Init.sh` not `Init.bat` to populate the repository and all submodules.
3. Install or update build tools. The shell commands below can be used to do this process on Ubuntu 20.04 or 18.04.
4. Update package manager.

```
sudo apt-get update
sudo apt-get upgrade
```

5. If Ubuntu 18.04 and possibly other older distributions are used, the mono package might be outdated causing the build to fail. For such distributions, add the mono repository to the system as described in <https://www.mono-project.com/download/stable/#download-lin> before installing the mono package.

The process is valid for Ubuntu 18.04 in December 2021:

```
sudo apt install gnupg ca-certificates
sudo apt-key adv --keyserver hkps://keyserver.ubuntu.com:80 --
recv-keys
\3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF
# Optionally key could be downloaded to a file and added
manually by 'apt-key add <keyfile>'.
# Now that certificate is installed we can add official mono
repository to repository list.
echo "deb https://download.mono-project.com/repo/ubuntu
stable-bionic main" | sudo tee /etc/apt
/sources.list.d/mono-official-stable.list
sudo apt update
```

6. Install the required software. Note that the `mu_project` currently requires python 3.8 and higher.

```
sudo apt-get install attr build-essential python3.8
python3.8-dev python3.8-venv
\device-tree-compiler bison flex swig iasl uuid-dev wget git
bc libssl-dev zlib1g-dev
\python3-pip mono-devel gawk
```

7. Download the Arm64 cross-compiler.

```
pushd ~
wget https://releases.linaro.org/components/toolchain/
binaries/7.4-2019.02/aarch64-linux-gnu
/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu.tar.xz
tar xf gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-
gnu.tar.xz
rm gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu.tar.xz
# The cross compiler prefix is required to be exported later
into AARCH64_TOOLCHAIN_PATH variable.
# export AARCH64_TOOLCHAIN_PATH=~/gcc-linaro-7.4.1-2019.02-
x86_64_aarch64-linux-gnu/bin
/aarch64-linux-gnu-
popd
```

8. Change the directory to the `BSP_DIR`. The following commands reference the files inside the BSP directory. That `BSP_DIR` contains extracted `W-imx-windows-bsp.zip`.

```
cd <BSP_DIR>
```

9. [Project MU](#) strongly suggests the use of Python Virtual Environment for each workspace. In this case, BSP revision-separated environments allow workspaces to

keep specific Pip module versions without modifying the global system state when the firmware is compiled.

**Note:** The virtual environment does not use system packages. Thus, do not use `sudo` when installing packages using `pip`.

```
python3.8 -m venv <path to new environment>
source <path to new environment>/bin/activate
eg.:
python3.8 -m venv ~/venv/win_fw_build
source ~/venv/win_fw_build/bin/activate
```

Note the path to the new environment that you have chosen. Activate the environment before building the firmware.

10. Install the required python packages.

- a. Install or update `mu_platform` Python dependencies using `pip`.

```
pushd mu_platform_nxp
pip3 install -r requirements.txt --upgrade
```

- b. Install the `pycryptodome` package (successor of `pycrypto`).

```
pip3 install pycryptodome
```

- c. Install the `pyelftools` package.

```
pip3 install pyelftools
```

11. Setup the Mu platform. (This step is optional because `buildme64.sh` does these steps automatically.)

- a. Setup and update submodules.

```
python3 NXP/MX8M_EVK/PlatformBuild.py --setup
# or NXP/MX8M_MINI_EVK/PlatformBuild.py
```

If you return here facing problems during UEFI build, use `--force` to clean the environment. Make sure to commit or stash all your changes first, `--force` argument performs `git reset --hard`.

- b. Initialize and update Mu platform dependencies. If this command fails try upgrading `mono`. The best way to do it is to uninstall `mono` and reinstall it from its official repository. The process is described at <https://www.mono-project.com/download/stable/#download-lin>.

```
python3 NXP/MX8M_EVK/PlatformBuild.py --update
# or NXP/MX8M_MINI_EVK/PlatformBuild.py
```

12. Return to BSP root.

```
popd
```

### 2.3.2 Building the firmware

This chapter assumes that the BSP sources have been prepared as described in the [Obtaining BSP sources](#) chapter.

To build the boot firmware:

1. Open cmd prompt at `BSP_DIR`.

```
cd <BSP_DIR>
```



2. Activate your python virtual environment (Use the path specified when creating the environment.)

```
source ~/venv/win_fw_build/bin/activate
```

3. Export AARCH64\_TOOLCHAIN\_PATH cross compiler prefix. In this guide, the toolchain has been placed inside home (~/) directory.

```
export AARCH64_TOOLCHAIN_PATH=~/.gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu/bin
/aarch64-linux-gnu-
```

4. Optionally, if there is a major update, you may need to step into mu\_platform\_nxp and run `python3 NXP/MX8M_EVK/PlatformBuild.py` with `--setup --force` and then `--update` manually. To get clean and up-to-date MU build environment, make sure to stash or commit your changes. The command performs `git reset --hard`.

5. Build the firmware and create `firmware.bin`. To build the boot firmware, execute the `buildme64.sh -b <BOARD_NAME> -t all [-clean] -fw script` provided in `BSP_DIR` (the root of extracted BSP sources).

The `buildme64.sh` script bundled in BSP also copies `flash.bin` and `uefi.fit` into `<BSP_DIR> /imx-windows-iot/components/Arm64BootFirmware/<board_name>`. It allows to rebuild only UEFI or U-boot.

```
./buildme64.sh -b MX8M_EVK -t all -c -fw
```

- Use `-b MX8M_EVK` or `-b 8M` to select i.MX 8M EVK
- Use `-b MX8M_MINI_EVK` or `-b 8Mm` to select i.MX 8M Mini EVK
- Use `-t all -t sign_images` to build `signed_firmware.bin`

6. To deploy `firmware.bin` to the i.MX development board, follow the process described in i.MX Windows 10 IoT Quick Start Guide.

### 2.3.3 Common causes of build errors

1. ImportError: No module named Crypto.PublicKey.
  - This error is encountered when the `pycryptodome` module is missing or in case obsolete `pycrypto` is removed.
2. Unable to enter directory. Directory does not exist.
  - We have run into this problem in case `gitmodules` were not downloaded completely (for example, `MSRSec` is empty) or `cst` or `firmware` directories were missing. Try repeating the [Obtaining BSP sources](#) step by step.
3. The build fails in WSL while the BSP is located somewhere in `/mnt/c` of the WSL.
  - Try `setfatattr -n system.wsl_case_sensitive -v 1 <BSP_DIR>`. OP-Tee also requires symbolic links. We have been able to build boot firmware in `/mnt/c/`

on Windows OS version 1909. Workaround is to copy the BSP to WSL filesystem, for example, to HOME.

4. RuntimeError: SDE is not current. Update your env before running this tool.
  - Try to update the mono\_devel package, repeat the setup and update steps for PlatformBuild.py in [Setting up your build environment](#).
5. cp: cannot stat '../.. /firmware-imx/firmware/ddr/synopsys/lpddr4\_pmu\_train\_\*.bin'
  - Make sure the firmware-imx obtained in [Obtaining BSP sources](#) chapter is placed inside the BSP in a way the following directories exist <BSP\_DIR>/firmware-imx/firmware/ddr/synopsys
6. Cannot find the elftools module. Probably it is not installed on your system. You can install this module with apt install python3-pyelftools.
  - This error means the pyelftools module is not installed inside of the python environment. To install this package into the environment, use pip3 install pyelftools without sudo.

### 3 Revision history

Table 1. Revision history

Revision number	Date	Substantive changes
W0.9.0	1/2022	Private preview release for i.MX8M platform.
W0.9.1	3/2022	Public preview release for i.MX8M platform.
W1.0.0	4/2022	Public release for i.MX8M and i.MX8M Mini platforms.
W1.1.0	6/2022	Public release for i.MX8M Nano and i.MX8M Plus platforms.
W1.2.0	9/2022	Section 1.7 is removed

## 4 Legal information

### 4.1 Definitions

**Draft** — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

### 4.2 Disclaimers

**Limited warranty and liability** — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

**Right to make changes** — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

**Suitability for use** — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

**Applications** — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

**Terms and conditions of commercial sale** — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at <http://www.nxp.com/profile/terms>, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

**Export control** — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

**Suitability for use in non-automotive qualified products** — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

**Translations** — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

**Security** — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at [PSIRT@nxp.com](mailto:PSIRT@nxp.com)) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

### 4.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

**NXP** — wordmark and logo are trademarks of NXP B.V.

## Contents

<b>1</b>	<b>Overview .....</b>	<b>2</b>
1.1	Audience .....	2
1.2	Conventions .....	2
1.3	How to start .....	2
1.4	Using Prebuilt Binaries to create an image .....	2
1.5	Using Source Files to create image .....	2
1.6	References .....	3
<b>2</b>	<b>Building Windows 10 IoT for NXP i.MX Processors .....</b>	<b>3</b>
2.1	Obtaining BSP sources .....	3
2.1.1	Required tools .....	3
2.1.2	The NXP i.MX Windows IoT BSP source package .....	3
2.2	Building the drivers in the BSP .....	5
2.2.1	Required tools .....	5
2.2.1.1	Visual Studio 2019 .....	5
2.2.1.2	Windows Kits from Windows 10, version 2004 (10.0.19041.685) .....	5
2.2.2	Structure of Windows driver sources .....	5
2.2.3	One-time environment setup .....	6
2.2.4	Building the drivers .....	6
2.3	Building ARM64 Firmware .....	6
2.3.1	Setting up your build environment .....	6
2.3.2	Building the firmware .....	8
2.3.3	Common causes of build errors .....	9
<b>3</b>	<b>Revision history .....</b>	<b>10</b>
<b>4</b>	<b>Legal information .....</b>	<b>11</b>

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.