



SWDownloader

User Manual

IFLSWD User Guide

v1.0

Not Approved by Document Control.
For Review Only.

MARVELL INTERNAL USE ONLY

DO NOT DISTRIBUTE

FOR <CUSTOMER> USE ONLY

Doc. No. MV-xxxxxxx-xx Re1.0

November 8, 2013

CONFIDENTIAL

Marvell

Document Conventions



Note: Provides related information or information of special importance.



Caution: Indicates potential damage to hardware or software, or loss of data.



Warning: Indicates a risk of personal injury.

Document Status

Doc Status: IFL SWD User Guide

Technical Publication: v1.0

This document is based on qqZhai.

For more information, visit our website at: www.marvell.com

Disclaimer

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose, without the express written permission of Marvell. Marvell retains the right to make changes to this document at any time, without notice. Marvell makes no warranty of any kind, expressed or implied, with regard to any information contained in this document, including, but not limited to, the implied warranties of merchantability or fitness for any particular purpose. Further, Marvell does not warrant the accuracy or completeness of the information, text, graphics, or other items contained within this document. Marvell products are not designed for use in life-support equipment or applications that would cause a life-threatening situation if any such products failed. Do not use Marvell products in these types of equipment or applications.

With respect to the products described herein, the user or recipient, in the absence of appropriate U.S. government authorization, agrees:

- 1) Not to re-export or release any such information consisting of technology, software or source code controlled for national security reasons by the U.S. Export Control Regulations ("EAR"), to a national of EAR Country Groups D:1 or E:2;
- 2) Not to export the direct product of such technology or such software, to EAR Country Groups D:1 or E:2, if such technology or software and direct products thereof are controlled for national security reasons by the EAR; and,
- 3) In the case of technology controlled for national security reasons under the EAR where the direct product of the technology is a complete plant or component of a plant, not to export to EAR Country Groups D:1 or E:2 the direct product of the plant or major component thereof, if such direct product is controlled for national security reasons by the EAR, or is subject to controls under the U.S. Munitions List ("USML").

At all times hereunder, the recipient of any such information agrees that they shall be deemed to have manually signed this document in connection with their receipt of any such information.

Copyright © 2013. Marvell International Ltd. All rights reserved. Marvell, the Marvell logo, Moving Forward Faster, Alaska, Fastwriter, Datacom Systems on Silicon, Libertas, Link Street, NetGX, PHYAdvantage, Prestera, Raising The Technology Bar, The Technology Within, Virtual Cable Tester, and Yukon are registered trademarks of Marvell. Ants, AnyVoltage, Discovery, DSP Switcher, Feroceon, GalNet, GalTis, Horizon, Marvell Makes It All Possible, UniMAC, and VCT are trademarks of Marvell. Intel XScale is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries. All other trademarks are the property of their respective owners.

Table of Contents

1	Purpose.....	4
1.1	Hardware Requirements	4
1.2	Software Requirements	4
1.3	Features List	4
1.4	Components	4
2	Installation	5
3	Develop downloader application	6
3.1	IFL_SWD.dll API reference	6
3.1.1	ReadBLF	6
3.1.2	MakeDownloadPackage	6
3.1.3	DownloadWithPackage	7
3.1.4	PrepareDownloadWithFBF	7
3.1.5	GenerateTimNoDSandTimHashFiles	7
3.1.6	GenerateDtimPublicKeyCompFile	8
3.1.7	BuildTimWithTimNoDSandSignedFile	9
3.1.8	GenerateDKBTimNoDSandTimHashFiles	10
3.1.9	GenerateFBFTimNoDSandTimHashFiles	11
3.1.10	MakeDownloadPackageWithAllExistImages	12
3.2	WtptpDownload.dll API reference	13
3.2.1	InitializeBL	13
3.2.2	TerminateBL	13
3.3	Parameters structure Definition	13
3.3.1	InstanceParams.....	13
3.3.2	NotifyStatus	14
3.3.3	eProcessStatus	15
3.3.4	eProcessState	15
3.4	Download Progress Display Example	16
4	Develop ReliableData tools application	18
4.1.1	RNDBuilder	18
4.1.2	RNDParser	19
4.2	Parameters structure Definition	19
4.2.1	RNDEntryParams	19
5	Appendix	21
5.1	Report issues	21
5.2	Update this document	21
5.3	Revision History	21

1 Purpose

Describe the how to develop downloader tool based on IFLSWD dlls

1.1 Hardware Requirements

Table 1.1 Hardware Requirements

Communication Ports	USB port must be available on PC
Cables and Connectors	USB cable

1.2 Software Requirements

Table 1.2 Software Requirements

Operating Systems	MS Windows XP/WIN7
Applications	Not relevant
Drivers	Marvell Wtptp driver

1.3 Features List

Table 1.3 Features Description

Feature name	Description
Full download	Download all images for full system
Partial download	Download partial images for a system
Single UE download	Download to a UE per PC one time
Multi-UE download	Download to multi-UEs parallel per PC
Upload Data from UE	Upload data from UE
Erase all Flash before download	Erase all flash before burning flash
Erase all flash only	Erase all flash without burning any things
Reset BBT	Reset BBT in burning flash

1.4 Components

Table 1.4 Components Description

Index	Dll File(s)	Header File(s)	Build environment	Description
1	IFL_SWD.dll	IFL_SWDAPI.h Macrodef.h RndParadef.h Typedef.h UploadParaDef.h	VS2008 SP1 unicode	Read blf file, process images and return image information to download in WtptpDownload.dll
2	WtptpDownload. dll	WtptpDownLoad.h/ ParaDefine.h ErrorCode.h	VS2008 SP1 unicode	Detect incoming USB and download images

2 Installation

Get SWDownloader release package and find all components in **Utilities\IFLSWD** folder.

There is a IFLSWD_Sample project to show how to call API of dlls.

3 Develop downloader application

Develop application, such as IFL Sample project.

Invoke APIs as following steps:

1. Use "MallocInstanceParams" to get the PInstanceParams to be used in "ReadBLF" or "PrepareDownloadWithFBF", or "DownloadWithPackage". This function is a must to call before call "ReadBLF" or "PrepareDownloadWithFBF" or "DownloadWithPackage"
2. Use "ReadBLF" or "PrepareDownloadWithFBF" to prepare download images Temp folder, or use "MakeDownloadPackage" to prepare zip download package
3. Use "PrepareUpload" to prepare for upload info
4. Use "InitializeBL" to initialize WtptDownload and start to waiting for user to plug in USB.
5. Show progress of downloading in a callback function which is specified in "InitializeBL" CallbackProc parameters.
6. Use "TerminateBL" to abort all download processes
7. Use "FreeInstanceParams" to free pointer new by "MallocInstanceParams"

3.1 IFL_SWD.dll API reference

3.1.1 ReadBLF

Function Name	ReadBLF
Description	Read blf file, process images and prepare download Temp folder
Parameters	PInstanceParams PInstParam, char * pszBLFfileName
Return Value	True if successful ,otherwise false
Prototype	BOOL ReadBLF(PInstanceParams PInstParam, char * pszBLFfileName);
Example	<pre> m_IPInstanceParams = MallocInstanceParams(); if(ReadBLF(m_IPInstanceParams,m_szBlfFilename)) { m_bInitSWD = true; } else { m_bInitSWD = false ; } </pre>

3.1.2 MakeDownloadPackage

Function Name	MakeDownloadPackage
Description	Parse blf file, process images and parse blf file and Prepare download zip package
Parameters	const TCHAR* pszBlfFileName
Return Value	True if successful ,otherwise false
Prototype	bool MakeDownloadPackage(const TCHAR* pszBlfFileName);
Example	<pre> if(MakeDownloadPackage (m_szBlfFilename)) { m_bInitSWD = true; } </pre>

	<pre> } else { m_bInitSWD = false ; } </pre>
--	--

3.1.3 DownloadWithPackage

Function Name	DownloadWithPackage
Description	initial PInstanceParams from Download Zip Package
Parameters	const TCHAR* pszDownloadPackagePath,PInstanceParams plnstParam
Return Value	True if successful ,otherwise false
Prototype	bool DownloadWithPackage(const TCHAR* pszDownloadPackagePath,PInstanceParams plnstParam)
Example	<pre> m_IPInstanceParams = MallocInstanceParams(); if(DownloadWithPackage(pszDownloadPackagePath,m_IPInstanceParams)) { InitializeBL(m_IPInstanceParams); } </pre>

3.1.4 PrepareDownloadWithFBF

Function Name	PrepareDownloadWithFBF
Description	Read blf file, process images and prepare download Temp folder
Parameters	PInstParam, pszBLFfileName, pszFlasherfileName, pszFBFfileName
Return Value	True if successful ,otherwise false
Prototype	BOOL PrepareDownloadWithFBF(PInstanceParams PInstParam,char * pszBLFfileName ,char* pszFlasherfileName,char *pszFBFfileName);
Example	<pre> m_IPInstanceParams = MallocInstanceParams(); if(PrepareDownloadWithFBF(m_IPInstanceParams,m_szBlfFilename,m_szFlasherFilename,m_szFBFFilename)) { m_bInitSWD = true; } else { m_bInitSWD = false ; } </pre>



Note:

This function is just for user who want to set their own path of FBF file ,flasher file and INI file , whatever , Blf file path must be set because it is necessary to generate Ntimheader file to know DDR setting and other reserved packages.

User have to generate Ntimheader file by NTimBuild.exe and then generate FBF file with FBF_Make.exe at first before calling this API

3.1.5 GenerateTimNoDSandTimHashFiles

Function Name	GenerateTimNoDSandTimHashFiles
Description	Generate Tim or DTim file without signed by private key
Parameters	const TCHAR* pszBlfFile, // blf file const TCHAR* pszpublicKeyBlfFile, const TCHAR* pszDtimPublicKeyRsaSysModulusDSFile, const TCHAR* pszDtimPublicKeyRsaExponentDSFile, TIMINCLUDEDTYPE_T eTimType, const TCHAR* pszTimNoDSFile, bool blsHSMSignature
Parameter introduction	1. pszBlfFile Decription: It is marvell blf format , in this function , user can set all key pair value to 0 as an invalid value if user put public key into 2 nd parameter pszpublicKeyBlfFile 2. pszpublicKeyBlfFile Decription: It is an optional parameter , if user set public key in blf already , this parameter don't need to set , please set it to NULL 3. pszDtimPublicKeyRsaSysModulusDSFile Description: it is a input file to set if user want to generate DTim file, user can get this file with API GenerateDtimPublicKeyCompFile 4. pszDtimPublicKeyRsaExponentDSFile Description: it is a input file to set if user want to generate DTim file, user can get this file with API GenerateDtimPublicKeyCompFile 5. eTimType Description: Tim Type , please find its definition in TIMINCLUDEDTYPE_T 6. pszTimNoDSFile Description: it is a ouput file to set
Return Value	True if successful ,otherwise false
Prototype	bool GenerateTimNoDSandTimHashFiles(const TCHAR* pszBlfFile, const TCHAR* pszpublicKeyBlfFile, const TCHAR* pszDtimPublicKeyRsaSysModulusDSFile, const TCHAR* pszDtimPublicKeyRsaExponentDSFile, TIMINCLUDEDTYPE_T eTimType, const TCHAR* pszTimNoDSFile, bool blsHSMSignature)

3.1.6 GenerateDtimPublicKeyCompFile

FunctionName	GenerateDtimPublicKeyCompFile
Description	Generate DTIM public key RSA System Modulus component file and RSA public exponent component file
Parameters	const TCHAR* pszBlfFile, const TCHAR* pszDtimPublicKeyFile, const TCHAR* pszRsaSysModulusFile, const TCHAR* pszRsaSysExponentFile
Parameter introduction	1. pszBlfFile Description: It is marvell blf format , in this function , user can set all key pair value to 0 as an invalid value if user put public key into 2 nd parameter pszDtimPublicKeyFile 2. pszDtimPublicKeyFile Decription: It is an optional parameter , if user set public key in blf already , this parameter don't need to set , please set it to NULL 3. pszRsaSysModulusFile : generated from RSA_System_Modulus of DTIM public key, it is a ouput file to set 4. pszRsaSysExponentFile : generated from RSA_Public_Exponent of DTIM public key, it is a ouput file to set

3.1.7 BuildTimWithTimNoDSandSignedFile

Function Name	BuildTimWithTimNoDSandSignedFile
Description	Generate a completed (D)TIM file from TimNoDS file and signature file of TimNoDs public key
	const TCHAR* pszBlfFile, const TCHAR* pszTimNoDSFile, const TCHAR* pszSignedFile, TIMINCLUDEDTYPE_T eTimType, bool blsHSMSTransaction const TCHAR* pszTIMHeaderFile
Parameter introduction	1. pszBlfFile Description: It is marvell blf format 2. pszTimNoDSFile Description: It is a output file of GenerateTimNoDSandTimHashFiles 3. pszSignedFile Description: It is a output signature file by HSM or others 4. eTimType Description: Tim Type , please find its definition in TIMINCLUDEDTYPE_T 5. blsHSMSTransaction Description: It is a switch , if signature file is from HSM , please set it true, otherwise set it false 6. pszTIMHeaderFile Description: It is a out file for DKB_timeHeader or FBF_timHeader , This parameter pszTIMHeaderFile must be specified if eTimType is DKB_TIM or FBF_TIM,and pszTIMHeaderFile can be ignored for other eTimType
Return Value	True if successful ,otherwise false
Prototype	bool BuildTimWithTimNoDSandSignedFile(const TCHAR* pszBlfFile, const TCHAR* pszTimNoDSFile, const TCHAR* pszSignedFile, TIMINCLUDEDTYPE_T eTimType, bool blsHSMSTransaction);

3.1.8 GenerateDKBTimNoDSandTimHashFiles

Function Name	GenerateDKBTimNoDSandTimHashFiles
Description	Generate DKB Tim file without signed by private key and DKB file
Parameters	const TCHAR* pszBlfFile, // blf file const TCHAR* pszpublicKeyBlfFile, const TCHAR* pszTimNoDSFile, const TCHAR* pszDKBFile,
Parameter introduction	1. pszBlfFile Decription: It is marvell blf format , in this function , user can set all key pair value to 0 as an invalid value if user put public key into 2 nd parameter pszpublicKeyBlfFile 2. pszpublicKeyBlfFile Decription: It is an optional parameter , if user set public key in blf already , this parameter don't need to set , please set it to NULL 3. pszTimNoDSFile Description: it is a ouput file to set 4. pszDKBFile Description: it is a ouput file to specify the generated DKB file name.
Return Value	True if successful ,otherwise false
Prototype	bool GenerateDKBTimNoDSandTimHashFiles (const TCHAR* pszBlfFile, const TCHAR* pszpublicKeyBlfFile, const TCHAR* pszTimNoDSFile, const TCHAR* pszDKBFile);

3.1.9 GenerateFBFTimNoDSandTimHashFiles

Function Name	GenerateFBFTimNoDSandTimHashFiles
Description	Generate FBF Tim file without signed by private key and all FBF file
Parameters	const TCHAR* pszBlfFile, // blf file const TCHAR* pszpublicKeyBlfFile, const TCHAR* pszTimNoDSFile, const TCHAR* pszFBFFilesFolder,
Parameters introduction	1. pszBlfFile Decription: It is marvell blf format , in this function , user can set all key pair value to 0 as an invalid value if user put public key into 2 nd parameter pszpublicKeyBlfFile 2. pszpublicKeyBlfFile Decription: It is an optional parameter , if user set public key in blf already , this parameter don't need to set , please set it to NULL 3. pszTimNoDSFile Description: it is a ouput file to set 4. pszFBFFilesFolder Description: it is a ouput folder, the generated all FBF files stored in the specified folder.
Return Value	True if successful ,otherwise false
Prototype	bool GenerateFBFTimNoDSandTimHashFiles (const TCHAR* pszBlfFile, const TCHAR* pszpublicKeyBlfFile, const TCHAR* pszTimNoDSFile, const TCHAR* pszFBFFilesFolder);

3.1.10 MakeDownloadPackageWithAllExistImages

Function Name	MakeDownloadPackageWithAllExistImages
Description	Make Download package zip to download with Marvell MPMD tool
Parameters	const TCHAR* pszBlfFileName, const TCHAR* pszDKBTim, const TCHAR* pszFBFTim, const TCHAR* pszFBFfolder, const TCHAR* pszPackageFile
Parameters introduction	1. pszBlfFile Decription: It is marvell blf format , in this function , user can set all key pair value to 0 as an invalid value if user put public key into 2 nd parameter pszpublicKeyBlfFile 2. pszDKBTim Decription: It is a file generated by BuildTimWithTimNoDSandSignedFile with DKB_TIM input type 3. pszFBFTim Decription: It is a file generated by BuildTimWithTimNoDSandSignedFile with FBF_TIM input type 4. pszFBFfolder Description: It is a folder set in GenerateFBFTimNoDSandTimHashFiles 5. PszPackageFile Description: it is a ouput Download package zip
Return Value	True if successful ,otherwise false
Prototype	bool MakeDownloadPackageWithAllExistImages(const TCHAR* pszBlfFileName, const TCHAR* pszDKBTim, const TCHAR* pszFBFTim, const TCHAR* pszFBFfolder, const TCHAR* pszPackageFile)

3.2 WtptpDownload.dll API reference

3.2.1 InitializeBL

Function Name	InitializeBL
Description	Initialize images information and waiting for USB device to download
Parameters	PInstanceParams
Return value	TRUE if successful ,otherwise FALSE
Prototype	BOOL InitializeBL(PInstanceParams pInstParam)
Example	<pre>if (InitializeBL(m_IPInstanceParams) == TRUE) { m_bInitFlag=true; }</pre>

3.2.2 TerminateBL

Function Name	TerminateBL
Description	Terminate All download processes and reset all parameters in InitializeBL
Parameters	None
Return value	TRUE if successful ,otherwise FALSE
Prototype	BOOL TerminateBL ()
Example	<pre>if (TerminateBL(m_IPInstanceParams) == TRUE) { PrintToView ("SW Downloader", "Succeeded to Initialize WtptpDownload!"); }</pre>

3.3 Parameters structure Definition

3.3.1 InstanceParams

parameter Name	Definition	Description
InstanceParams *PInstanceParams	<pre>typedef struct _InstanceParams { list<tstring>* pImagesList; //Images linked list ULONG nImgNumber; //Images Number const TCHAR* pszDKBbin;// Download file for Bootrom const TCHAR* pszDKBTim;// flasher file path const TCHAR* pszOBMFile;// OBM file path const TCHAR* pszWTMFile;// WTM file path const TCHAR* pszIMEIFile;// IMEI file path const TCHAR* pszMEPFile;// MEP file path CALLBACKPROC CallbackProc;// Callback function that will back relevant process information LPVOID UserData;//Parameter to be passed to the Callback function pWTPTPREAMBLECOMMAND pWtptpPreaCmd; // Customize Preamble Command unsigned int PlaformType; unsigned int FlashType; // 0-NAND, 1-eMMC,3- SPINOR,4-ONENAND,others-Unknown unsigned int Commands; // flags description /* bit switches */,</pre>	Initialization of WtptpDownlo ad structure

	<pre> /*bit 0:Erase all flash flag*/ /*bit 1: Upload flag */ /*bit 2: Reset BBT flag */ /*bit 3: Customized preamble flag*/ /*bit 4 ~ bit31 reserved */ unsigned int FlashPageSize; // Nand Flash Data page size unsigned int ReservedVal[MAX_RESERVED_DATA]; list<UPLOAD_DATA_SPEC*> pUploadSpecs; // Upload spec Info _InstanceParams(): nImgNumber(0),pszDKBbin(NULL),pszDKBTim(NULL),plmage sList(NULL),CallbackProc(NULL), pszOBMFile(NULL),pszWTMFile(NULL),pszIMEIFile(NULL),ps zMEPFile(NULL),UserData(NULL),pUploadSpecs(NULL),pWtp tpPreaCmd(NULL), Commands(0),FlashType(5),PlaformType(4),FlashPageSize(0) ,DownloadMode(0){ ~_InstanceParams(){ if (NULL != plimagesList) {delete plimagesList;plimagesList = NULL;} if (NULL != pszDKBbin) {delete pszDKBbin;pszDKBbin = NULL;} if (NULL != pszDKBTim) {delete pszDKBTim;pszDKBTim = NULL;} if (NULL != pszOBMFile) {delete pszOBMFile;pszOBMFile = NULL;} if (NULL != pszWTMFile) {delete pszWTMFile;pszWTMFile = NULL;} if (NULL != pszIMEIFile) {delete pszIMEIFile;pszIMEIFile = NULL;} if (NULL != pszMEPFile) {delete pszMEPFile;pszMEPFile = NULL;} if (NULL != pUploadSpecs) { for (t_UploadDataSpecIter iter = pUploadSpecs->begin();iter != pUploadSpecs- >end();++iter) { delete *iter; } delete pUploadSpecs; pUploadSpecs = NULL; } } _InstanceParams, *PInstanceParams; </pre>	
--	--	--

3.3.2 NotifyStatus

parameter Name	Definition	Description
	<pre> //Providing process evaluation information via Callback function typedef struct _NotifyStatus { int nDevice, // Active device number </pre>	Callback

NotifyStatus, *PProcInfo;	<pre> nDevType, // Device type kUnknown = 0, kBootRom = 1, kBootloader = 2 nDownloadedPercent; // Downloading percent TCHAR lpProcMsg[2000]; // Callback message TCHAR lpUSBPortAddress[260]; // USB socket address eProcessState eProcState; // Current process state eProcessStatus eProcStatus; // Current process status CTime StartDownloadTime; // start Download sys time int nErrorCode; // Errorcode in download or burning void operator = (struct _NotifyStatus& rhs) { nDevice = rhs.nDevice; nDevType = rhs.nDevType; nDownloadedPercent = rhs.nDownloadedPercent; eProcState = rhs.eProcState; eProcStatus = rhs.eProcStatus; StartDownloadTime = rhs.StartDownloadTime; nErrorCode = rhs.nErrorCode; memcpy(&lpProcMsg,&rhs.lpProcMsg,sizeof(lpProcMsg)); memcpy(&lpUSBPortAddress,&rhs.lpUSBPortAddress,siz eof(lpUSBPortAddress)); } }NotifyStatus, *PProcInfo; </pre>	information structure
------------------------------	---	--------------------------

3.3.3 eProcessStatus

parameter Name	Definition	Description
	<pre> typedef enum { stOK, // Finished OK stInvalid, // Couldn't create instance stUsbError, // Couldn't connect to the USB driver stNoReply, // Target is not responding stBadReply, // Protocol error stDownloadMError, // Downloading error stFailedToConnect, // Handshake failure/TimeOut stFailedFileCreate, // Create file failed stParseFile, // Parse file error stEmptyFileList, // Image file list is empty stFileError, // AXF file not found or file format error stFilePathError, // File path is not exist stAborted, // Aborted using abort Process() stThreadError, // Error creating the instance thread stNack, // Not acknowledge stCalculationError, // Check sum calculation error stImgConversionFail, // Error Image conversion stImgCheckOverlapFail, // Error image overlap stImgBuildFail, // Error image build stFolderError, // Work folder error stMepBndError, // Error binding Mep data stUEErrorTrace = 0xff } eProcessStatus; </pre>	Error enum

3.3.4 eProcessState

parameter	Definition	Description
-----------	------------	-------------

Name		
	<pre>typedef enum EProcessState { kProcIdle = 0, // Not active kProcInit = 1, // Initialization kProcBootRom = 2, // Initialization for BootRom kProcPrepareData = 3, // Preparation images downloading kProcConnecting = 4, // Open connection and handshake kProcDownloading = 5, // Downlodng Data kProcAborting = 6, // Aborting in process kProcFileCompleted= 7, // Filedownloading process is completed kProcCompleted = 8, // Downloading process is completed kProcUsbRemove = 9, // USB remove kProcDebugLog = 10, // debug log information kProcBurningFlash = 11, // burning flash kProcBootSecondFlasher =12, // Initialization for BootSecondFlasher = 13, // Second flasher kProcStateMax = 0xfe }eProcessState;</pre>	Download process state

3.4 Download Progress Display Example

Using callback function to display download progress and status

Example:

```
LRESULT CIFLSWD_SampleDlg::OnCallbackFunc(WPARAM wParam, LPARAM lParam)
{
    NotifyStatus* l_pNotifyStatus = (NotifyStatus*)lParam;
    CString strState, strStatus;
    BOOL bIsError = FALSE;
    SetProcState(l_pNotifyStatus);
    switch(l_pNotifyStatus->eProcState)
    {
        case kProcConnecting:
            AddDevice((NotifyStatus*)lParam);
            break;

        case kProcDownloading:
            OnProgressToView((LPARAM)l_pNotifyStatus);
            break;

        case kProcCompleted:
            OnProgressToView((LPARAM)l_pNotifyStatus);
            break;

        case kProcUploadingData:
            OnProgressToView((LPARAM)l_pNotifyStatus);
            break;

        case kProcUsbRemove:
            // To do your code
    }
```

```
        break;
    case kProcAborting:
        // To do your code Download failed
        break;
        case kProcBurningFlash:
            OnProgressToView((LPARAM)l_pNotifyStatus);
            break;
        default:
            {
                // To do your code
                break;
            }
    }
    delete l_pNotifyStatus;
    return 0;
}
```

4 Develop ReliableData tools application

Invoke APIs as following steps:

1. Use "RNDBuilder" to develop ReliabeData tool
2. Use "RNDDParser" to develop RndParser tool to unpack an exist RD.bin

4.1.1 RNDBuilder

Function Name	RNDBuilder
Description	Generate ReliableData.bin with file entries.
Parameters	RNDEntryParams * pRNDEntryParams : entry list including entry type and file name. char *pRNDFilename : filename of Reliable Data bin building
Return value	bool
Prototype	RNDBuilder(RNDEntryParams * pRNDEntryParams, char *pRNDFilename)
Example	<pre> RNDEntryParams* m_RNDEntryParams=new RNDEntryParams(); m_RNDEntryParams->TypeDetails[0].EntryTypeID=IMEI_TYPEID; m_RNDEntryParams->TypeDetails[0].nEntryFileNumber=1; m_RNDEntryParams->TypeDetails[0].FileList.push_back("IMEI.bin"); m_RNDEntryParams->TypeDetails[1].EntryTypeID=MEP_TYPEID; m_RNDEntryParams->TypeDetails[1].nEntryFileNumber=1; string str=argv[currentOption]; m_RNDEntryParams->TypeDetails[1].FileList.push_back("MEP.bin"); m_RNDEntryParams->TypeDetails[2].EntryTypeID=NVM_TYPEID; m_RNDEntryParams->TypeDetails[2].nEntryFileNumber=2; m_RNDEntryParams->TypeDetails[2].FileList.push_back("RFCal.nvm"); m_RNDEntryParams->TypeDetails[2].FileList.push_back("audiocal.nvm"); m_RNDEntryParams->nEntryTypeNumber=3; if(RNDBuilder(m_RNDEntryParams,"OPhone888CMCC.rnd")) { printf("\n"); printf("%s is Generated Successfully!\n"); } </pre>

**Note:**

```
#define TYPEID_VENDOR_BASE = 0xCAFE8000;
```

Please "#define TYPEID_BT_WLAN_MAC_DAT TYPEID_VENDOR_BASE+1" like definitions for vendor specific entry type.

If you use ACAT editable NVM file which has NVM file header, you can directly use "NVM_TYPEID" since NVM file header can be used to indentify file structure.

4.1.2 RNDParser

Function Name	RNDParser
Description	Generate ReliableData.bin with file entries.
Parameters	const TCHAR *pszRNDfileName : filename of Reliable Data bin building const TCHAR *pszDestDir: destination folder to save unpack files list<FLASH_ENTRY_INFO>& flashEntryInfo): unpack files' info , see details in FLASH_ENTRY_INFO
Return value	bool
Prototype	RNDParser(const TCHAR *pszRNDfileName, const TCHAR *pszDestDir, list<FLASH_ENTRY_INFO>& flashEntryInfo);
Example	<pre>if(!RNDParser((LPCTSTR)m_strRdFileName, (LPCTSTR)m_strParseFileDir,m_FlashEntryInfo)) { return; }</pre>

4.2 Parameters structure Definition

4.2.1 RNDEntryParams

parameter Name	Definition	Description
EntryType	<pre>typedef struct _EntryType { ULONG nEntryFileNumber; //Count of same entry type bin UINT32 EntryTypeID; //TypeID (HexFormat) list<string> FileList; //Filename list }</pre>	EntryType
RNDEntryParams	<pre>typedef struct _RNDEntryParams { ULONG nEntryTypeNumber; //Count of Entry TypeNumbers in ReliableData.bin, EntryType TypeDetails[MAXTYPENUMBER]; }RNDEntryParams;</pre>	RNDEntryParams structure

Table 3.5 Definition of parameters

parameter Name	Definition	Description
	<pre>typedef struct tag_flash_entry_info { </pre>	

FLASH_ENTRY_INFO	<pre>enum{ FILE_NAME_LEN = 116, }; unsigned long entryType; unsigned char fileName[FILE_NAME_LEN]; unsigned long entryVersion; unsigned long entryDate; unsigned long entryPayload; } FLASH_ENTRY_INFO;</pre>	
------------------	---	--

5 Appendix

5.1 Report issues

Please visit webs: <http://support.marvell.com>.

Or submit an issue at CQ web: <http://sh2-cq01.marvell.com/cqweb/login>.

5.2 Update this document

Please send updated document to qqzhai@marvell.com with “PHS CTHall document update: ...” title for merging, and you will receive merged latest version.

5.3 Revision History

Table 5.1: Revision History

Document No and Revision	Author	Description	Date
0.1	qqzhai	The initial version	06/06/2013



Marvell Semiconductor, Inc.
5488 Marvell Lane
Santa Clara, CA 95054, USA
Tel: 1.408.222.2500
Fax: 1.408.752.9028
www.marvell.com

Marvell. Moving Forward Faster