Qualcomm Technologies, Inc.

RJIL – Preparation Guideline Note for ODMs

80-P4064-1 J

September 12, 2016

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
A	December 2015	Initial release
B	December 2015	Added Sections 5.1.3 and 5.1.7, and Chapters 6 and 7
C	February 2016	Added Section 5.2.7; updated Sections 5.1 and 5.2.3, and Appendix A.1
D	March 2016	Added Section 5.2.8; updated Sections 5.1, 5.2.7, and 6.4
E	March 2016	Updated Sections 3.3, 5.1, and 5.2.6
F	May 2016	Updated Sections 3.3, 5.1, 5.2.6, and A.1
G	June 2016	Updated Sections 2.2 and 5.1
H	July 2016	Updated Section 5.1 and Chapter 7
J	September 2016	Updated Sections 5.1 and 6.2

Note: There is no Rev. I, O, Q, S, X, or Z per Mil. standards.

Contents

Figures

Tables

1 Introduction

1.1 Purpose

The document serves as a guideline for ODMs/OEMs to develop their software for Reliance Jio Infocomm Limited (RJIL). It also provides information on various processes recommended by Qualcomm Technologies, Inc. (QTI) to be followed by ODMs for a smooth lab entry at RJIL.

This document is intended for QTI's customers who target to support RJIL.

1.2 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `#include`.

Code variables appear in angle brackets, for example, `<number>`.

Commands to be entered appear in a different font, for example, **`copy a:*.* b:`**.

Button and key names appear in bold font, for example, click **Save** or press **Enter**.

If you are viewing this document using a color monitor, or if you print this document to a color printer, **red boldface** indicates code that is to be **added**, and ~~blue strikethrough~~ indicates code that is to be replaced or removed.

Shading indicates content that has been added or changed in this revision of the document.

1.3 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://createpoint.qti.qualcomm.com/.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.qualcomm.com.

2 RJIL operator and feature information

2.1 RJIL features

RJIL is a LTE-only operator with IMS core as the key technology. Currently, there is no graceful fallback support to 2G/3G systems; hence, the traditional support for CSFB is not available in RJIL. Voice services support in the RJIL network is through Voice over LTE (VoLTE).

Following are the key features that are supported/mandated by RJIL:

- IMS/VoLTE
- ViLTE (Video)
- VoWiFi
- eMBMS
- RCS

2.1.1 QTI's compliance to RJIL feature

The current version of RJIL Smartphone Requirement Specification is Ver 4.0. QTI conducted a detailed compliance study for its MSM8916/MSM8939/MSM8909 platforms to suit RJIL requirement specification Ver 3.0 (The compliance study for Ver 4.0 is still in progress). The compliance document (available at *RJIL Smartphone Specifications Compliance of QTI Devices (MSM8916 and MSM8909)* (MH80-P1185-1)) will be shared to ODM after due approval from RJIL.

2.2 Enable RJIL features for ODMs

QTI supports both IMS and eMBMS features, which have also been tested at RJIL infra.

Both IMS and eMBMS middleware are part of QTI licensed services. ODMs need to obtain the following AOST licenses from QTI:

- IMS PRO (for IMS client to support features, such as VoLTE, VT, VoWiFi, etc.)
- LTE Broadcast (eMBMS middleware – eMBMS device client software)

ODMs need to obtain the IMS PRO license from QTI.

Although eMBMS middleware is a part of AOST; and since RJIL is also a licensee, eMBMS middleware will be available to all ODMs downloadable from Google Play store along with RJIL's Broadcast Application.

The VoLTE and eMBMS RAN features are part of the baseline software.

In addition, ODMs need to build their IMS CS dialer capability into their dialer. ODMs are licensed to use QTI's QRD solution with the default integrated VoLTE/CS dialer that is part of the QRD solution.

ODMs are requested to get in touch with their respective Technical Account Managers (TAMs) and their Sales team to acquire the above licenses.

2.2.1 Optional features requirements from RJIL

Following are the additional features of QTI that RJIL may be interested:

- TrustZone (enabled by default)
- Secure Boot (Send mail to “CASS.Support@qualcomm.com”)

  To enable Secure boot, ODMs need to raise a Salesforce case and provide below details:

  - User name and email ID
  - CSMS security officer (could be the same as 1)
  - Shipping address along with the phone number and postal code
- Few Audio+, Voice+, Fluence™ HD, SD Digital Camera, and NSRM details
- SVI, STA, Assertive Display, Wi-Fi Display, Smart Camera Pack B (Ubifocus, ChromaFlash, Optimzoom, Object Cloning and Removal, Touch to track)

NOTE: The above features may be restricted to only a few set of devices. ODMs are requested to get the required clarification from TAM and RJIL.

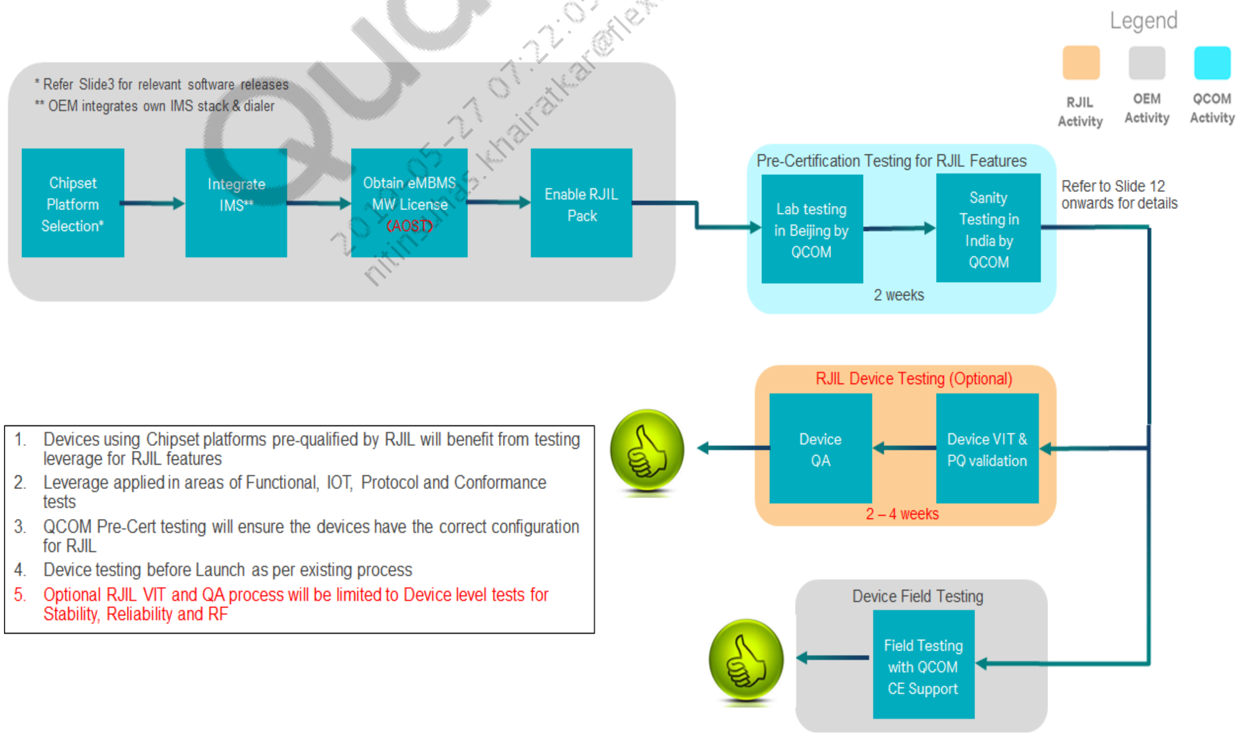
NOTE: To enable these features, ODMs are requested to contact their respective TAMs and Sales team as some of these are part of AOST.

3 Device readiness for RJIL

3.1 Device test cycle at RJIL

ODMs/OEMs supplying devices to RJIL or general open market device testing at RJIL must follow the process below:

- OEMs/ODMs must ensure that they have procured the required AOST features from QTI (IMS and eMBMS)
- Once OEMs/ODMs pick up the correct baseline corresponding to a chipset, they go through the pre-cert test process (See Section 3.2 for details.)
- After completion of the pre-cert test, the devices undergo thorough internal testing (VIT and QA) by the RJIL team


Legend
RJIL Activity
OEM Activity
QCOM Activity
* Refer Slide3 for relevant software releases
** OEM integrates own IMS stack & dialer
Chipset Platform Selection*
Integrate IMS**
Obtain eMBMS MW License (AOST)
Enable RJIL Pack
Pre-Certification Testing for RJIL Features
Lab testing in Beijing by QCOM
Sanity Testing in India by QCOM
2 weeks
Refer to Slide 12 onwards for details
RJIL Device Testing (Optional)
Device VIT & PQ validation
Device QA
2 – 4 weeks
Device Field Testing
Field Testing with QCOM CE Support

1. Devices using Chipset platforms pre-qualified by RJIL will benefit from testing leverage for RJIL features
2. Leverage applied in areas of Functional, IOT, Protocol and Conformance tests
3. QCOM Pre-Cert testing will ensure the devices have the correct configuration for RJIL
4. Device testing before Launch as per existing process
5. Optional RJIL VIT and QA process will be limited to Device level tests for Stability, Reliability and RF

Figure 3-1 RJIL device readiness process

3.2 QTI-defined pre-cert process for RJIL

QTI has established a pre-certification process that will assist ODM devices to better prepare their software prior to RJIL lab entry.

RJIL Pre-cert process execution is captured in Figure 3-2.

Following is a short overview of the pre-cert process:

1. ODMs/OEMs are recommended to select the correct baseline for RJIL.
2. ODMs/OEMs submit the below to QTI's Beijing Pre-cert lab:
   a. "OEM Device checklist" capturing details of the product with respect to hardware/software support. RF calibration tree to be shared as well. Refer to *India/RJIL Pre-Cert: OEM Checklist* (80-P4064-1) for details.
   b. Sample devices and supported accessories to undergo defined pre-cert test plan, including detailed RF verification for all supported bands. Refer to *India/RJIL Pre-Cert: MPSS Test Plan* (80-P1034-3) for details.
   c. As part of RJIL pre-cert field test, Thermal and Power test coverage has also been incorporated.
   d. RF verification is a key step in the Beijing pre-cert process
3. After Beijing pre-cert lab test, ODMs must ship two of the RF calibrated devices from Beijing lab to QTI, India to complete pre-cert field test at RJIL.
4. The above process is tracked through a Salesforce case and any issues identified are tracked with respective case as well.

NOTE: At the end of the pre-cert session, ODMs are ensured to have the right configuration and software in place to proceed further with RJIL lab test.

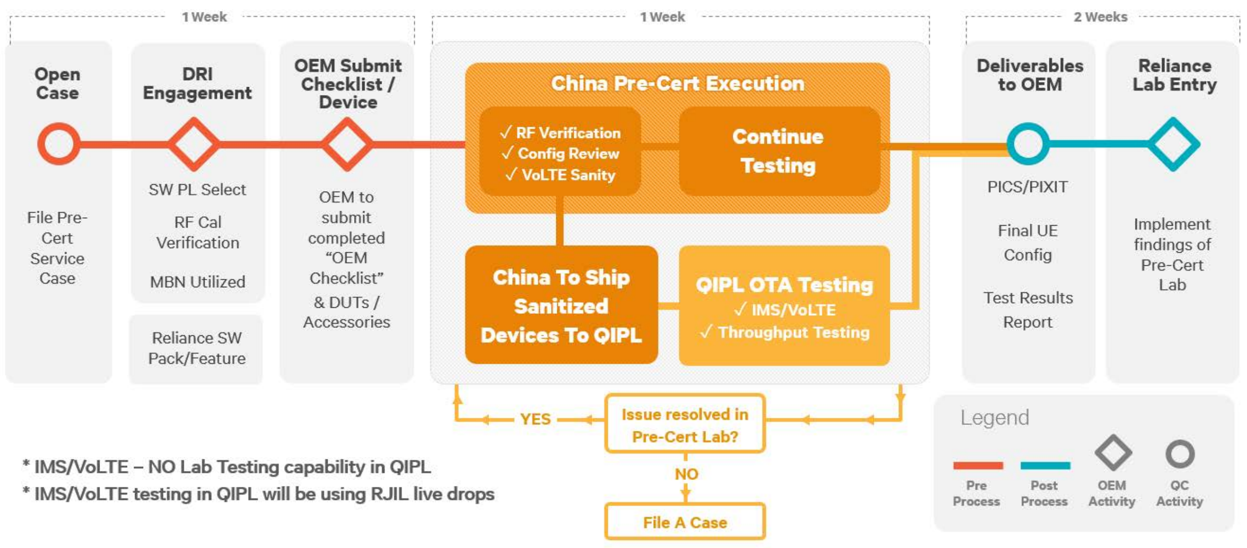


Figure 3-2 RJIL pre-cert process

3.3 Design review of ODM devices procured by RJIL

For ODM devices procured by RJIL, QTI recommends having a design review completed prior to the official lab entry. For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://createpoint.qti.qualcomm.com/.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.qualcomm.com.

Following are the common design review items for RJIL procured devices:

- Hardware schematic review
- Thermal design review
  - ODMs are requested to create a Salesforce case to get the thermal design review done by QTI
  - Both hardware and software thermal profiling/tuning are covered under the thermal design review.
- Power/current consumption analysis
  - ODMs are recommended to review QTI power dashboard provided along with the release and check the power contributions of the respective designs.
- Audio tuning
  - ODMs are recommended to avail the audio tuning facility provided by QTI.
  - ODMs are requested to book a lab slot ataudiolab.hotline.external@qti.qualcomm.com

ODMs are recommended to get camera and audio tuning done prior to RJIL lab entry.

4 RJIL operator-specific configuration

RJIL is a 4G-only operator with "**IMS**" being the key technology to support its features. There are several operator-specific configurations that need to be taken into account in the software.

All such operator specific/dependent configurations are incorporated in a separate Modem Binary (MBN) configuration file. The RJIL-specific MBN ensures to have all required NV settings for successful camping/IMS registration and access to VoLTE/VT/VoWiFi features including modem level configuration support for eMBMS feature. This avoids any manual intervention support from ODM.

- RJIL MBN file is available at:
  \modem_proc\mcfg\configs\mcfg_sw\generic\APAC\Reliance\Commercial\mcfg_sw.mbn

  ODMs must integrate this MBN into their build so that the file is picked up at run time.
- RJIL-specific modem configuration details are available at:
  modem_proc\mcfg\mcfg_gen\generic\APAC\Reliance\MCFG_SW_Items_List_Macro.xls

  The current Modem Configuration (MCFG) framework ensures that RJIL MBN is picked up (based on SIM's ICCID value) only when RJIL (or JIO) specific USIM in used in the device.
- For other non-JIO SIMs, Rest Of World (ROW) must be used. ROW MBN is available at:

  \modem_proc\mcfg\configs\mcfg_sw\generic\common\ROW\Gen_3GPP\mcfg_sw.mbn

Table 4-1 NV that are related to RJIL configuration

NV	EFS file/EFS item	Value
67218	ims/IMS_enable	1
65957	QIPCall Precondition Enabled	0 (Currently not supported in RJIL N/w)
65814/ 69749	ds_3gpp_mtu/ip6_default_mtu	1300
67264	qp_ims_reg_config	Refer to MCFG_SW_Items_List_Macro.xls sheet
71527	qp_ims_reg_config_db	Refer to MCFG_SW_Items_List_Macro.xls sheet
67257	qp_ims_voip_config	Refer to MCFG_SW_Items_List_Macro.xls sheet
70263	qp_ims_ut_config (contains config info to enable XCAP/UT)	Refer to MCFG_SW_Items_List_Macro.xls sheet
69750	qp_ims_config	
70291	qp_ims_vt_4G_media_capability	Refer to MCFG_SW_Items_List_Macro.xls sheet
73545	wifi_config	Refer to MCFG_SW_Items_List_Macro.xls sheet
Data profiles	APN settings for IMS and Emergency PDN	

5 Software baseline for RJIL

5.1 Recommended software baseline

Table 5-1 Recommended software baselines

NOTE: The following table has been updated.

Chipset/platform SP	Baseline release
MSM8939.LA.2.1.c9	M8939AAAAANLYD2109003.1 (new RJIL CPL for Android Lollipop)
MSM8939.LA.1.0.2.1	M8939AAAAANLYD102124.1 onwards
MSM8939.LA.2.1	M8939AAAAANLYD2161.1 onwards
MSM8909.LA.1.1.c7	MSM8909.LA.1.1.c7-011337128-STD.PROD-2 onwards
MSM8909.LA.1.2	MSM8909.LA.1.2-01241-STD.PROD-1 onwards
MSM8994.LA.1.3	M8994AAAAANLYD1362.1 onwards
Android Marshmallow baselines	
MSM8939.LA.2.1.5.c2	M8939AAAAANLYD2150204.1 onwards (RJIL CPL for Android M)
MSM8939.LA.2.1.5	M8939AAAAANLYD21540.1 onwards
MSM8909.LA.1.3.c2	MSM8909.LA.1.3.c2-20003-STD.PROD-1 (RJIL CPL for Android M)
MSM8909.LA.1.3	MSM8909.LA.1.3-01338-STD.PROD-1 onwards
MSM8976.LA.1.1	MSM8976.LA.1.1-00250-STD.PROD-1 onwards
MSM8953.LA.1.0	Post-CS3 release onwards (for RJIL V4.0 support)
MSM8937.LA.1.1	CS onwards
MSM8940.LA.1.1	CS onwards

NOTE: All latest VoWiFi and RJIL UI requirements (V1.8) are available only in the above list of Android Marshmallow product lines.

NOTE: RJIL UI 1.9 and V4.5 smartphone specification is current being evaluated for Android N based PLs.

5.2 Customizations for RJIL

Based on RJIL requirements and issues addressed during RJIL testing, there are some customizations that are suggested to be incorporated into the build. Refer to *RJIL Customization Requirements* (80-P3336-1) for details.

5.2.1 Enable a specific language

Follow the steps below to enable a specific language for RJIL devices. For example, Gujarati.

1. Add gu_IN to PRODUCT_LOCALES in “build/target/product/languages_full.mk” and “device/qcom/msm8916_64/msm8916_64.mk”
2. Add " $(QCPATH)/qrdplus/globalization/multi-language/res-overlay" to PRODUCT_PACKAGE_OVERLAYS in “device/qcom/msm8916_64/msm8916_64.mk”APN configuration for JIO-specific network

A complete list of all operators and their respective APNs and their IP type configuration details are part of “**apns-conf.xml**” file. The file is available at: **/LA.BR.1/vendor/qcom/proprietary/qrdplus/Extension/apps/etc/apns-conf.xml**

```
<apn carrier="Internet"
      apn="jionet"
      mcc="405"
      mnc="840"
      type="default"
      protocol="IPV4V6"
      roaming_protocol="IPV4V6"/>
```

5.2.2 NV configuration for RJIL VoLTE/VT/SMS/VoWiFi

Item	NV item number	Required value	Comments
NV 67261: DPL Parameters	PtimeValue	0	Packetization interval in milliseconds
	InitialBufferTimeValue	0	Initial buffer value in milliseconds
	AMR_Mode	0	AMR mode
	IPV6Enabled	0	1 – IPv6
	MSRPPktSz	0	Not currently used
	RUIMIMSIValue	0	0 – None, 1 – IMSI_T, 2 – IMSI_M
	DscpValue	1	Not currently used
	IMSParamSrc	2	Location to read from for IMS registration parameters: 2 – Read from UICC card
NV 67264: IMS Registration Module Parameters	RegONMode	1	1 – Service-based reg
	RegModeConfig	0	2 – IMS with non-IPSec
	regManagerPdpProfile Name		IMS PDP profile name
	RegEventPacket	1	Reg event package: 1 – Enabled, 0 – Disabled
	RegPCOEnabled	1	0 – Disabled, 1 – Enabled OTA testing, must be set as 1
	RegDHCPEnabled	0	0 – Disabled, 1 – Enabled
	RegPreConfigEnabled	0	0 – Disabled, for OTA testing
	regManagerPreConfigSer ver Base		Only used if RegPreConfigEnabled is enabled
	RegRATConfig	10	10 – LTE

Item	NV item number	Required value	Comments
	RegNVPCSCFEnabled	1	QoS profile name; currently not used
	RegWLANEnabled	0	0 – Disabled
	RegUserNameIMSI	0	Currently not used
	RegResponseforOptions	0	0 – Disabled
	RegConfigMaxDiscovery Count	0	Maximum amount of P-CSCF address discovery attempts; PCO, DHCP, and preconfig are included
	regManagerDiscovery Schedule		CSV string for minutes to wait between P-CSCF discovery attempts. This value can be left <blank> because 7200 represent the default value (if undefined). See related comment for RegConfigDelayAttempt Timer
	regManagerCDMAPdp Profile Name		eHRPD profile name
	RegConfigPdnRecovery Delay TimerVal	60	PDN delay timer value
	regManagerPDPFailure Schedule	5,5,5	CSV string for seconds to wait before next PDN establish attempt on LTE
	RegConfigMaxIntermediat ePDPRetries	1	Maximum timer for intermediate PDP retries
	RegConfigEHRPD Recovery Timer	15	Time in minutes to wait for next IMS data call bring up attempt on eHRPD if previous attempt failed
	RegConfigRegistration Attempts	3	Maximum IMS registration attempt possible on registration failure
	RegConfigDelayAttempt Timer	120	**Only if required**: Time in sec to wait for next registration attempt after registration failure. *RegConfigDelayAttempt Timer* does not impact the delay between trying the primary and secondary addresses. It determines the wait time after all PCSCFs in the list are tried. If the UE has attempted to register to every known P-CSCF and received temporary errors OR received a temporary error when attempting to establish a connection to the IMS APN, it shall calculate a wait time before attempting further IMS registrations. The timer value shall be determined using the procedures in RFC 5626 section 4.5. Base time shall be set to 120 sec and max time (regManagerDiscoverySchedule) shall be 7200 sec. When this timer expires, the UE shall attempt an IMS registration using the first P-CSCF from the list of discovered proxies

Item	NV item number	Required value	Comments
	RegConfigTestMode	0	0 – Test mode OFF; IMS registration is enabled: 1 – Test mode ON; IMS registration is disabled
	RegPCSCFPort	5060	Port number for P-CSCF
NV 67257: IMS VoIP Configuration	VoipConfigQOS	0	0 – Disable, QoS not shared
	VoipConfigDomainNotification Enable	1	1 – Enabled
	VoipConfigRTCP	1	1 – Enabled
	voipConfigAcceptContact	urn:urn-7:3gpp-service.ims.icsi.mmtel	IMS Component Service Identifier (ICSI)
	VoipConfigExpires	1800	Value of Session Expires header to be used in initial Session Expires headers (in sec)
	VoipMinSessionExpires	600	Minimum session expires time (in sec)
	VoipSessionTimerEnabled	0	0 – Disabled
	voipConfigConfURI		URI of conference server. If you keep this as blank, automatically URI will be constructed.
	VoipSilentRedialEnabled	0	0 – Disabled
	VoipConfigSessionExpires	1800	Session expires (in sec)
	VoipConfigSessionRefresher Type	0	Value of refresher parameter in Session Expires header sent: 0 – UAC (user-agent Client)
	VoipConfigSessionRefresher Method	1	Method used to refresh session: 1 – UPDATE
	VoipConfigInviteHeader		Special value included for header P-com.HDVVServiceType in INVITE
NV 67259: IMS SMS Configuration	smsConfigVDN	10138	VDN phone number
	SMSFormat	1	MO SMS format: 0 – 3GPP2, 1-3GPP
	smsAcceptContact	+g.3gpp.smsip	SMS feature tag
	smsRATMaskString	0x00000440	Bitmask specifying upon which radio access technologies to perform IMS registration. Enable LTE, WCDMA (includes HSPA), EDGE, and GPRS.
	RatMaskValue	1088	Bitmask specifying upon which radio access technologies to perform IMS registration. Enable LTE, WCDMA (includes HSPA), EDGE, and GPRS.
	PhoneContextURI		Phone context URI used only for TEL URI, not used in SIP URI

Item	NV item number	Required value	Comments
	SMSOverIPNetworkIndication	1	SMS over IP network indication: 1 – MO user SMS over IMS allowed
NV 67258: IMS Configuration	regConfigUserName		If using ISIM, then the set of parameters is read from the card
	regConfigPassword		
	regConfigPrivateURI		
	regConfigDisplayName		
	regConfigDomainName		
	regAuthSecertKey		
	3GPPEnabled		
	regConfigOPField		
NV 69744: IMS SIP Extended Configuration	Version	17	Version = 17 is needed going forward
	SipLocalPort	5060	
	TimerSipRegValue	600000	
	TimerSipSubscribeValue	600000	
	Timer_T1Value	2000	Others SIP timers are calculated based on T1, T2 and T4 respectively. Timer B has been added into NV list in MPSS.DPM.2.0/MPSS.NI.6.1 only (in that case, the version of NV 69744 should be 5 in order to be effective)
	Timer_T2Value	16000	
	Timer_T4Value	17000	
	Timer_TFValue	30000	
	Timer_TJValue	30000	
	TCPThreshholdValue	1300	TCP/UDP enabled. Set this item to "0" to disable TCP
	CompactFormEnabled	1	
	SigCompEnabled	0	
	IsSipInstanceNeeded	1	
	IPSec Integration Schema	3	To publish both Integrity algorithms.
	IPSec Encryption Algo	7	To publish all encryption algorithms (Null, AES and DES)
	AuthScheme	0	
	InitialAuthConfig	0	
	Timer_TBValue	45000	
	RouteHeaderEnabled	1	
	iTimer_Tcall	10000	
	iTimerEmergency_TD Value	0	

Item	NV item number	Required value	Comments
	iTimer_Treg	64000	
NV 67348: IMS QIPCall Config Items	Version	13	Provided as requirement to specify payload type (PT).
	EnableRTCPforActive VOIPCall	1	RTCP enable for VoLTE
	VT RTCP Reporting Interval	1	RTCP for VT.
	qipcall_rtp_nb_amr_ payload_type	0	Leave default value (shown)
	qipcall_rtp_wb_amr_ payload_type	0	Leave default value (shown)
	qipcall_rtp_nb_dtmf_ payload_type	0	Leave default value (shown)
	qipcall_rtp_wb_dtmf_ payload_type	0	Leave default value (shown)
	MediaAttribute		Leave default value (shown)
	qipcall_desired_qos_ strength	1	Specify desired QoS Strength. Value 1 indicates Local = Mandatory & Remote = Optional desired strengths
	AUDIO_CMR_AMR_NB	0	Leave default value (shown)
	AUDIO_CMR_AMR_NB	0	Leave default value (shown)
	emerg_call_cs_only	0	
	video_media_profile_ mode	3	
	VT calling enabled	1	
	mobile_data_enabled	1	
	volte_disabled	0	
	cvo_enabled	1	
	audio_feature_tag	<empty>	
	video_feature_tag	video	
	qipcall_rtp_tty_text_payload_type	111	
	qipcall_rtp_tty_red_payload_type	112	
	qipcall_rtp_tty_red_level	2	
	qipcall_rtp_tty_cps	6	
	SRTPAudioCryptoSuites	<empty>	
	SRTPVideoCryptoSuites	<empty>	
	E911_call_timer	10000	

Item	NV item number	Required value	Comments
	local_upgrade_accept_timer	150000	
	eQipcall_Conference_Subscription_Type	1	
NV 70291: IMS VT 4G Media Capability	Version	0	
	H263 preferred frame rate	0	
	H263 preferred bit rate	0	
	H263 preferred bit rate	0	
	H263 preferred profile level	0	
	H263 min frame rate	0	
	H263 max frame rate	0	
	H263 min bit rate	0	
	H263 max bit rate	0	
	H263 resolutions supported	0	
	H263 min profile levels per resolution	0	
	H263 Profile	0	
	H264 preferred frame rate	15	
	H264 preferred bit rate	384	
	H264 preferred resolution	7	
	H264 preferred profile level	4	
	H264 min frame rate	0	
	H264 max frame rate	20	
	H264 min bit rate	160	
	H264 max bit rate	384	
	H264 resolutions supported	0	
	H264 min profile levels per resolution	0	
	H264 Profile	1	
NV 71527 IMS Reg Config Db	Version	0	
	ims_rat_apn_info[0].iRAT	272	For RATs LTE/WCDMA/EDGE/GPRS – 23; for LTE only – 16

Item	NV item number	Required value	Comments
	ims_rat_apn_info[0].iAPN Type_APNindex	17	▪ For IMS APN pointing to APN 1 (ims_apn_name_db[0].cAPNName) – 17 ▪ For Internet APN pointing to APN 2 (ims_apn_name_db[1].cAPNName) – 34
	ims_rat_apn_info[0].iIMSS erviceInfo	7	VoLTE/VT/SMS – 7
	ims_rat_apn_info[0].iAuth _SecType	136	For Non-IPSec – 200 For IPSec – 136
	ims_rat_apn_info[0].iIPTy peInfo	208	V6 preferred
	ims_rat_apn_info[1].iRAT	512	WCDMA, W_HSPA, HSPA =100
	ims_rat_apn_info[1].iAPN Type_APNindex	17	
	ims_rat_apn_info[1].iIMSS erviceInfo	0	SMS/Presence – 260
	ims_rat_apn_info[1].iAuth _SecType	0	For Non-IPSec – 200 For IPSec – 136
	ims_rat_apn_info[1].iIPTy peInfo	0	IPv6 – 64 IPv4 – 32
	ims_rat_apn_info[3].iRAT	0	
	ims_rat_apn_info[3].iAPN Type_APNindex	0	
	ims_rat_apn_info[3].iIMSS erviceInfo	0	
	ims_rat_apn_info[3].iAuth _SecType	0	
	ims_rat_apn_info[3].iIPTy peInfo	0	
	ims_rat_apn_info[4].iRAT	0	
	ims_rat_apn_info[4].iAPN Type_APNindex	0	
	ims_rat_apn_info[4].iIMSS erviceInfo	0	
	ims_rat_apn_info[4].iAuth _SecType	0	
	ims_rat_apn_info[4].iIPTy peInfo	0	
	ims_rat_apn_info[5].iRAT	0	
	ims_rat_apn_info[5].iAPN Type_APNindex	0	
	ims_rat_apn_info[5].iIMSS erviceInfo	0	
	ims_rat_apn_info[5].iAuth _SecType	0	

Item	NV item number	Required value	Comments
	ims_rat_apn_info[5].iIPTypeInfo	0	
	ims_rat_apn_info[6].iRAT	0	
	ims_rat_apn_info[6].iAPNType_APNindex	0	
	ims_rat_apn_info[6].iIMSServiceInfo	0	
	ims_rat_apn_info[6].iAuth_SecType	0	
	ims_rat_apn_info[6].iIPTypeInfo	0	
	ims_rat_apn_info[7].iRAT	0	
	ims_rat_apn_info[7].iAPNType_APNindex	0	
	ims_rat_apn_info[7].iIMSServiceInfo	0	
	ims_rat_apn_info[7].iAuth_SecType	0	
	ims_rat_apn_info[7].iIPTypeInfo	0	
	ims_rat_apn_info[8].iRAT	0	
	ims_rat_apn_info[8].iAPNType_APNindex	0	
	ims_rat_apn_info[8].iIMSServiceInfo	0	
	ims_rat_apn_info[8].iAuth_SecType	0	
	ims_rat_apn_info[8].iIPTypeInfo	0	
	ims_rat_apn_info[9].iRAT	272	
	ims_rat_apn_info[9].iAPNType_APNindex	48	
	ims_rat_apn_info[9].iIMSServiceInfo	7	
	ims_rat_apn_info[9].iAuth_SecType	136	
	ims_rat_apn_info[9].iIPTypeInfo	208	
	rat_apn_fb_info[0].iRATAPNFallback	20992	LTE = 20992
	rat_apn_fb_info[0].iServicePriorityWWAN	0	Set the priority of WWAN against WLAN for each service.
	rat_apn_fb_info[1].iRATAPNFallback	37376	WCDMA = 12800, eHRPD = 16896
	rat_apn_fb_info[1].iServicePriorityWWAN	0	

Item	NV item number	Required value	Comments
	rat_apn_fb_info[2].iRATAPNFallback	0	EDGE = 8704, HSPA = 25088
	rat_apn_fb_info[2].iServicePriorityWWAN	0	
	rat_apn_fb_info[3].iRATAPNFallback	0	GPRS = 4608, W_HSPA = 29184
	rat_apn_fb_info[3].iServicePriorityWWAN	0	
	rat_apn_fb_info[4].iRATAPNFallback	0	
	rat_apn_fb_info[4].iServicePriorityWWAN	0	
	rat_apn_fb_info[5].iRATAPNFallback	0	
	rat_apn_fb_info[5].iServicePriorityWWAN	0	
	rat_apn_fb_info[6].iRATAPNFallback	0	
	rat_apn_fb_info[6].iServicePriorityWWAN	0	
	rat_apn_fb_info[7].iRATAPNFallback	0	
	rat_apn_fb_info[7].iServicePriorityWWAN	0	
	rat_apn_fb_info[8].iRATAPNFallback	0	
	rat_apn_fb_info[8].iServicePriorityWWAN	0	
	rat_apn_fb_info[9].iRATAPNFallback	0	
	rat_apn_fb_info[9].iServicePriorityWWAN	0	
	iAllowedIMSSrvOnWLAN	2055	
	bAddAllFTs	0	Currently not used
	iACSPriority	0	
	iISIMPriority	2	
	iNVPriority	3	
	iPCOPriority	1	
	iIMSServiceStatus	32775	All services supported by device Non RCS devise - 32775
	ims_apn_name_db[0].cAPNName	ims	
	ims_apn_name_db[1].cAPNName		

Item	NV item number	Required value	Comments
71597: QIPCALL QOS Reservation Timer	QIPCALL QOS Reservation Timer	8000	
NV 70263 IMS UT Specific Config			
	Version	10	
	utAPNName	jionet	
	utDomainName	7077	
	utPAssociatedURI		
	utApplicationUID	simservs.ngn.etsi.org	
	iUtRatConfig	10	
	iUtIPAddrType	1	
	iUtRetryTimerValue	0	
	iUtRetryAttemptCount	0	
	iUtAPNType	5	
	eUtGBAUbType	0	
	eUtGBAUbMode	2	
	utBSFAddr	7080	
	iDisableUt	0	
	eUtGBATLSMode	0	
	iUtPDNHysTimerValue	0	
	iRatMaskValue	0	
	iUtIPAddrType_APN2		
	utAPNName_APN2		
	eUtUbTlsSupport		
	eUtMediaElementSupport	1	
	eUtEmptySIBUsage	2	
	reserved		
NV 70235 qp_ims_reg_extended_0_config	Version	5	
	RegConfigNetworkInitiatedDeRegTimer	60	
	Tdelay	0	
	iEmerIPFallback	0	
	RegConfigPdnRecoveryImmediateTimer	0	
	iRegRetryBaseTime	0	
	iRegRetryMaxTime	0	
	eEnableRegInLPM	2	
NV 73545 MMODE Wi-Fi	cmph_wifi_config_s_type.version	0	

Item	NV item number	Required value	Comments
	cmph_wifi_config_s_type.is_wifi_only_mode_for_voice	1	
	cmph_wifi_config_s_type.is_e911_over_wifi	1	
	cmph_wifi_config_s_type.wifi_cs_scan_timer	5	
	cmph_wifi_config_s_type.reserved[0]	0	
	cmph_wifi_config_s_type.reserved[1]	0	
	cmph_wifi_config_s_type.reserved[2]	0	
	cmph_wifi_config_s_type.reserved[3]	0	
	cmph_wifi_config_s_type.reserved[4]	0	
	cmph_wifi_config_s_type.reserved[5]	0	
	cmph_wifi_config_s_type.reserved[6]	0	
NV 73713	Version	3	

Item	NV item number	Required value	Comments
IMS handover config	ims_ho_hyst_timer_ePDG_LTE	0	
	ims_ho_hyst_timer_ePDG_1X	0	
	ims_ho_hyst_timer_ePDG_WiFi	0	
	ims_ho_enabled	0	
	ims_drvcc_enable	0	
	ims_drvcc_roaming_enabled	0	
	ims_drvcc_backoff_timer_max	0	
	ims_drvcc_backoff_timer_min	0	
	ims_stn		
	ims_ho_lte_qual_th1	0	
	ims_ho_lte_qual_th2	0	
	ims_ho_lte_qual_th3	0	
	ims_ho_1x_qual_th	0	
	ims_ho_wifi_qual_thA	0	
	ims_ho_wifi_qual_thB	0	
	ims_ho_wifi_connectivity_backoff_timer	10	
	ims_ho_midcall_connectivity_timer	5	
	ims_ho_RTT_threshold	3000	
	ReservedBytes		

5.2.3 Create profiles for internet APN, IMS, APN and emergency APN

Following three APNs exist during the IMS registration.

Internet APN

This is used as the default APN. For example, the default APN for RJIL is *jionet or NULL*. The UE attaches to the default APN and initiates the default bearer for the data service.

IMS APN

After the Internet APN connection, the UE initiates the PDN connection to the IMS APN to activate the default bearer QCI = 5 for the IMS SIP MSG. APN name for IMS profile is “ims”.

Emergency APN

This is used for emergency calls over IMS. Even though emergency calls over IMS support is not present at the network side, RJIL mandates to trigger the emergency PDN and then falls back to CS while initiating the emergency calls.

Two profiles are created for VoLTE by QMICM, the first one is for the Internet APN and the second is for the IMS APN. The MTP attaches to the Internet APN and then initiates the PDN connect request to the IMS APN. After a successful IMS APN connection and QCI = 5 bearer activation, the MTP sends IMS SIP Register Message to the P-CSCF.

5.2.4 Enable PCO by AT commands for P-CSCF address

IMS APN

`AT$QCPDPIMSCFGE=2,1,0,0` (Set PCO for profile 2)

Emergency APN

`AT$QCPDPIMSCFGE=3,1,0,0` (Set PCO for profile 3)

5.2.5 IMPI-based ePDG discovery

To meet the requirements mentioned in Section 5.2.4, configure the following settings in iwlan_s2b_config.txt file (file present in \modem_proc\mcfg\mcfg_gen\scripts\data\efs_files\rel\).

```
epdg_fqdn:vowifi.jio.com;
static_fqdn_enabled:FALSE;
epdg_plmn_list:<plmn>;
ke_payload_enabled:FALSE;
pcscfv4_attr_type_val:16384;
ikev2_sa_rekey_timer_soft_sec:86400;
ikev2_sa_rekey_timer_hard_sec:86500;
esp_rekey_timer_soft_sec:86400;
esp_rekey_timer_hard_sec:86500;
natt_keep_alive_timer_sec:20;
epdg_fqdn_impi_based:TRUE;
ikev2_self_id_type:ID_RFC822_ADDR_MAC_IMPI_BASED;
```

PLMN format is MCCMNC – For example, if PLMN is updated as 123456, then 123 will be the MCC and 456 will be the MNC; and for PLMN 12345, 123 will be the MCC and 045 will be the MNCr example,.

- Provision the above ePDG PLMN list with inter-circle PLMN list only and do not provision the EHPLMN list.
- When the registered PLMN is one of the inter-circle PLMNs present in 'epdg_plmn_list', FQDN is constructed with this registered PLMN.
- When the registered PLMN is not one of the PLMNs (outside inter-circle roaming) present in 'epdg_plmn_list', there would be no match for this RPLMN against the ePDG PLMN list, hence it would use the HPLMN to construct the FQDN.

- If all the above fail, then FQDN falls back to static FQDN provisioned in iwlan_s2b_config.txt file.
- OEMs need to configure static ePDG FQDN in iwlan_s2b_config.txt file as mentioned above. OEMs need to work with RJIL to get the VPMN list to be filled in iwlan_s2_config.txt file as value for epdg_plmn_list.
- VPLMN based FQDN construction for which we configure ePDG PLMN list parameter is supported only from MPSS.JO.1.0, MPSS.BO.2.5 product lines.
- MPSS.DPM.2.0.2.c1 supports only FQDN construction based on MCC and MNC of the card. VPLMN support is not present on DPM.
- epdg_fqdn_impi_based field needs to be set to TRUE and ikev2_self_id_type to be set as ID_RFC822_ADDR_MAC_IMPI_BASED for IMPI-based ePDG discovery functionality to kick-in.
- ke_payload_enabled field determines if the KE payload should be sent during UE-initiated ESP SA rekey and it has to be set to “false” for RJIL.

5.2.6 Enable few features for RJIL

- ADB property to enable VoWiFi

```
adb shell setprop persist.data.iwlan.enable true
```

- ADB property to enable Video Telephony (VT)

```
adb setprop persist.radio.NO_STAPA 1
```

- ADB properties to be set for Primary card feature (DSDS – LTE SIM preference feature)

```
persist.radio.detect4gcard=true
persist.radio.primarycard=true
ro.telephony.default_network=9
```

NOTE: Refer to *RJIL Customization Requirements* (80-P3336-1) for a complete list of recommended customizations for RJIL.

5.2.7 User agent in IMS message for RJIL

RJIL has a specific requirement to include user agent in a specific format “Vendorname_Modelnumber_Version”. This can be done by including the relevant string in NV 69689. QTI recommends to check with RJIL for the exact format.

6 Load multi-MBN files into device

This chapter provides an overview of multi-MBN loading using Golden EFS method and MBN-OTA approach. It also provides an overview of SSR enabling and Modem Configuration (MCFG) related NVs.

6.1 Load multiple MBNs during factory production

6.1.1 System requirements

- Windows 7 64-bit operating system
- QPST ver 2.7.420 or later to download MBN; QPST MCFG_PDC tool must find the USB port
- Qualcomm Technologies, Inc. (QTI) USB driver USB_WWAN_WINDOWS (new series) ver 1.0025 or later

6.1.2 Installers

NOTE: Make sure the following installers are installed in your system before starting installation.

- Download Perl and Python installers.
  - ActivePerl_5.16.2.3010812913.msi – http://dlsw.baidu.com/sw-search-sp/soft/4a/14792/ActivePerl_5.16.2.3010812913.msi
  - Python-2.7.6.amd64.msi − https://www.python.org/download/releases/2.7.6/
  - Select the Windows x86-64 MSI installer (2.7.6)
- Install OpenSSL with version 0.9.8y or later. Review the terms and conditions mentioned in the https://www.openssl.org/ before installing OpenSSL.
  a. Install vcredist_x64.exe
  b. Install Win64OpenSSL-0_9_8y.exe
  c. Set the environmental variable to the installed path “C:\openssl\bin”

6.1.3 Generate golden EFS with multiple MBNs

For detailed procedure with screenshots on Golden EFS generation, refer to *Creating MBN and Golden EFS Builds* (80-NU184-1).

6.1.3.1 Scripts/Files to generate golden EFS

The scripts used in golden EFS generation are available in the following paths of modem source code.

a. Copy all these files and place them under single folder in your local machine.

b. Execute all the commands in command prompt from this path.

```
efsreadimage.pl.
..\modem_proc\core\storage\tools

QDSTMBN.py
..\ modem_proc\core\storage\tools\qdst

Efs_image_create.py.
..\modem_proc\core\bsp\efs_image_header\tools

efs_image_meta.bin
..\modem_proc\core\bsp\efs_image_header\build\efs_image_header\qdsp6\
EAAAANVZ
```

6.1.3.2 Disable the security

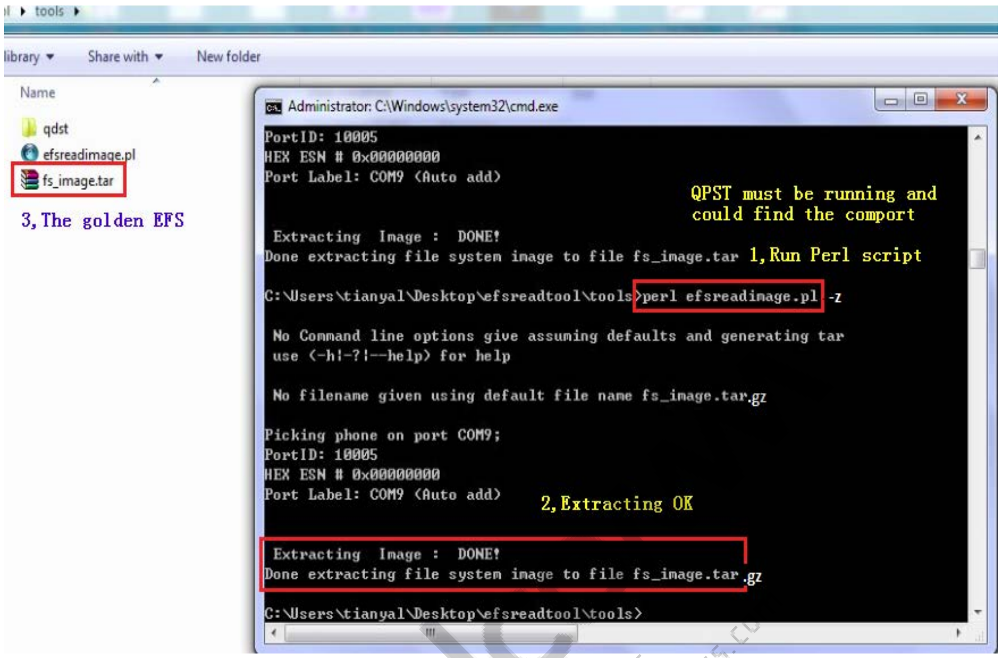
For taking EFS backup from the device, prepare a build with feature flag FEATURE_EFS_ENABLE_FACTORY_IMAGE_SECURITY_HOLE macro defined in custfaaaanuaq.h header file.

6.1.3.3 Take EFS backup image

1. Load the software prepared in above step with FEATURE_EFS_ENABLE_FACTORY_IMAGE_SECURITY_HOLE flag enabled.
2. Using PDC tool, load multiple software MBNs and hardware MBN that are required for your device.
3. If you are not using the hardware MBN, then update all the NV/EFS files you want to change before taking backup of the EFS from device.
4. Take EFS backup by typing the below command in command prompt:

   ```
   perl efsreadimage.pl –z
   ```

   The fs_image.tar.gz file is created.

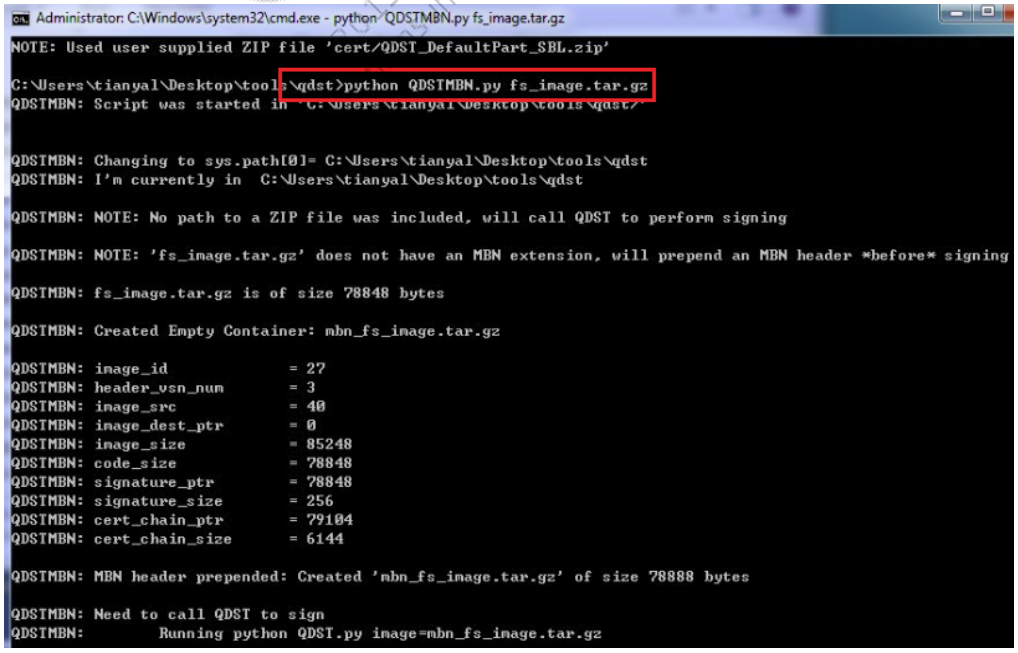


```
PortID: 10005
HEX ESN # 0x00000000
Port Label: COM9 (Auto add)

 Extracting  Image :  DONE!
Done extracting file system image to file fs_image.tar

C:\Users\tianyal\Desktop\efsreadtool\tools>perl efsreadimage.pl -z

 No Command line options give assuming defaults and generating tar
 use (-h|-?|--help) for help

 No filename given using default file name fs_image.tar.gz

Picking phone on port COM9;
PortID: 10005
HEX ESN # 0x00000000
Port Label: COM9 (Auto add)

 Extracting  Image :  DONE!
Done extracting file system image to file fs_image.tar.gz

C:\Users\tianyal\Desktop\efsreadtool\tools>
```

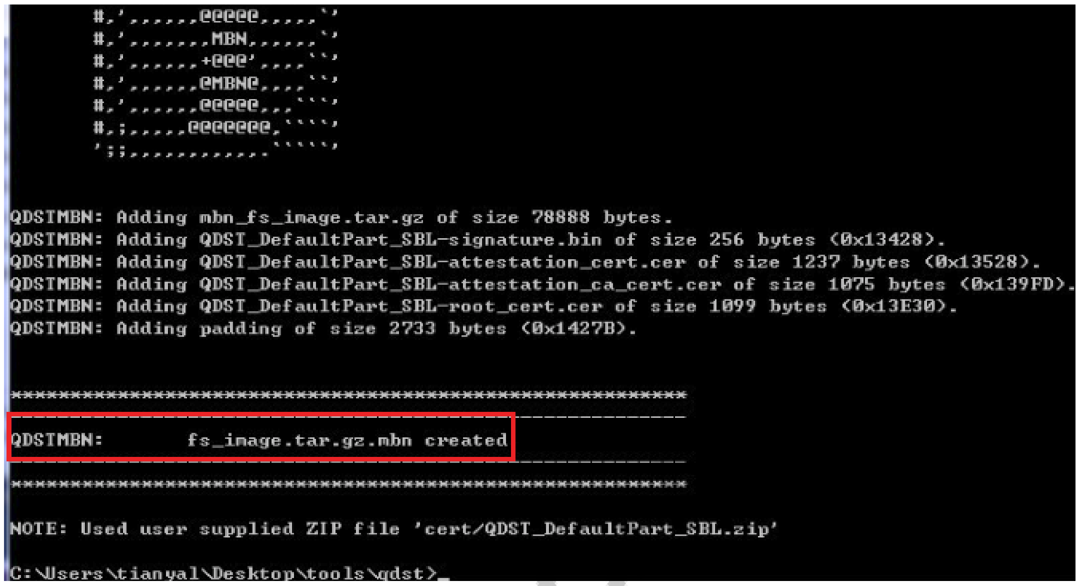
5. Sign the image using the following command:

   ```
   python QDSTMBN.py fs_image.tar.gz
   ```

   The fs_image.tar.gz.mbn file is created.


Administrator: C:\Windows\system32\cmd.exe - python QDSTMBN.py fs_image.tar.gz

```
NOTE: Used user supplied ZIP file 'cert/QDST_DefaultPart_SBL.zip'

C:\Users\tianyal\Desktop\tools\qdst>python QDSTMBN.py fs_image.tar.gz
QDSTMBN: Script was started in 'C:\Users\tianyal\Desktop\tools\qdst\'


QDSTMBN: Changing to sys.path[0]= C:\Users\tianyal\Desktop\tools\qdst
QDSTMBN: I'm currently in  C:\Users\tianyal\Desktop\tools\qdst

QDSTMBN: NOTE: No path to a ZIP file was included, will call QDST to perform signing

QDSTMBN: NOTE: 'fs_image.tar.gz' does not have an MBN extension, will prepend an MBN header *before* signing

QDSTMBN: fs_image.tar.gz is of size 78848 bytes

QDSTMBN: Created Empty Container: mbn_fs_image.tar.gz

QDSTMBN: image_id            = 27
QDSTMBN: header_vsn_num      = 3
QDSTMBN: image_src           = 40
QDSTMBN: image_dest_ptr      = 0
QDSTMBN: image_size          = 85248
QDSTMBN: code_size           = 78848
QDSTMBN: signature_ptr       = 78848
QDSTMBN: signature_size      = 256
QDSTMBN: cert_chain_ptr      = 79104
QDSTMBN: cert_chain_size     = 6144

QDSTMBN: MBN header prepended: Created 'mbn_fs_image.tar.gz' of size 78888 bytes

QDSTMBN: Need to call QDST to sign
QDSTMBN:        Running python QDST.py image=mbn_fs_image.tar.gz
```

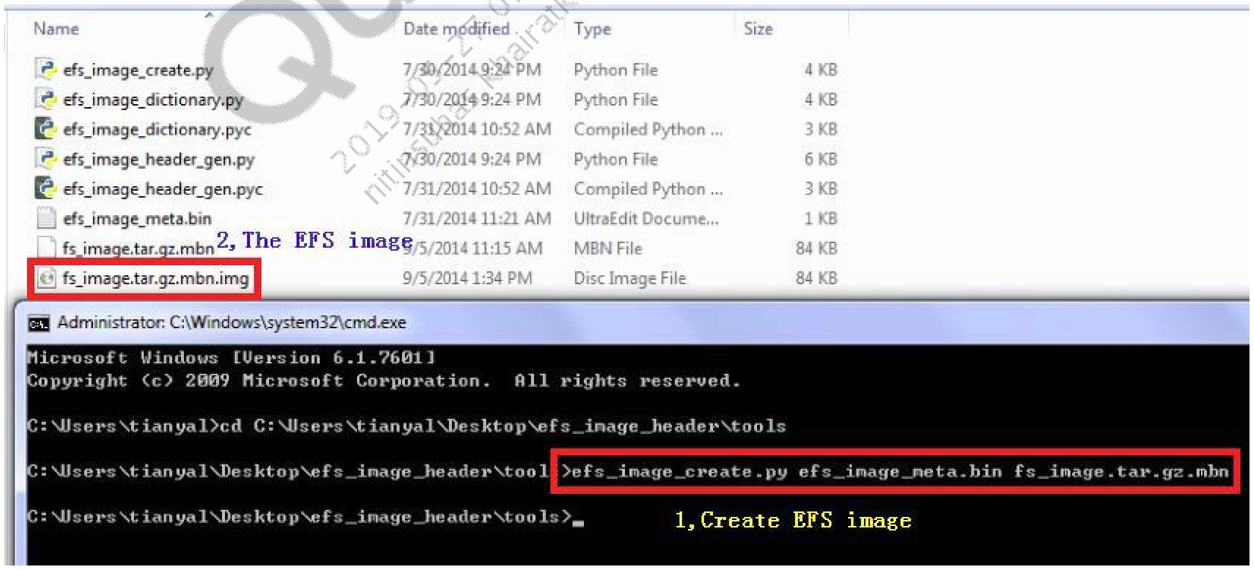


```
        #,',,,,,,,@@@@@,,,,,`'
        #,',,,,,,,,MBN,,,,,,`'
        #,',,,,,,,+@@@',,,,``'
        #,',,,,,,,@MBN@,,,,``'
        #,',,,,,,,@@@@@,,,```'
        #,;,,,,,,@@@@@@@,````'
        ';;,,,,,,,,,,,,`````'


QDSTMBN: Adding mbn_fs_image.tar.gz of size 78888 bytes.
QDSTMBN: Adding QDST_DefaultPart_SBL-signature.bin of size 256 bytes (0x13428).
QDSTMBN: Adding QDST_DefaultPart_SBL-attestation_cert.cer of size 1237 bytes (0x13528).
QDSTMBN: Adding QDST_DefaultPart_SBL-attestation_ca_cert.cer of size 1075 bytes (0x139FD).
QDSTMBN: Adding QDST_DefaultPart_SBL-root_cert.cer of size 1099 bytes (0x13E30).
QDSTMBN: Adding padding of size 2733 bytes (0x1427B).


**********************************************************
----------------------------------------------------------
QDSTMBN:      fs_image.tar.gz.mbn created
----------------------------------------------------------
**********************************************************

NOTE: Used user supplied ZIP file 'cert/QDST_DefaultPart_SBL.zip'

C:\Users\tianyal\Desktop\tools\qdst>_
```

6. Create EFS image for the target device using the command below:

   ```
   efs_image_create.py efs_image_meta.bin fs_image.tar.gz.mbn
   ```

   The fs_image.tar.gz.mbn.img file is created.



Name	Date modified	Type	Size
efs_image_create.py	7/30/2014 9:24 PM	Python File	4 KB
efs_image_dictionary.py	7/30/2014 9:24 PM	Python File	4 KB
efs_image_dictionary.pyc	7/31/2014 10:52 AM	Compiled Python ...	3 KB
efs_image_header_gen.py	7/30/2014 9:24 PM	Python File	6 KB
efs_image_header_gen.pyc	7/31/2014 10:52 AM	Compiled Python ...	3 KB
efs_image_meta.bin	7/31/2014 11:21 AM	UltraEdit Docume...	1 KB
fs_image.tar.gz.mbn	9/5/2014 11:15 AM	MBN File	84 KB
fs_image.tar.gz.mbn.img	9/5/2014 1:34 PM	Disc Image File	84 KB

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\tianyal>cd C:\Users\tianyal\Desktop\efs_image_header\tools

C:\Users\tianyal\Desktop\efs_image_header\tools>efs_image_create.py efs_image_meta.bin fs_image.tar.gz.mbn

C:\Users\tianyal\Desktop\efs_image_header\tools>_
```

6.1.3.4 Erase modemst1 and modest2

```
fastboot flash modemst1 C:\<temp_folder>\zero.bin
fastboot flash modemst2 C:\ <temp_folder>\zero.bin
```

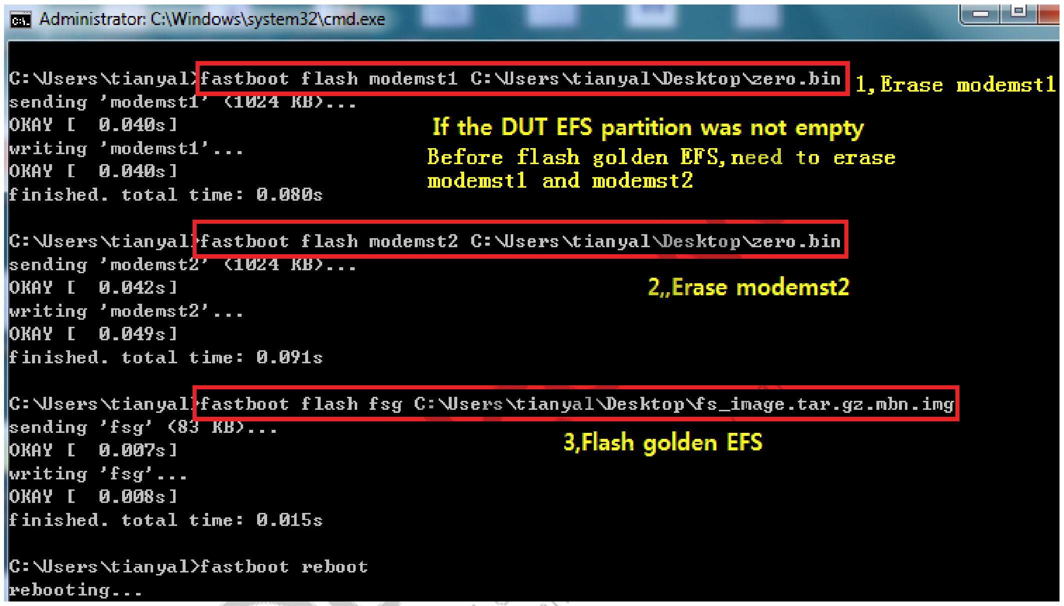
Generate zero.bin using the command below:

```
dd if=/dev/zero of=./single_factory_image/zero.bin bs=1024 count=1536
```

6.1.3.5 Flash Golden EFS

Run the following command to flash Golden EFS:

```
fastboot flash fsg fs_image.tar.gz.mbn.img
```



Administrator: C:\Windows\system32\cmd.exe

```
C:\Users\tianyal>fastboot flash modemst1 C:\Users\tianyal\Desktop\zero.bin
sending 'modemst1' (1024 KB)...
OKAY [  0.040s]
writing 'modemst1'...
OKAY [  0.040s]
finished. total time: 0.080s

C:\Users\tianyal>fastboot flash modemst2 C:\Users\tianyal\Desktop\zero.bin
sending 'modemst2' (1024 KB)...
OKAY [  0.042s]
writing 'modemst2'...
OKAY [  0.049s]
finished. total time: 0.091s

C:\Users\tianyal>fastboot flash fsg C:\Users\tianyal\Desktop\fs_image.tar.gz.mbn.img
sending 'fsg' (83 KB)...
OKAY [  0.007s]
writing 'fsg'...
OKAY [  0.008s]
finished. total time: 0.015s

C:\Users\tianyal>fastboot reboot
rebooting...
```

1,Erase modemst1

If the DUT EFS partition was not empty
Before flash golden EFS,need to erase
modemst1 and modemst2

2,,Erase modemst2

3,Flash golden EFS

6.1.4 Program the above image to eMMC

- Copy the fs_image.tar.gz.mbn.img to the download directory.
- Modify the following line rawprogram0.xml present in the metabuild path:

```
<program SECTOR_SIZE_IN_BYTES="512" file_sector_offset="0" filename="
fs_image.tar.gz.mbn.img" label="fsg" num_partition_sectors="3072" ph
ysical_partition_number="0" size_in_KB="1536.0"sparse="false" start_
byte_hex="0xc008000" start_sector="393280"/>
```

6.2 OTA update for MBNs

This OTA method applies to MPSS.DPM/MPSS.JO only (for later PLs this is deprecated and it is included as a part of NHLOS.bin).

Set the following android property to enable MBN OTA feature:

```
setprop persist.raido.start_ota_daemon 1
```

6.2.1 Check device support for MBN OTA update

If your file “\android\device\qcom\common\rootdir\etc\init.qcom.sh” has the following lines, then your device supports the MBN OTA update.

```
“rm -rf /data/misc/radio/modem_config
+mkdir /data/misc/radio/modem_config
+chmod 770 /data/misc/radio/modem_config
+cp -r /firmware/image/modem_pr/mbn_ota/* /data/misc/radio/modem_config
+chown -hR radio.radio /data/misc/radio/modem_config
+echo 1 > /data/misc/radio/copy_complete”
```

6.2.2 Create mbn_ota folder

1. Create mbn_ota folder in the local path as given below and place the MBNs here. The mbn_ota is bundled in MPSS.

   MPSS.TA.2.0.r2\Main\modem_proc\mbn_ota
2. After flash, copy the MPSS to /firmware/image/modem_pr/mbn_ota/*.
3. The implemented MBN OTA functions take care of auto-loading the MBNs by selecting the correct MBN.

6.3 Important NV/EFS items

- Device mode: NV 70266 – /nv/item_files/modem/mmode/device_mode

  The NV indicates whether the device has a single or dual SIM card.

  [SYS_MODEM_DEVICE_MODE_SINGLE_SIM = 0

  SYS_MODEM_DEVICE_MODE_DUAL_SIM_DUAL_STANDBY = 1]
  - For multi-SIM card devices the value of this EFS value must be set to 1.
  - For single-SIM card devices the value must be set to 0.
- Auto MBN loading: NV 71546 - /nv/item_files/mcfg/mcfg_autoselect_by_uim

  Auto loading of MBN is based on the value of this NV.
  - 0 means auto–MBN loading is disabled
  - Generally, the recommended value is 3(0x3)/7(0x7) for devices in which the auto-MBN loading is enabled.

This EFS value should be set as part of factory default setting or as part of hardware MBN.

Enabling SSR

Whenever the MBN reloads, the device crashes if the SSR is not enabled for modem. If the SSR is enabled, the device does not crash during MBN loading/Segment loading. During the testing phase of the device, the SSR should be enabled.

Follow the steps below to enable SSR for modem:

```
echo related > /sys/bus/msm_subsys/devices/subsys0/restart_level
```

Note that RAM dumps must be disabled for the commercial product. This can be done using the following command:

```
echo 0> /sys/module/subsystem_restart/parameters/enable_ramdumps

#after reboot, double check the parameters are set permanently
adb shell cat /sys/bus/msm_subsys/devices/subsys0/restart_level
adb shell cat /sys/module/subsystem_restart/parameters/enable_ramdumps
```

6.4 Types of MBNs

- Operator specific MBN:

  Reliance MBN means Reliance JIO operator. This is loaded only when Reliance JIO SIM card is inserted. Reliance JIO MBN has all the IIN list (ICCID) corresponding to Reliance JIO PLMNs.

- Open market MBN

  ROW MBN means Rest Of World MBN. This MBN is loaded if there is no explicit MBN matching found for the inserted SIM card.

7 Logging/debug procedure in the software

This section provides steps/commands to collect logs/information for various modem issues.

7.1 Enable debug codes

Ensure all debug codes are enabled in necessary files. Enable DBG and VBDG in the following files:

- ConnectivityService.java
- Tethering.java
- DcTracker.java
- DcTrackerBase.java
- CallTracker.java
- CallConnector.java
- GsmCallTracker.java
- NatController.java
- TetherController.java
- NatController.java

For example, change the following DBG and VDBG flags to "true":

```
private static final boolean DBG = ~~false~~ true;
private static final boolean VDBG = ~~false~~ true;
```

7.2 Device configuration to enable additional debug codes

Add the below lines in /data/local.prop (create new file, if not already available)

```
log.tag.TelecomFramework=VERBOSE
log.tag.InCall=VERBOSE
log.tag.Telecom=VERBOSE
log.tag.Telephony=VERBOSE

adb push local.prop /data/local.prop
adb shell chmod 644 /data/local.prop
adb shell chown root /data/local.prop
adb shell sync
adb reboot
```

7.3 Device configuration to collect optimized logs for IMS Video Telephony issues

Execute the below commands before capturing logs for IMS VT issues:

```
adb root
adb wait-for-device
adb remount
adb shell setprop persist.ims.disableADBLogs    2
adb shell setprop persist.ims.disableDebugLogs 0
adb shell setprop persist.ims.disableQXDMLogs 0
adb shell setprop persist.ims.disableIMSLogs     0
adb shell setprop persist.camera.cpp.debug.mask    0
adb shell setprop persist.camera.hal.debug.mask     0
adb shell setprop persist.camera.hal.debug              0
adb shell setprop persist.camera.ISP.debug.mask    0
adb shell setprop persist.camera.pproc.debug.mask  0
adb shell setprop persist.camera.stats.debug            0
adb shell setprop persit.camera.imglib.logs               1
adb shell setprop persist.camera.mct.debug.mask    1
adb shell setprop persist.camera.sensor.debug         0
adb shell setprop persist.camera.global.debug   0
adb shell setprop vidc.debug.level 1
adb reboot
adb wait-for-device
adb shell setprop log.tag.TelecomFramework VERBOSE
db shell setprop log.tag.Telecom VERBOSE
adb shell setprop log.tag.Telephony VERBOSE
adb shell setprop log.tag.InCall VERBOSE
DO NOT REBOOT THE DEVICE AFTER THIS.
```

7.4 Logging commands/procedures

This section covers general logging needed for key use-cases.

7.4.1 Regular/general issues

For general issues like voice call/registration failures/icon display issues etc., following logs are needed.

7.4.1.1 ADB logs

- Radio: `adb wait-for-device & adb shell logcat -v time > logcat.txt`
- Main: `adb wait-for-device & adb shell logcat -v time -b radio > logcat_radio.txt`
- Dump state: `adb wait-for-device & adb shell dumpstate > dumps.txt`

- Kernel: `adb wait-for-device & adb shell cat /proc/kmsg > dmsg.txt`
- Event: `adb wait-for-device & adb shell logcat –v time –b events > logcat_event.txt`
- Bug Report (When issue is observed): `adb shell bugreport > bugr.txt`

Set the ADB property: `persist.radio.adb_log_on to 0 in ADB Shell`

Use the following command to collect all logs in one file

```
adb wait-for-device & adb shell logcat -b main -b system -b radio -b events
-v threadtime > logcat.txt
```

ADB properties

```
adb shell getprop > android_getprop.txt
```

Telephony databases

```
/data/data/com.android.providers.telephony/databases/mmssms.db
/data/data/com.android.providers.settings/databases/settings.db
/data/data/com.android.providers.telephony/databases/telephony.db
```

NOTE:

- Perform the following steps where issues such as VT call and other test scenarios having timestamps and snapshot of UI are required:
  a. Record a video and explain the issue being observed.
  b. Take a snapshot at the time the issue was observed.
  c. The timestamp at which the issue was observed.
- Issues where you compare a REF device (benchmarking issues) – Use the logs/build information.
  - Use `ps | grep [processname]` to check whether a particular process is active/inactive.
- Ensure to include this information when raising a Salesforce case. This will help QTI to provide effective support during case analysis.

7.4.2 Crash issues

For crash issues, in addition to logs mentioned in 7.4.1 , below logs are needed:

adb pull /data/anr/traces.txt

adb pull /data/tombstones

7.4.3 Memory issues

For memory issues, in addition to logs mentioned in 7.4.1 , below logs are needed:

adb shell procrank > procrank.txt

7.5 For iWLAN, VoWiFi, IMS registration, and MMS issues

In addition to the logs mentioned for regular issues, the following are required:

TCPDUMP command

- For collection on specific interfaces:

```
adb shell tcpdump -i wlan0 -w /data/wlan.pcap
```

- For collection on all interfaces:

```
adb shell tcpdump -i any -s 0 -w /data/tcpdump.pcap
```

CNE related logging is enabled only when you push certain libraries. Follow the CNE logging procedure using the link below:

https://createpoint.qti.qualcomm.com/search/#search/KBA-151203111117\

Collect Routing rules and interface information using the link below:

https://createpoint.qti.qualcomm.com/search/#search/KBA-151207152805

7.6 Browsing issues

In addition to the logs mentioned for regular issues, the following info/logs are required:

- Run the following commands to collect the TCPDUMP:

```
PING to DNS server;
PING to Gateway;
Ip route get [destination ip-address]
```

TCPDUMP command

- For collection on specific interfaces. Example for “wlan0” is given below:

```
adb shell tcpdump -i wlan0 -w /data/wlan.pcap
```

- For collection on all interfaces:

```
adb shell tcpdump -i any -s 0 -w /data/tcpdump.pcap
```

Collect Routing rules and interface information using the link below:
https://createpoint.qti.qualcomm.com/search/#search/KBA-151207152805

7.7 For throughput issues

- Secondary boot image should be used; this is compulsory.
- Following steps are recommended for throughput testing:
  a. First try with MODEM-only call.
  b. If no issue is observed, try with embedded call.
  c. If no issue is observed, try with tethered call.
- Make a note of DUT and REF builds and mention the same.
- Collect the following output on DUT and REF for checking the TCP parameters.

```
- ls -l /proc/sys/net/ipv4/tcp*
```

To get a list of TCP-related parameters present in the folder on DUT and REF in order:

```
a) adb shell cat /proc/sys/net/ipv4/tcp*
b) adb shell cat /proc/sys/net/core/rmem*
c) adb shell cat /proc/sys/net/core/wmem*
d) adb shell getprop net.tcp.buffersize.*
(net.tcp.buffersize.default/wifi/lte/umts/hspa/hsupa/hsdpa/edge/
grps/evdo_b)
```

- Following logs need to be collected depending on the kind of throughput testing being performed:
  - QXDM log with DPL enabled.
  - Wireshark log captured on APPS side
  - TCPDUMP command to collect on all interfaces

    ```
    adb shell tcpdump -i any -s 0 -w /data/tcpdump.pcap
    ```

  - Wireshark log collected on PC

7.8 Scripts for collecting logs

1. **Create adb_commands_START.BAT file with below content:**

```
:: Script to collect all necessary logging

adb wait-for-device
adb root
adb remount
adb disable-verity

::Push DPM & CNE logging libraries
adb push dpmlog_32\libdpmlog.so /system/vendor/lib/.
adb push dpmlog_64\libdpmlog.so /system/vendor/lib64/.
adb push dpmlog_32\libdpmlog.so /data/dpm/.
adb push dpmlog_64\libdpmlog.so /data/dpm/.
adb shell setprop persist.dpm.loglevel 7825
adb shell setprop persist.cne.logging.qxdm 3974
adb push cnelog_32\libcnelog.so /system/vendor/lib32/
adb push cnelog_64\libcnelog.so /system/vendor/lib64/

:: Enable Verbose logging for IMS issues
adb shell setprop persist.ims.disableADBLogs    0
adb shell setprop persist.ims.disableDebugLogs 0
adb shell setprop persist.ims.disableQXDMLogs 0
adb shell setprop persist.ims.disableIMSLogs     0
adb shell setprop persist.camera.cpp.debug.mask    0
adb shell setprop persist.camera.hal.debug.mask     0
adb shell setprop persist.camera.hal.debug              0
adb shell setprop persist.camera.ISP.debug.mask    0
adb shell setprop persist.camera.pproc.debug.mask  0
adb shell setprop persist.camera.stats.debug            0
adb shell setprop persit.camera.imglib.logs               1
adb shell setprop persist.camera.mct.debug.mask    1
adb shell setprop persist.camera.sensor.debug         0
adb shell setprop persist.camera.global.debug   0
adb shell setprop vidc.debug.level 1

:: Add these lines in /data/local.prop adn keep it in patch where you are
running this script
:: log.tag.TelecomFramework=VERBOSE
:: log.tag.InCall=VERBOSE
:: log.tag.Telecom=VERBOSE
:: log.tag.Telephony=VERBOSE
:: log.tag.NetworkController=DEBUG
:: log.tag.NetworkControllerChat=DEBUG
adb push local.prop /data/local.prop
adb shell chmod 644 /data/local.prop
adb shell chown root /data/local.prop
:: qmi_fw Conf file
adb push qmi_fw.conf /system/etc/qmi_fw.conf
adb push qmi_fw.conf /etc/qmi_fw.conf
adb shell sync
adb reboot

timeout 1

adb wait-for-device & adb root
adb wait-for-device & adb remount
:: Screen Toggle
adb wait-for-device & adb shell input keyevent 26

adb wait-for-device & adb shell date -s `date +%G%m%d.%H%M%S` >
radio_file.txt
echo "Windows System Time is %time%" >> radio_file.txt
START "Android-Radio" cmd /c  "adb wait-for-device & adb logcat -v
threadtime -b radio >> radio_file.txt"
START "Main" cmd /c  "adb wait-for-device & adb logcat -v threadtime >
main_file.txt"
START "Kernel" cmd /c  "adb wait-for-device & adb shell cat /proc/kmsg >
klog_file.txt"
START "Android-ScreenRecorder" cmd /c "adb wait-for-device & adb shell
screenrecord /sdcard/ScreenRecord.mp4"
START "TCPDUMP" cmd /c "adb shell tcpdump -i any -s 0 -w
/data/tcpdump.pcap"
START "TCPDUMP_DUMMY" cmd /c "adb shell tcpdump -i dummy0 -s 0 -w
/data/tcpdump_dummy0.pcap"
START "IPTABLES" cmd /c "iptables_rules_routes_NwParam_v3.bat"

adb shell getprop > prop.txt
adb pull "/data/data/com.android.providers.telephony/databases/mmssms.db"
mmssms_begin.db
adb pull "/data/data/com.android.providers.settings/databases/settings.db"
settings_begin.db
adb pull
"/data/data/com.android.providers.telephony/databases/telephony.db"
telephony_begin.db
adb pull /data/anr/traces.txt   anr_begin.txt
adb pull /data/tombstones       tombstone_begin

START "Dumpstate" cmd /c  "adb shell dumpstate > dumpstate_begin.txt"
START "BugReport" cmd /c  "adb shell bugreport > bugr_begin.txt"
START "Dumpsys" cmd /c  "adb shell dumpsys > dumpsys_begin.txt"
START "Procrank" cmd /c  "adb shell procrank > procrank_begin.txt"

::Secondary boot image (Throughput)
```

```
adb pull /proc/config.gz
adb shell "cat /d/clk/enabled_clocks | grep qdss" > secboot_qdss.txt
adb shell cat /proc/version > secboot_version.txt

::RPS setting (Throughput)
adb shell cat /sys/class/net/rmnet0/queues/rx-0/rps_cpus > TPUT_RPS.txt

::GRO Status (Throughput)
adb pull /system/etc/data/netmgr_config.xml > TPUT_GRO_netmgr_config.xml

::APN-CONF XML
adb shell system/etc/apns-conf.xml > apns-conf.xml

::CNE, DPM config
adb pull /system/etc/cne/wqeclient/ cne_system_wqeclient
adb pull /system/etc/cne/  cne_system_etc
adb pull /system/etc/dpm/  dpm_system_etc
adb pull /data/dpm/        dpm_data

:: Screen Toggle
adb shell input keyevent 26
adb shell input keyevent 26
```

2. **Create Android_Logs_PULL.BAT file with below content:**

```
:: Run this script @ end of test case or when issue occurs
echo "PULLING logs";

adb wait-for-device
adb root

taskkill /F /FI "WindowTitle eq Android-ScreenRecorder" /T
taskkill /F /FI "WindowTitle eq Android-Radio" /T
taskkill /F /FI "WindowTitle eq Main" /T
taskkill /F /FI "WindowTitle eq Kernel" /T
taskkill /F /FI "WindowTitle eq Dumpstate" /T
taskkill /F /FI "WindowTitle eq BugReport" /T
taskkill /F /FI "WindowTitle eq Procrank" /T
taskkill /F /FI "WindowTitle eq TCPDUMP" /T
taskkill /F /FI "WindowTitle eq TCPDUMP_DUMMY" /T
taskkill /F /FI "WindowTitle eq IPTABLES" /T

timeout 2
adb pull /sdcard/ScreenRecord.mp4
adb pull /data/tcpdump.pcap
adb pull "/data/data/com.android.providers.telephony/databases/mmssms.db"
mmssms_end.db
```

```
adb pull "/data/data/com.android.providers.settings/databases/settings.db"
settings_end.db
adb pull
"/data/data/com.android.providers.telephony/databases/telephony.db"
telephony_end.db
adb pull /data/radio/rdsh_proxy_log.dat
adb pull /data/radio/rdsh_proxy_raw.dat
adb pull /data/anr/traces.txt   anr_end.txt
adb pull /data/tombstones       tombstone_end
echo "PULLING logs" > net_param/time.txt
adb shell dumpstate > dumpstate_end.txt
adb shell bugreport > bugr_end.txt
adb shell dumpsys > dumpsys_end.txt
adb shell procrank > procrank_end.txt
echo "PULLING logs COMPLETED";
```

3. **Create iptables_rules_routes_NwParam_v3.BAT file with below content:**

```
:: Must to collect for all EPDG, Data & TPUT issues
:: it is executed as a part of adb_commands_START.
adb root
adb wait-for-devices
adb root

:: ---------------------------------------------------------------------
-------------------------
:: Net Parameters - Begin
mkdir net_param
cd net_param
adb shell date -s `date +%G%m%d.%H%M%S` >> time.txt
adb shell date -s `date +%G%m%d.%H%M%S` >> Network_Tuning_Paramters3.txt
adb shell ls /proc/sys/net/ipv4/ip* >> Network_Tuning_Paramters3.txt
adb shell ls /proc/sys/net/ipv4/tcp* >> Network_Tuning_Paramters3.txt
adb shell cat /proc/sys/net/ipv4/ip* >> Network_Tuning_Paramters3.txt
adb shell cat /proc/sys/net/ipv4/tcp* >> Network_Tuning_Paramters3.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> r_rmnet_data0.txt
adb shell ls /proc/sys/net/ipv4/conf/r_rmnet_data0/* >> r_rmnet_data0.txt
adb shell cat /proc/sys/net/ipv4/conf/r_rmnet_data0/* >> r_rmnet_data0.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> r_rmnet_data1.txt
adb shell ls /proc/sys/net/ipv4/conf/r_rmnet_data1/* >> r_rmnet_data1.txt
adb shell cat /proc/sys/net/ipv4/conf/r_rmnet_data1/* >> r_rmnet_data1.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> rmnet_data0.txt
adb shell ls /proc/sys/net/ipv4/conf/rmnet_data0/* >> rmnet_data0.txt
adb shell cat /proc/sys/net/ipv4/conf/rmnet_data0/* >> rmnet_data0.txt
adb shell date -s `date +%G%m%d.%H%M%S` >> rmnet_data1.txt
adb shell ls /proc/sys/net/ipv4/conf/rmnet_data1/* >> rmnet_data1.txt
adb shell cat /proc/sys/net/ipv4/conf/rmnet_data1/* >> rmnet_data1.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> rmnet_data7.txt
adb shell ls /proc/sys/net/ipv4/conf/rmnet_data7/* >> rmnet_data7.txt
adb shell cat /proc/sys/net/ipv4/conf/rmnet_data7/* >> rmnet_data7.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> wlan0.txt
adb shell ls /proc/sys/net/ipv4/conf/wlan0/* >> wlan0.txt
adb shell cat /proc/sys/net/ipv4/conf/wlan0/* >> wlan0.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> rndis0.txt
adb shell ls /proc/sys/net/ipv4/conf/rndis0/* >> rndis0.txt
adb shell cat /proc/sys/net/ipv4/conf/rndis0/* >> rndis0.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> r_rmnet_data0_v6.txt
adb shell ls /proc/sys/net/ipv6/conf/r_rmnet_data0/* >>
r_rmnet_data0_v6.txt
adb shell cat /proc/sys/net/ipv6/conf/r_rmnet_data0/* >>
r_rmnet_data0_v6.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> r_rmnet_data1_v6.txt
adb shell ls /proc/sys/net/ipv6/conf/r_rmnet_data1/* >>
r_rmnet_data1_v6.txt
adb shell cat /proc/sys/net/ipv6/conf/r_rmnet_data1/* >>
r_rmnet_data1_v6.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> rmnet_data0_v6.txt
adb shell ls /proc/sys/net/ipv6/conf/rmnet_data0/* >> rmnet_data0_v6.txt
adb shell cat /proc/sys/net/ipv6/conf/rmnet_data0/* >> rmnet_data0_v6.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> rmnet_data1_v6.txt
adb shell ls /proc/sys/net/ipv6/conf/rmnet_data1/* >> rmnet_data1_v6.txt
adb shell cat /proc/sys/net/ipv6/conf/rmnet_data1/* >> rmnet_data1_v6.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> rmnet_data7_v6.txt
adb shell ls /proc/sys/net/ipv6/conf/rmnet_data7/* >> rmnet_data7_v6.txt
adb shell cat /proc/sys/net/ipv6/conf/rmnet_data7/* >> rmnet_data7_v6.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> rndis0_v6.txt
adb shell ls /proc/sys/net/ipv4/conf/rndis0/* >> rndis0_v6.txt
adb shell cat /proc/sys/net/ipv4/conf/rndis0/* >> rndis0_v6.txt

:: --------------------------------------------------------------------------
------------------------
cd ..
```

```
mkdir logs_iwlan
cd logs_iwlan
FOR /L %%I IN (1,1,500) DO (
    echo ===================== Iteration %%I ========================= >>
time.log
   adb shell date -s `date +%G%m%d.%H%M%S` >> time.log

    echo ===================== Iteration %%I ========================= >>
xfrm_state.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> xfrm_state.txt
   adb shell ip xfrm state show >> xfrm_state.txt

   echo ===================== Iteration %%I ========================= >>
xfrm_policy.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> xfrm_policy.txt
    adb shell ip xfrm policy show >> xfrm_policy.txt

   echo ===================== Iteration %%I ========================= >>
v46_ip_addr.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> v46_ip_addr.txt
   adb shell ip addr >> v46_ip_addr.txt

   echo ===================== Iteration %%I ========================= >>
v4_iptables_raw.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> v4_iptables_raw.txt
   adb shell iptables -t raw -L -n -v >> v4_iptables_raw.txt

   echo ===================== Iteration %%I ========================= >>
v4_iptables_mangle.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> v4_iptables_mangle.txt
   adb shell iptables -t mangle -L -n -v >> v4_iptables_mangle.txt

   echo ===================== Iteration %%I ========================= >>
v4_iptables_filter.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> v4_iptables_filter.txt
   adb shell iptables -L -n -v >> v4_iptables_filter.txt

   echo ===================== Iteration %%I ========================= >>
v4_iptables_nat.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> v4_iptables_nat.txt
   adb shell iptables -t nat -L -n -v >> v4_iptables_nat.txt

   echo ===================== Iteration %%I ========================= >>
v6_iptables_raw.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> v6_iptables_raw.txt
   adb shell ip6tables -t raw -L -n -v >> v6_iptables_raw.txt
   echo ===================== Iteration %%I ======================== >>
v6_iptables_mangle.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> v6_iptables_mangle.txt
   adb shell ip6tables -t mangle -L -n -v >> v6_iptables_mangle.txt

   echo ===================== Iteration %%I ======================== >>
v6_iptables_filter.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> v6_iptables_filter.txt
   adb shell ip6tables -L -n -v >> v6_iptables_filter.txt

   echo ===================== Iteration %%I ======================== >>
v6_iptables_nat.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> v6_iptables_nat.txt
   adb shell ip6tables -t nat -L -n -v >> v6_iptables_nat.txt

   echo ===================== Iteration %%I ======================== >>
v4_ip_rule_show.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> v4_ip_rule_show.txt
   adb shell ip rule show >> v4_ip_rule_show.txt

   echo ===================== Iteration %%I ======================== >>
v6_ip_rule_show.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> v6_ip_rule_show.txt
   adb shell ip -6 rule show >> v6_ip_rule_show.txt

   echo ===================== Iteration %%I ======================== >>
v4_ip_route_show_table_all.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v4_ip_route_show_table_all.txt
   adb shell ip route show table all >> v4_ip_route_show_table_all.txt

   echo ===================== Iteration %%I ======================== >>
v6_ip_route_show_table_all.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v6_ip_route_show_table_all.txt
   adb shell ip -6 route show table all >> v6_ip_route_show_table_all.txt

   echo ===================== Iteration %%I ======================== >>
v4_ip_route_show_table_rmnet_data0.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v4_ip_route_show_table_rmnet_data0.txt
   adb shell ip route show table rmnet_data0 >>
v4_ip_route_show_table_rmnet_data0.txt

   echo ===================== Iteration %%I ======================== >>
v4_ip_route_show_table_rmnet_data1.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v4_ip_route_show_table_rmnet_data1.txt
```

```
   adb shell ip route show table rmnet_data1 >>
v4_ip_route_show_table_rmnet_data1.txt

   echo ===================== Iteration %%I ========================= >>
v4_ip_route_show_table_rmnet_data2.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v4_ip_route_show_table_rmnet_data2.txt
   adb shell ip route show table rmnet_data2 >>
v4_ip_route_show_table_rmnet_data2.txt

   echo ===================== Iteration %%I ========================= >>
v4_ip_route_show_table_rmnet_data6.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v4_ip_route_show_table_rmnet_data6.txt
   adb shell ip route show table rmnet_data6 >>
v4_ip_route_show_table_rmnet_data6.txt

   echo ===================== Iteration %%I ========================= >>
v4_ip_route_show_table_rmnet_data7.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v4_ip_route_show_table_rmnet_data7.txt
   adb shell ip route show table rmnet_data7 >>
v4_ip_route_show_table_rmnet_data7.txt

   echo ===================== Iteration %%I ========================= >>
v4_ip_route_show_table_r_rmnet_data0.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v4_ip_route_show_table_r_rmnet_data0.txt
   adb shell ip route show table r_rmnet_data0 >>
v4_ip_route_show_table_r_rmnet_data0.txt

   echo ===================== Iteration %%I ========================= >>
v4_ip_route_show_table_r_rmnet_data1.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v4_ip_route_show_table_r_rmnet_data1.txt
   adb shell ip route show table r_rmnet_data1 >>
v4_ip_route_show_table_r_rmnet_data1.txt

   echo ===================== Iteration %%I ========================= >>
v4_ip_route_show_table_r_rmnet_data2.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v4_ip_route_show_table_r_rmnet_data2.txt
   adb shell ip route show table r_rmnet_data2 >>
v4_ip_route_show_table_r_rmnet_data2.txt

   echo ===================== Iteration %%I ========================= >>
v4_ip_route_show_table_1.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> v4_ip_route_show_table_1.txt
   adb shell ip route show table 1 >> v4_ip_route_show_table_1.txt
   echo ===================== Iteration %%I ========================= >>
v4_ip_route_show_table_9.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> v4_ip_route_show_table_9.txt
   adb shell ip route show table 9 >> v4_ip_route_show_table_9.txt

   echo ===================== Iteration %%I ========================= >>
v4_ip_route_show_table_dummy0.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v4_ip_route_show_table_dummy0.txt
   adb shell ip route show table dummy0 >>
v4_ip_route_show_table_dummy0.txt

   echo ===================== Iteration %%I ========================= >>
v4_ip_route_show_table_wlan0.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v4_ip_route_show_table_wlan0.txt
   adb shell ip route show table wlan0 >> v4_ip_route_show_table_wlan0.txt

   echo ===================== Iteration %%I ========================= >>
v6_ip_route_show_table_rmnet_data0.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v6_ip_route_show_table_rmnet_data0.txt
   adb shell ip -6 route show table rmnet_data0 >>
v6_ip_route_show_table_rmnet_data0.txt

   echo ===================== Iteration %%I ========================= >>
v6_ip_route_show_table_rmnet_data1.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v6_ip_route_show_table_rmnet_data1.txt
   adb shell ip -6 route show table rmnet_data1 >>
v6_ip_route_show_table_rmnet_data1.txt

   echo ===================== Iteration %%I ========================= >>
v6_ip_route_show_table_rmnet_data2.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v6_ip_route_show_table_rmnet_data2.txt
   adb shell ip -6 route show table rmnet_data2 >>
v6_ip_route_show_table_rmnet_data2.txt

   echo ===================== Iteration %%I ========================= >>
v6_ip_route_show_table_rmnet_data6.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v6_ip_route_show_table_rmnet_data6.txt
   adb shell ip -6 route show table rmnet_data6 >>
v6_ip_route_show_table_rmnet_data6.txt

   echo ===================== Iteration %%I ========================= >>
v6_ip_route_show_table_rmnet_data7.txt
```

```
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v6_ip_route_show_table_rmnet_data7.txt
   adb shell ip -6 route show table rmnet_data7 >>
v6_ip_route_show_table_rmnet_data7.txt

   echo ===================== Iteration %%I ========================= >>
v6_ip_route_show_table_r_rmnet_data0.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v6_ip_route_show_table_r_rmnet_data0.txt
   adb shell ip -6 route show table r_rmnet_data0 >>
v6_ip_route_show_table_r_rmnet_data0.txt

   echo ===================== Iteration %%I ========================= >>
v6_ip_route_show_table_r_rmnet_data1.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v6_ip_route_show_table_r_rmnet_data1.txt
   adb shell ip -6 route show table r_rmnet_data1 >>
v6_ip_route_show_table_r_rmnet_data1.txt

   echo ===================== Iteration %%I ========================= >>
v6_ip_route_show_table_r_rmnet_data2.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v6_ip_route_show_table_r_rmnet_data2.txt
   adb shell ip -6 route show table r_rmnet_data2 >>
v6_ip_route_show_table_r_rmnet_data2.txt

   echo ===================== Iteration %%I ========================= >>
v6_ip_route_show_table_wlan0.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v6_ip_route_show_table_wlan0.txt
   adb shell ip -6 route show table wlan0 >>
v6_ip_route_show_table_wlan0.txt

   echo ===================== Iteration %%I ========================= >>
v6_ip_route_show_table_1.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> v6_ip_route_show_table_1.txt
   adb shell ip -6 route show table 1 >> v6_ip_route_show_table_1.txt

   echo ===================== Iteration %%I ========================= >>
v6_ip_route_show_table_9.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >> v6_ip_route_show_table_9.txt
   adb shell ip -6 route show table 9 >> v6_ip_route_show_table_9.txt

   echo ===================== Iteration %%I ========================= >>
v6_ip_route_show_table_dummy0.txt
   adb shell date -s `date +%G%m%d.%H%M%S` >>
v6_ip_route_show_table_dummy0.txt
   adb shell ip -6 route show table dummy0 >>
v6_ip_route_show_table_dummy.txt
```

```
    echo ===================== Iteration %%I ========================= >>
xfrm_stat.txt
    adb shell date -s `date +%G%m%d.%H%M%S` >> xfrm_stat.txt
    adb shell cat /proc/net/xfrm_stat >> xfrm_stat.txt

    timeout 5
)
echo %DATE%-%TIME% >> time.log
cd ..

:: ---------------------------------------------------------------------------
-------------------------
:: Net Parameters - Begin
cd net_param
adb shell date -s `date +%G%m%d.%H%M%S` >> time.txt
adb shell date -s `date +%G%m%d.%H%M%S` >> Network_Tuning_Paramters3.txt
adb shell ls /proc/sys/net/ipv4/ip* >> Network_Tuning_Paramters3.txt
adb shell ls /proc/sys/net/ipv4/tcp* >> Network_Tuning_Paramters3.txt
adb shell cat /proc/sys/net/ipv4/ip* >> Network_Tuning_Paramters3.txt
adb shell cat /proc/sys/net/ipv4/tcp* >> Network_Tuning_Paramters3.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> r_rmnet_data0.txt
adb shell ls /proc/sys/net/ipv4/conf/r_rmnet_data0/* >> r_rmnet_data0.txt
adb shell cat /proc/sys/net/ipv4/conf/r_rmnet_data0/* >> r_rmnet_data0.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> r_rmnet_data1.txt
adb shell ls /proc/sys/net/ipv4/conf/r_rmnet_data1/* >> r_rmnet_data1.txt
adb shell cat /proc/sys/net/ipv4/conf/r_rmnet_data1/* >> r_rmnet_data1.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> rmnet_data0.txt
adb shell ls /proc/sys/net/ipv4/conf/rmnet_data0/* >> rmnet_data0.txt
adb shell cat /proc/sys/net/ipv4/conf/rmnet_data0/* >> rmnet_data0.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> rmnet_data1.txt
adb shell ls /proc/sys/net/ipv4/conf/rmnet_data1/* >> rmnet_data1.txt
adb shell cat /proc/sys/net/ipv4/conf/rmnet_data1/* >> rmnet_data1.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> rmnet_data7.txt
adb shell ls /proc/sys/net/ipv4/conf/rmnet_data7/* >> rmnet_data7.txt
adb shell cat /proc/sys/net/ipv4/conf/rmnet_data7/* >> rmnet_data7.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> wlan0.txt
adb shell ls /proc/sys/net/ipv4/conf/wlan0/* >> wlan0.txt
adb shell cat /proc/sys/net/ipv4/conf/wlan0/* >> wlan0.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> rndis0.txt
adb shell ls /proc/sys/net/ipv4/conf/rndis0/* >> rndis0.txt
adb shell cat /proc/sys/net/ipv4/conf/rndis0/* >> rndis0.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> r_rmnet_data0_v6.txt
adb shell ls /proc/sys/net/ipv6/conf/r_rmnet_data0/* >>
r_rmnet_data0_v6.txt
adb shell cat /proc/sys/net/ipv6/conf/r_rmnet_data0/* >>
r_rmnet_data0_v6.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> r_rmnet_data1_v6.txt
adb shell ls /proc/sys/net/ipv6/conf/r_rmnet_data1/* >>
r_rmnet_data1_v6.txt
adb shell cat /proc/sys/net/ipv6/conf/r_rmnet_data1/* >>
r_rmnet_data1_v6.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> rmnet_data0_v6.txt
adb shell ls /proc/sys/net/ipv6/conf/rmnet_data0/* >> rmnet_data0_v6.txt
adb shell cat /proc/sys/net/ipv6/conf/rmnet_data0/* >> rmnet_data0_v6.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> rmnet_data1_v6.txt
adb shell ls /proc/sys/net/ipv6/conf/rmnet_data1/* >> rmnet_data1_v6.txt
adb shell cat /proc/sys/net/ipv6/conf/rmnet_data1/* >> rmnet_data1_v6.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> rmnet_data7_v6.txt
adb shell ls /proc/sys/net/ipv6/conf/rmnet_data7/* >> rmnet_data7_v6.txt
adb shell cat /proc/sys/net/ipv6/conf/rmnet_data7/* >> rmnet_data7_v6.txt

adb shell date -s `date +%G%m%d.%H%M%S` >> rndis0_v6.txt
adb shell ls /proc/sys/net/ipv4/conf/rndis0/* >> rndis0_v6.txt
adb shell cat /proc/sys/net/ipv4/conf/rndis0/* >> rndis0_v6.txt

cd ..
:: ---------------------------------------------------------------------
------------------------
Pause
```

4. **Create local.prop file with below content:**

```
log.tag.TelecomFramework=VERBOSE
log.tag.InCall=VERBOSE
log.tag.Telecom=VERBOSE
log.tag.Telephony=VERBOSE
log.tag.NetworkController=DEBUG
log.tag.NetworkControllerChat=DEBUG
```

5. **Create qmi_fw.conf file with below content:**

```
QMI_CCI_DEBUG_LEVEL=3
QMI_CSI_DEBUG_LEVEL=3
```

6. **Copy all the below scripts to a folder on your PC/laptop:**
   - adb_commands_START.BAT
   - Android_Logs_PULL.BAT
   - iptables_rules_routes_NwParam_v3.BAT
   - local.prop
   - qmi_fw.conf

7. **Make sure you download libdpmlog.so and libcnelog.so from KBA**

```
LIBDPMLOG.SO   ==> https://createpoint.qti.qualcomm.com/search/#search/KBA-
151207131520
LIBCNELOG.SO   ==>
https://createpoint.qti.qualcomm.com/search/#search/KBA-151203111117

You could select one of products from "View in context:", then please find
the attached files under "Related Docs".
```

8. **Start collecting logs by clicking "adb_commands_START.BAT"**

9. **At end of test case (or) when issue occurs, Run "Android_Logs_PULL.BAT"**

A References

A.1 Related documents

Title	Number
Qualcomm Technologies, Inc.	
RJIL Smartphone Specifications Compliance of QTI Devices (MSM8916 and MSM8909)	MH80-P1185-1
India/RJIL Pre-Cert: OEM Checklist	80-P1034-2
India/RJIL Pre-Cert: MPSS Test Plan	80-P1034-3
RJIL Pre-Cert: APSS Test Plan	80-P6073-1
RJIL Customization Requirements	80-P3336-1
Updating Modem Configurations in Factory and OTA	80-NV514-1
Creating MBN and Golden EFS Builds	80-NU184-1
Data Throughput Analysis for Modem and HLOS Data	80-NJ361-1
Data Throughput Issue Checkpoints	80-NH569-1
WFC Voice Quality Indicator	80-P4651-1

A.2 Acronyms and terms

Acronym or term	Definition
AOST	Advanced Optional Software Technologies
eMBMS	Evolved Multimedia Broadcast/Multicast Service
IMS	IP Multimedia Subsystem
LTE	Long Term Evolution
MBN	Modem Binary
ODM	Original Device Manufacturer
OEM	Original Equipment Manufacturer
RCS	Rich Communication Suite
RJIL	Reliance JIO Infocomm Limited
VoLTE	Voice over LTE
ViLTE	Video over LTE (also referred to as VT)
VoWiFi	Voice over Wi-Fi