Manufacturing Tool V2 (MFGTool2) Workflow

Contents

[Overview 2](#_Toc337198389)

[The first thing you need to do is to tell MFGTool2 what kind of SoC it works on. 2](#_Toc337198390)

[The second thing you need to do is tell MFGTool2 where it can get ucl2.xml. 2](#_Toc337198391)

[Define the max number of devices you want to operate at the same time. 2](#_Toc337198392)

[Specify the target storage and target board. 3](#_Toc337198393)

[Prepare the binary files for operation list. 3](#_Toc337198394)

[After all this necessary information is implemented, MFGTool2 begins to work: 4](#_Toc337198395)

# Overview

This document describes the relationship between MFGTool2.exe, cfg.ini, UICfg.ini, ucl2.xml, the operation list defined in ucl2.xml, and the binary files which will be used in the burning process.

The basic function of MFGTool2 is to burn the image to a specific storage on a specific target (board) with a specific SoC.

# The first thing you need to do is to tell MFGTool2 what kind of SoC it works on.

Q: Why does MFGTool2 need this information?

A: The MFGTool2 is based on the USB, different SoCs have different PID/VID, and MFGTool2 needs this information to monitor the device tree to get the exactly right Device on which it manipulates.

Q: How to provide this information to MFGTool2?

A: By using the ucl2.xml. One ucl2.xml is used for a specific SoC. At the beginning of the ucl2.xml, a <CFG> pare term defines the SoC name and the PID/VID for it. The detailed information about <CFG> can be found in the “Manufacturing Tool V2 UCL user manual.doc”

# The second thing you need to do is tell MFGTool2 where it can get ucl2.xml.

Q: How to let MFGTool2 know where it can get the ucl2.xml?

A: By using the cfg.ini. In this cfg.ini, there is a

***[profiles]***

***chip = MX6Q Linux Update***

MFGTool2 will try to find the ucl2.xml with the path (relative path to mfgtool2.exe) Profiles\${chip}\OS Firmware. If you take the above chip value as an example, MFGTool2 will try to find the ucl2.xml at “Profiles\MX6Q Linux Update\OS Firmware”.

# Define the max number of devices you want to operate at the same time.

Q: Why does MFGTool2 need this information?

A: MFGTool2 needs this information to create the related structure, thread, and UI related resource.

Q: How to configure it?

A: By using UICfg.ini. In this file, only PortMgrDlg needs to be configured.

# Specify the target storage and target board.

Q: Why do you need this information?

A: Since different boards may have different uboot.bin file and different kernel image, the user needs to differentiate them.

Additionally, the operation on a different target and board will be different too.

Q: How to differentiate storage and board?

A: By using the operation list defined in ucl2.xml. One operation list is dedicated to defining a specific storage on a specific board.

Q: I have defined many different operation lists. How do I let MFGTool2 know which one should be used?

A: By using the cfg.ini. In this cfg.ini, there is

***[LIST]***

***name = Sabre-SD***

MFGTool2 will try to find the operation list named “Sabre-SD” in the ucl2.xml which should allocate at Profiles\${chip}\OS Firmware.

# Prepare the binary files for operation list.

Q: What kind of binary files are needed?

A: Two kinds of binary files:

1. The ram kernel and the corresponding U-Boot run on the target board.
2. The normal kernel and the corresponding U-Boot which will be used by end user on the target.

Q: Where should I put these binary files?

A: It is decided by the operation list. For example, the following is extracted from operation “Sabre-SD” defined in Profiles\MX6Q Linux Update\OS Firmware\ucl2.xml

***<CMD state="BootStrap" type="boot" body="BootStrap" file ="u-boot-mx6q-sabresd.bin" >Loading U-boot</CMD>***

***<CMD state="BootStrap" type="load" file="uImage" address="0x10800000"***

***loadSection="OTH" setSection="OTH" HasFlashHeader="FALSE" >Loading Kernel.</CMD>***

***<CMD state="Updater" type="push" body="send" file="files/u-boot-mx6q-sabresd.bin">Sending u-boot.bin</CMD>***

***<CMD state="Updater" type="push" body="send" file="files/uImage">Sending kernel uImage</CMD>***

The file name marked by red indicates where to find this file, and related path to the ucl2.xml.

Q: Which binary files are ram kernel images and which are normal images?

A: Usually, the command “BootStrap” is used to burn ram kernel.

So, in the above case, ***u-boot-mx6q-sabresd.bin, uImage*** is ram kernel and ***files/u-boot-mx6q-sabresd.bin, files/uImage*** is normal kernel.

Q: How can I generate these binaries?

A: The user needs to refer to “Mfgtool Linux or Android Firmware Development Guide V2.docx” and generate ram kernel image. How to generate normal binary is out of MFGTool scope.

# After all this necessary information is implemented, MFGTool2 begins to work:

* 1. Parse the operation list to get all the commands which need to be executed.
  2. Monitor the device tree to get the exactly right device with the PID/VID defined by <CFG> in ucl2.xml
  3. After you find right device, use the PID/VID to get the right burn phase (state). Different phases (states) have different PID/VID pares.

The detailed information about phase (state) can be found in “Manufacturing Tool V2 UCL user manual.doc” and “Manufacturing Tool V2 Quick Start Guide.doc.”

* 1. Get the commands according to the phase (state) in Step3.
  2. Execute commands



