



Prolin API Programming Guide

V 2.4.0



PAX Computer Technology(Shenzhen)Co.,Ltd.

Copyright Statement

Copyright © 2000-2020 PAX Computer Technology (Shenzhen) Co., Ltd. All rights reserved. Any part of this documentation is strictly prohibited to be reproduced or distributed in any form or for any purpose without the prior written permission of PAX Computer Technology (Shenzhen) Co., Ltd. Despite the effort to enhance the accuracy of this documentation, there are possible errors or omissions left herein. The content of this documentation is subject to change without further notice. The examples and sample programs demonstrated in this manual are for illustrations only and may fail to meet practical needs. Please test and verify their applicability before putting into commercial use.

History of Revisions

Date	Version	Note	Author
2012-08-15	V1.0.0	1. Exported the original version of Prolin from WIKI.	Prolin Team
2012-11-06	V1.0.1	1. Internal review and modification.	Prolin Team
2012-12-26	V1.0.2	1. Added Return Code List in system function; 2. Added a new interface: <i>OsVerifySignExternal()</i> ; 3. Added WiFi module; 4. Added Appendix registry table.	Prolin Team
2013-02-20	V1.0.3	Added instruction of <i>OsOpenFont()</i> .	Prolin Team
2013-03-06	V1.0.4	1. Modified the instruction of <i>OsModemOpen()</i> ; 2. Added two return values in Modem: -3219 and -3220; 3. Added notes of <i>DetectDialTone</i> ; Added a new interface: <i>OsModemSwitchPower()</i> ; 4. Modified parameter description of <i>OsPrnOpen()</i> to “support bmp format only”; 5. Modified <i>sci_get_fd()</i> .	Prolin Team
2013-04-17	V1.0.5	1. Modified the description of Chapter 4.10 “S series do not support this function as S model has no landline”; 2. Modified <i>open()</i> node in Chapter 14.4.1; 3. Updated time setting code “ <i>pow-hwclock -w</i> ” to “ <i>pow-hwclock -w -u</i> ”; 4. Updated instruction of <i>OsWLInit()</i> : When return value is <i>ERR_WL_NOSIM</i> , some functions can be used without an SIM card; 5. Updated the instruction of <i>OsCheckBattery()</i> .	Prolin Team
2013-05-17	V1.0.6	1. Modified the description of <i>Timer</i> ;	Prolin Team

		<ul style="list-style-type: none"> 2. Supplemented the description of <i>Time Delay</i>; 3. Added description of Registry Table; 4. Modified the parameter description of <i>ValueinOsRegGetValue()</i>; 5. Modified the parameter description of <i>ShaOut</i> in <i>OsSHA()</i>; 6. Updated description of GUI; 7. Added “1200, V22” and “2400, FC” to the parts of synchronous variable of <i>ConnectRate[20]</i> in <i>MODEM</i>. 	
2013-08-09	V1.0.7	<ul style="list-style-type: none"> 1. Modified the Registry; 2. Modified the data structure in Font Library; 3. Updated interface <i>OsCheckBattery()</i>; 4. Updated chapter 15 IC Card Reader and chapter 16 RF Reader. 	Prolin Team
2013-10-22	V1.0.8	<ul style="list-style-type: none"> 1. Added chapter 4.11 Save the Crash Report for System Function; 2. Modified the brightness level to [0~10] in <i>OsScrBrightness()</i> in the chapter of LCD and 0 means closing the backlight; 3. Updated key value definition list in the chapter of Keyboard. 	Prolin Team
2013-10-31	V1.0.9	<ul style="list-style-type: none"> 1. Updated the chapter of SystemFunction; 2. Modified character conversion interface; 3. Added new interfaces <i>OsPortReset()</i> and <i>OsPortCheckTx()</i> in the chapter of Communication; 4. Updated chapter Audio and added a new interface: <i>OsPlayWave()</i>. 	Prolin Team
2013-11-20	V2.0.0	<ul style="list-style-type: none"> 1. Added parameter description of <i>KeyVarType</i> about 0x02 in 	Prolin Team

		<p>OsPedDesDukpt();</p> <ol style="list-style-type: none"> 2. Modified the value ranges of several parameters in chapter PED; 3. Modified the parameter description of ucATQ0 in <i>OsPiccAntiSel()</i>; 4. Deleted <i>OsPiccGetParam()</i> and <i>OsPiccSetParam()</i>; 5. Updated the instruction of <i>OsRegGetValue()</i>; 6. Added description of <i>OsPortOpen()</i>; 7. Added description of <i>OsSysSleep()</i>; 8. Updated Return Code List in Network Configuration chapter; 9. Added instruction of <i>OsNetDns()</i>. 	
2014-02-25	V2.0.1	<ol style="list-style-type: none"> 1. Modified Appendix 4; 2. Modified the instruction of <i>OsWISel/Sim()</i>; 3. Added a new return value for <i>OsMsrOpen()</i>; 4. Added new interfaces <i>OsWILoginEx()</i>, <i>OsMount()</i> and <i>OsUmount()</i>; 5. Added chapter 26 Barcode; 6. Added description of AES functional interface; 7. Modified the parameter description of <i>Name</i> in <i>OsInstallFile()</i>; 8. Added an error code “<i>ERR_APP_MODE</i>” for <i>OsInstallFile()</i>; 9. Added a new interface <i>OsCheckPowerSupply()</i>; 10. Deleted invalid codes in chapter 7 LCD for upgraded Prolin-2.4. 	Prolin Team
2014-03-04	V2.0.2	<ol style="list-style-type: none"> 1. Updated <i>OsSysSleep()</i>; 2. Updated the WiFi module. 	Prolin Team
2014-03-24	V2.0.3	Modified the description of WiFi module.	Prolin Team
2014-04-16	V2.0.4	<ol style="list-style-type: none"> 1. Updated WiFi module; 3. Added a new interface <i>OsSysSleepEx()</i>; 4. Removed MountFlag’s support to MS-MOVE in <i>OsMount()</i>; 	Prolin Team

		5. Added new interfaces <code>OsReboot()</code> and <code>OsPowerOff()</code> .	
2014-06-03	V2.0.5	<ul style="list-style-type: none"> 1. Modified the instruction of <code>OsPrnSetIndent()</code>; 2. Modified functionality description of <code>OsModemCheck()</code>; 3. Modified parameter description of <code>TimeoutMS</code> in <code>OsPortRecv()</code>; 4. Modified value range of <code>Volume</code> in <code>OsPlayWave()</code>; 5. Modified parameter description of <code>OsMifareOperate()</code>; 6. Added the corresponding relationship between device node and serial port. 7. Updated the instruction of <code>OsWiSwitchSleep()</code> and <code>OsWifiSwitchPower()</code>. 	Prolin Team
2014-07-10	V2.0.6	<ul style="list-style-type: none"> 1. Updated the instruction of <code>OsLog()</code>; 2. Updated the return value of <code>OsPedSetInterval()</code>; 3. Added a new interface <code>OsPedCancelPinEntry()</code>; 4. Added new return values of <code>OsVerifySign()</code> and <code>OsVerifySignExternal()</code>; 5. Updated the instruction of <code>OsCheckBattery()</code>; 6. Updated Appendix 3 and Appendix 4. 	Prolin Team
2014-10-09	V2.0.7	<ul style="list-style-type: none"> 1. Updated the return value <code>PPP_LOGOUTING</code> in <code>OsWiLogin()</code> and add corresponding instruction; 2. Modified <code>int OsWifiClose(void)</code> to <code>void OsWifiClose(void)</code>; 3. Modified device nodes of LCD, keypad and touch screen. 	Prolin Team
2014-11-26	V2.0.8	<ul style="list-style-type: none"> 1. Modified the instruction of <code>OsRunApp()</code>; 2. Added U disk file format limitation in <code>OsMount()</code>; 3. Added the instruction of parameter <code>Channel</code> about S920 model in <code>OsPortOpen()</code>; 4. Modified the instruction of 	Prolin Team

		<p><i>OsNetPing();</i></p> <ol style="list-style-type: none"> 5. Modified the instruction of <i>OsNetDns();</i> 6. Added the instruction of <i>OsNetSetConfig();</i> 7. Added the instruction of <i>OsNetStartDhcp();</i> 8. Modified the functionality of <i>OsNetSetRoute()</i> and added a new return value; 9. Modified functionality, return value and instruction of <i>OsNetGetRoute();</i> 10. Modified the instruction of <i>OsPppoeLogin();</i> 11. Modified the instruction of <i>OsWILogin();</i> 12. Modified the instruction of <i>OsWILoginEx();</i> 13. Added sampling frequency limitation of WAVE file in <i>OsPlayWave();</i> 14. Added new instruction for S920 models in Appendix 4. 	
2014-12-18	V2.0.9	<ol style="list-style-type: none"> 1. Added a new interface <i>OsNetPingEx()</i> to check the status of network connection; 2. Added some return codes in chapter 2.2 General Return Code; 3. Added a return value and instruction in <i>OsPrnOpen();</i> 4. Added some return values and instruction in <i>OsWILock();</i> 5. Added a new return value and instruction in <i>OsWILogout();</i> 6. Modified the description of return value in <i>OsCheckBattery();</i> 7. Added new instruction in <i>OsNetSetRoute();</i> 8. Added two functions <i>OsFirmwareGetVersion()</i> and <i>OsFirmwareUpgrade()</i> in the chapter of System Function. 	Prolin Team
2015-02-06	V2.1.0	<ol style="list-style-type: none"> 1. Added a new return value <i>ERR_APP_NOT_EXIST</i> in <i>OsRunApp();</i> 	Prolin Team

		<p>2. Modified the instruction of <i>OsPedSetInterval()</i>;</p> <p>3. Updated the parameter description of <i>DataIn</i> in <i>OsPedUpdatePinBlock()</i>;</p> <p>4. Modified the instruction of <i>OsPedDesDukpt()</i>;</p> <p>5. Modified parameter description of <i>DataInLen</i> in <i>OsPedRsaRecover()</i>;</p> <p>6. Modified Appendix 1;</p> <p>7. Added a new return value <i>ERR_WL_NOREG</i> in <i>OsWlInit()</i> denoting “fail to register GPRS network ”;</p> <p>8. Added a new restriction in chapter Wifi: Prolin WiFi only supports modules whose keyword <i>ro.fac.wifi</i> is “RS9110-N-11-02”, “01”, “03” or “04”;</p> <p>9. Added a new use instruction in <i>OsPortOpen()</i>: Software flow control and hardware flow control will be closed after calling this function;</p> <p>10. Added two lines of code in “Set Configuration Parameters of Communication Port”.</p>	
2015-04-13	V2.1.1	<p>1. Added specifications of LCD size and rotation method of different POS models in Appendix 4;</p> <p>2. Added a new macro <i>ERR_BATTERY_ABSENT</i> in return value of <i>OsWifiOpen()</i> and added its instruction;</p> <p>3. Added two new battery management functions: <i>OsPmGetEvent()</i> and <i>OsPmRequest()</i>;</p> <p>4. Added key words <i>persist.sys.sleeptime</i> and <i>persist.sys.sleepwaittime</i>.</p>	Prolin Team
2015-07-09	V2.1.2	<p>1. Revised the V2.1.1 version of this documentation;</p> <p>2. Added a return value and function instruction in condition</p>	Prolin Team

		<p>that the battery power of D200 or S920 gets too low (<3.65V);</p> <ol style="list-style-type: none"> 3. Added the instruction of <code>OsPedWriteRsaKey()</code>; 4. Supplemented the instruction of <code>OsSysSleep()</code>; 5. Added and modified some codes of user configuration structure: <code>PCD_USER_ST</code>; 6. Modified parameter description of <code>PUKType</code> in <code>OsVerifySign()</code>; 7. Added some codes in TCP programming; 8. Modified the instruction of <code>OsInstallFile()</code>; 9. Added the instruction of <code>P^X7</code> and <code>P^X5</code> in <code>OsPortOpen()</code>; 10. Supplemented the instruction of Prolin-2.4 and Prolin-phoenix-2.5 in <code>OsPedWriteKey()</code>; 11. Added some descriptions of <code>P^X5</code> and <code>P^X7</code> in chapter POSIX Interface; 12. Added <code>P^XX</code> model in Appendix 4. 	
2015-11-24	V2.1.3	<ol style="list-style-type: none"> 1. Added a return value <code>ERR_PUK_NOT_EXIST</code> in the chapter of System Function; 2. Added the instruction of <code>OsMount()</code>; 3. Added the instruction of <code>OsUmount()</code>; 4. Added the macro of SM (Secure Module) key in the chapter of PED; 5. Added the SM part in <code>OsPedWriteKey()</code> function; 6. Added nine functions in the chapter of PED, including <code>OsPedGenSM2Pair()</code>, <code>OsPedWriteSM2Key()</code>, <code>OsPedSM2Sign()</code>, <code>OsPedSM2Verify()</code>, <code>OsPedSM2Recover()</code>, <code>OsPedSM()</code>, <code>OsPedSM()</code>, <code>OsPedGetMacSM()</code>, and <code>OsPedGetPinBlockSM4()</code>; 7. Added two new return values <code>ERR_MSR_END_ZEROERR</code> and 	Prolin Team

		<p>ERR_MSR_PED_DECRYPTER R in the chapter of MSR;</p> <p>8. Added new return values in Return Code List and added PPP_DIRECT link in Physical Channel List in the chapter of Network Configuration;</p> <p>9. Added a return value ERR_MODE_NOT_SUPPORT in the chapter of WiFi;</p> <p>10. Added return value and instruction in OsSysSleep();</p> <p>11. Added return value and instruction in OsSysSleepEx().</p>	
2016-06-13	V2.1.4	<p>1. Updated the instruction of OsMount();</p> <p>2. Updated the instruction of OsUmount();</p> <p>3. Added a return value and instruction in OsFirmwareUpgrade();</p> <p>4. Added introduction of three-level key system and PED key mechanism in PED chapter;</p> <p>5. Added OsGetOptInfo() and its corresponding data structure in chapter System Function;</p> <p>6. Added instruction of low battery in RF, Wifi, Printer and GPRS chapters;</p> <p>7. Added a member <i>unsigned char anti_interference_flag</i> in user configuration structure <i>struct pcd_user_t</i> in RF chapter;</p> <p>8. Modified some errors of Ipv6 related interfaces;</p> <p>9. Added some related return values and data structure of Ipv6 in chapter Network Configuration;</p> <p>10. Added some system configuration parameters in Appendix 3 Registry.</p>	Prolin Team
2016-07-04	V2.1.5	<p>1. Updated the instruction of OsSysSleep();</p> <p>2. Modified the value range of ConnectTimeout from 0~300 to 0~60 in the table of Variable</p>	Prolin Team

		<p>definitions of ST_MODEM_SETUP in chapter MODEM;</p> <ul style="list-style-type: none"> 3. Updated the parameter KeepAlive in OsWILogin() and OsWILoginEx(); 4. Updated the instruction of OsModemConnect(); 5. Updated the instruction of OsSysSleepTime(); 6. Added a new system configuration parameter <i>persist.sys.lcdsaverbrightness</i> and deleted <i>persist.sys.eth0.enable</i> in the Appendix 3 Registry; 7. Added three encrypted file system interfaces OsCryptFormat(), OsCryptMount() and OsCryptUnmount() and related return code list in chapter System Function; 8. Added chapter 23 GPS; 9. Added a new return value POWER_WPC and an related note in OsCheckPowerSupply(); 10. Modified the second point in the note area under the OsPmGetEvent(). 	
2016-07-21	V2.1.6	<ul style="list-style-type: none"> 1. Added an interface OsPedSetPinIconLayout() in PED chapter; 2. Added an interface OsPrnClrBuf() in Printer chapter; 3. Added a note about Bssid member of ST_WifiApSet structure in the instruction of OsWifiConnect(); 4. Added two new system configuration parameters <i>persist.sys.wifi.roam.dhcp</i> and <i>persist.sys.tm.imei</i> in Appendix 3 Registry; 5. Updated the instruction of OsGpsClose(); 6. Updated the instruction of OsWISwitchPower(); 7. In 6.5.6 	Prolin Team

		<p>OsPedGetPinBlock(),6.6.1 OsPedGetPinDukpt(),6.4.4 OsPedVerifyPlainPin(),6.4.5 OsPedVerifyCipher() and 6.9.9 OsPedGetPinBlockSM4()</p> <p>interfaces, added a note about how calling they might cause an exception to the applications that are running XUI and how to solve this problem.</p>	
2016-10-17	V2.1.7	<ol style="list-style-type: none"> 1. In Appendix 4 Validity of models and contents, added Q80 and D220 models to the validity table; 2. In MODEM chapter, added two new return values, the error codes are -3180 and -3184; 3. Added new system configuration parameters <i>persist.sys.timezone.tz</i>, <i>persist.sys.delayshutdown</i> and <i>persist.sys.continue.print</i> in Appendix 3 Registry; 4. Added <i>DnsAddrInfo</i> structure and <i>OsNetDnsEx()</i> function in chapter 20 Network Configuration; 5. Added a new interface <i>OsWifiWpsConnect()</i> in WiFi chapter; 6. In Power Management chapter, added poweroff event, forbidding poweroff and allowing poweroff functions; 7. In Printer chapter, updated the instruction of interface <i>OsPrnStart()</i> and added a new interface <i>OsPrnSetParam()</i>. 	Prolin Team
2017-03-10	V2.1.8	<ol style="list-style-type: none"> 1. Added new system configuration parameters <i>persist.sys.autouload.enable</i>, <i>persist.sys.autostartup.enable</i>, <i>persist.sys.hostname</i>, <i>persist.sys.wnet.version</i> and <i>persist.sys.confirmshutdown</i> in Appendix 3 Registry; 2. In Network Configuration chapter, added two new return values ERR_ROUTE_EXIST 	Prolin Team

		<p>and ERR_ROUTE_NONEXIST, a structure Ipv4RouteTable and three Ipv4 interfaces OsNetSetRouteTable(), OsNetDelRouteTable() and OsNetGetRouteTable();</p> <ul style="list-style-type: none"> 3. In Keyboard chapter, added a new key value KEYCAMERA in the key definition table; 4. In Barcode chapter, added data definition and OsScanSetParam() interface of camera's scan barcode function; 5. In Power Management chapter, added return code list and POWER_TYPE structure; 6. Added description of ports used by USB scanner; 7. Added instructions for OsPortOpen() and OsPortSend(); 8. Added chapter 24 Base; 9. Added instruction and return value for OsScanSetParam(); 10. Modified instruction of OsBaseOpen() and OsBaseClose(); 11. Delete the repeated return value -3606 in barcode module; 12. Added <i>persist.sys.keypad.duty_cycle</i> and items related to barcode module in registry table; 13. Deleted <i>ro.fac.scanner</i> in registry table. 	
2017-07-13	V2.1.9	<ul style="list-style-type: none"> 1. Added a return value of OsGpsOpen(); 2. Added instruction of OsNetSetDns(); 3. Added note of OsSysSleep(); 4. Added <i>ro.fac.coulomb_counter</i> key in registry table; 5. Modified the function of OsGetSysVer(); 6. Added a note of OsSysSleep(); 7. Modified the instruction of OsPortRecv(); 8. Added <i>persist.sys.enable.debug</i> 	Prolin Team

		<p>and rt.sys.mainapp.restart in registry table, and updated the relevant function instructions;</p> <ul style="list-style-type: none"> 9. Modified the description of lights in OsScanSetParam(); 10. Updated the parameter description of KeyVarType in OsPedDesDukpt(); 11. Added note of OsGetAppInfo(); 12. Added the length limit of key in OsRegSetValue() function; 13. Added return code list, OsRecordOpen(), OsRecordStart(), OsRecordStop(), OsRecordCheck(), OsRecordClose() and OsStopPlayWave() functions in chapter 27 Audio. 14. Add new functions OsPiccApplePoll() and OsPiccOffCarrier(). 	
2017-09-07	V2.2.0	<ul style="list-style-type: none"> 1. Add a Enumeration structure WAKEUP_SOURCE; 2. Add a new function OsWakeupsSource(); 3. Add a new function OsPedSetPinBg(). 4. Add a new function OsPedCustomKeypad(). 	Prolin Team
2017-11-15	V2.2.1	<ul style="list-style-type: none"> 1. Modified the description of OsRecordStart(); 2. Updated the instruction of OsPedCustomKeypad(); 3. Added persist.sys.ped.keypad.mask in registry table; 4. Added persist.sys.sound.enable in registry table; 5. Modified the description of persist.sys.delayshutdown in registry table; 6. Modified the instructions of OsSysSleep() and OsSysSleepEx(); 7. Updated the instruction of OsPmGetEvent(); 8. Added ro.fac.security_level and ro.fac.security_mode in registry 	Prolin Team

		table; 9. Added camera photography function.	
2018-01-08	V2.2.2	1. Added persist.sys.mainapp.restart in registry table; 2. Added persist.sys.ped.keypad.type in registry table; 3. Added a new function OsGetAppExitCode(); 4. Added persist.sys.batterylow.offcnt, persist.sys.reboot.cnt and persist.sys.powerkey.offcnt in registry table; 5. Added a new function OsPedSetOfflinePin(); 6. Modified instruction and parameter Dns of OsNetSetDns(); 7. Modified the value of parameter InputLen in PED SM2 Algorithm.	Prolin Team
2018-04-02	V2.2.3	1. Added macro PORT_USBHOST1, and modified the instruction of OsPortOpen(); 2. Modified the instruction of OsNetSetDns() in section 20.4.1; 3. Added persist.sys.dns.static in registry table; 4. Added persist.sys.xcb.installapp.lock in registry table; 5. Modified the description of <i>DataIn</i> in section 6.5.6 OsPedGetPinBlock; 6. Updated section 24.3 API 7. Updated the instructions of OsInstallFile() and OsFirmwareUpgrade().	Prolin Team
2018-05-23	V2.2.4	1. Updated the instructions of OsPedSetPinIconLayout(), the description of parameters <i>DataIn</i> and <i>Mode</i> in OsPedUpdatePinBlock() and the description of parameters <i>InitVector</i> , <i>DataInLen</i> and <i>Mode</i> in OsPedAes();	Prolin Team

		<ul style="list-style-type: none"> 2. Modified the description of persist.sys.ped.keypad.type and added persist.sys.ped.pin.icon.alignin in registry table; 3. Added interface OsScanDecodeBuf(). 4. Added the permission instruction of OsInstallFile() and OsUninstallFile(); 5. Deleted section 14.4 POSIX; 6. Added return code -3606. 	
2018-08-16	V2.2.5	<ul style="list-style-type: none"> 1. Updated the content of chapter “IC Card Reader” and “RF Reader”; 2. Updated the description of “POSIX Interface”. 3. Updated the description of parameters <i>DataInLen</i> and <i>Mode</i> in OsPedDes(); 4. Added section “6.10 DES FIRE”; 5. Added interface OsPiclInitIso15693(); 6. Added a new return code -2959 to RF module; 7. Added interfaces OsPedEndPinEntry() and OsPedPinKeyNotify(), and updated section “6.5.10 OsPedGetKcv”. 	Prolin Team
2018-11-15	V2.2.6	<ul style="list-style-type: none"> 1. Added NET_LINK_BT for Bluetooth link; 2. Modified the instruction of the registry key “persist.sys.ped.keypad.type”; 3. Added two interfaces OsPlayAudio() and OsStopPlayAudio(); 4. Added macros <i>RESOLUTION_WIDTH_1024_H_EIGHT_480</i> and <i>CAMERA_PIXEL_FORMAT_SB_GGR8</i> in chapter “Barcode and Camera”; 5. Updated the instruction in section OsCameraCapture(); 6. Updated the parameter and instruction in OsPedGetPinBlock(); 	Prolin Team

		<ul style="list-style-type: none"> 7. Updated the parameter and instruction in OsPedWriteAesKey(); 8. Updated the function description of OsPedAes(); 9. Added a new return value ERR_MSR_NO_TRACK_ERR in “MSR” module; 10. Added a new interface OsMsrReadJIS(); 11. Added NET_LINK_USB link in “Network Configuration” module; 12. Added event type PM_MSG_POWER_ABNORMA L and interface OsCheckPowerStatus() in “Power Management” module. 	
2018-11-29	V2.2.7	<ul style="list-style-type: none"> 1. Updated the description of parameter Level in “OsSysSleepEx”; 2. Added key word ro.security_version in “Appendix 3 Registry”; 3. Modified the description of DUKPT mechanism; 4. Updated the structure WAKEUP_SOURCE; 5. Updated the function instruction and description of parameter level in OsSysSleepEx(); 6. Updated the applicable platform of OsWakeupSource(). 	Prolin Team
2019-02-01	V2.2.8	<ul style="list-style-type: none"> 1. Updated the limited models for OsSysSleep() and OsSysSleepEx(); 2. Added interface OsWIInitEx(); 3. Modified the description of return code -3510 in chapter “GPRS/CDMA”. 	Prolin Team
2019-03-13	V2.2.9	<ul style="list-style-type: none"> 1. Added keyword persist.sys.usbd.mode in the registry; 2. Added new interface OsLed(), and list the interface parameter settings related to the model in the instruction; 3. Updated parameter KeyFlag in OsPedSetFunctionKey(), and the instruction of 	Prolin Team

		OsPedGetPinBlock(), and the keyword persist.sys.ped.keypad.type in the registry.	
2019-04-22	V2.3.0	Added PM_MSG_BATTERY_DAMAGE and upgraded section “OsCheckPowerStatus()”.	Prolin Team
2019-05-15	V2.3.1	<ol style="list-style-type: none"> 1. Added a new interface OsTerminalConsumInfo(); 2. Added description of usbd.mode=4 in keyword persist.sys.usbd.mode 3. Added PCD_ERR_RXLEN_EXCEED_F LAG in the return code list of RF reader; 4. Updated section “User Configuration Structure”; 5. Added macro <i>ERR_EAP_ID</i> in the WiFi return code list. 6. Added a new interface OsPedRkilInjectKey(). 	Prolin Team
2019-06-20	V2.3.2	<ol style="list-style-type: none"> 1. Added a new interface OsSetKeyTone(); 2. Modified the instruction of OsRegSetValue(). 3. Modified the parameter description in OsPedGetKcv(); 4. Added a new interface OsNetNat(). 	Prolin Team
2019-07-23	V2.3.3	<ol style="list-style-type: none"> 1. Modified the instruction of OsCheckPowerStatus(); 2. Added <i>ERR_USB_MODE</i>; 3. Added instruction in OsOnBase(); 4. Updated the return code and instruction of OsBaseOpen(); 5. Added interface OsCheckPortStatus(). 	Zhang Mengying Zheng Zhihai
2019-09-12	V2.3.4	<ol style="list-style-type: none"> 1. Added instruction for OsPedCancelPinEntry() and OsPedEndPinEntry(); 2. Updated the parameter description of OsPedGetPinBlock(); 3. Added keywords “persist.sys.ped.reboot.cycle” and 	Chen Xiaoyong Zheng Zhihai

		<p>“persist.sys.ped.pinwait,retain” in the Registry table;</p> <p>4. Updated the instruction and parameter description of OsCheckPortStatus().</p>	
2019-11-20	V2.3.5	<p>1. Added a new interface OsGetBaseInfo();</p> <p>2. Updated the description of parameter <i>DutyCycle</i>;</p> <p>3. Updated interface OsCameraCapture();</p> <p>4. Added a new interface OsCameraDetectMotion();</p> <p>5. Modified the definition of wakeup source;</p> <p>6. Added description for interface OsCheckBMSMode(), keyword <i>rt.app.key.tone</i> and <i>rt.app.key.backlight</i>.</p>	Zheng Zhihai Huang Ruzhen Li Xin Zhang Mengying
2019-12-03	V2.3.6	<p>1. Added section “Digital IO”;</p> <p>2. Updated section “Macro Definition of Camera” and “OsScanSetParam”;</p> <p>3. Updated chapter “Power Management” and the description of “persist.sys.confirmshutdown” in registry;</p> <p>4. Added section “OsPortCheckRx”;</p> <p>5. Updated the return value in section “OsCheckPortStatus” and “OsGetBaseInfo”;</p> <p>6. Added section “OsBaseCmd”.</p>	Zheng Zhihai Fang Wei Zhang Mengying Zheng Zhihai
2019-12-31	V2.3.7	<p>1. Updated the instruction of section “OsPortRecv”;</p> <p>2. Added the instruction of “OsPortCheckRx”.</p>	Xu Bin Zhang Mengying
2020-04-01	V2.3.8	<p>1. Added the return code description for installing fwp in the OsInstallFile();</p> <p>2. Updated the description of the registry key “persist.sys.usbd.mode”.</p>	Zhang Mengying
2020-05-13	V2.3.9	<p>1. Update the user configuration structure in the chapter “RF Reader”.</p> <p>2. Modified the description of the keyword <i>ro.fac.videocard</i>;</p>	Zhang Mengying Huang Ruzhen

		3. Added descriptions of keywords persist.sys.tm.cpu, persist.sys.tm.flash and persist.sys.tm.ram.	
2020-06-30	V2.4.0	Updated “OsTerminalConsumerInfo” and “OsPiccApplePoll” and “persist.sys.usbd.mode” in the Registry.	Zhang Mengying Huang Ruzhen

Table of Contents

1	Introduction	1
1.1	Purpose	1
1.2	Target Readers	1
1.3	Prerequisites	2
1.4	Document Conventions	2
2	Return Code and Parameter.....	4
2.1	Return Code Classification.....	4
2.2	General Return Code	5
2.3	Parameter	6
3	Thread	7
4	System Function	8
4.1	Return Code List	8
4.2	Data Definition.....	9
4.3	Time set	10
4.3.1	OsSetTime	10
4.3.2	OsGetTime	11
4.4	Timer	11
4.4.1	OsTimerSet	11
4.4.2	OsTimerCheck	11
4.5	Delay	12

4.5.1	OsSleep.....	12
4.6	Thread.....	12
4.7	Log	13
4.7.1	OsLogSetTag	13
4.7.2	OsLog.....	13
4.8	Get the Count Value.....	13
4.8.1	OsGetTickCount.....	13
4.9	Get Application Information	14
4.9.1	OsGetAppInfo.....	14
4.9.2	OsGetOptInfo	14
4.10	Buzzer.....	15
4.10.1	OsBeep	15
4.11	Key tone	16
4.11.1	OsSetKeyTone	16
4.12	Run Application	16
4.12.1	OsRunApp.....	16
4.12.2	OsGetAppExitCode	17
4.13	Set and Read the Registry	17
4.13.1	OsRegSetValue.....	17
4.13.2	OsRegGetValue	18
4.14	Install and Uninstall Files.....	19
4.14.1	OsInstallFile.....	19

4.14.2 OsUninstallFile	20
4.15 System Firmware Upgrade.....	21
4.15.1 OsFirmwareGetVersion.....	21
4.15.2 OsFirmwareUpgrade	21
4.16 Verify Signature.....	22
4.16.1 OsVerifySign	22
4.16.2 OsVerifySignExternal	23
4.17 Get System Version	24
4.17.1 OsGetSysVer	24
4.18 Test Whether on the Base.....	24
4.18.1 OsOnBase.....	24
4.19 Save the Crash Report.....	25
4.19.1 OsSaveCrashReport	25
4.20 Mount and Unmount the External File System.....	26
4.20.1 OsMount.....	26
4.20.2 OsUmount	28
4.21 Encrypted File System Interface	29
4.21.1 OsCryptFormat.....	29
4.21.2 OsCryptMount	30
4.21.3 OsCryptUmount.....	31
4.22 LED	31
4.22.1 OsLed.....	31

4.23	Get Terminal Accumulated Usage Information.....	33
4.23.1	OsTerminalConsumelInfo.....	33
4.24	Digital IO	34
4.24.1	OsDigitalIOGetStat.....	34
4.24.2	OsDigitalIOSetStat	34
5	Encryption and Decryption.....	36
5.1	Return Code List	36
5.2	Random Number.....	36
5.2.1	OsGetRandom	36
5.3	SHA Algorithm.....	37
5.3.1	OsSHA	37
5.4	DES Algorithm.....	38
5.4.1	OsDES	38
5.5	AES Algorithm.....	38
5.5.1	OsAES	39
5.6	RSA Algorithm.....	39
5.6.1	OsRSA	39
5.6.2	OsRSAKeyGen	40
6	PED	42
6.1	Return Code List	43
6.2	Data Definition.....	45
6.2.1	Key Type	45

6.2.2	DisplayAttribute of Asterisk.....	46
6.3	Data Structure	46
6.3.1	Structure of RSA Key	46
6.3.2	RSA Key Structure for Verifying the Ciphertext IC Card PIN	46
6.4	Basic PED	47
6.4.1	OsPedOpen.....	47
6.4.2	OsPedGetVer.....	47
6.4.3	OsPedSetInterval	47
6.4.4	OsPedVerifyPlainPin	48
6.4.5	OsPedVerifyCipherPin	49
6.4.6	OsPedEraseKeys	52
6.4.7	OsPedSetFunctionKey	52
6.4.8	OsPedClose	53
6.4.9	OsPedCancelPinEntry.....	53
6.4.10	OsPedSetOfflinePin	53
6.4.11	OsPedEndPinEntry	54
6.4.12	OsPedPinKeyNotify	55
6.5	MK/SK.....	56
6.5.1	OsPedWriteKey.....	56
6.5.2	OsPedWriteTIK	60
6.5.3	OsPedWriteKeyVar	62
6.5.4	OsPedSetAsteriskLayout	63

6.5.5	OsPedSetPinIconLayout	64
6.5.6	OsPedGetPinBlock.....	66
6.5.7	OsPedUpdatePinBlock.....	68
6.5.8	OsPedGetMac.....	69
6.5.9	OsPedDes	70
6.5.10	OsPedGetKcv.....	71
6.5.11	OsPedDeriveKey	73
6.5.12	OsPedSetPinBg	74
6.5.13	OsPedCustomKeypad.....	75
6.6	DUKPT	76
6.6.1	OsPedGetPinDukpt	76
6.6.2	OsPedGetMacDukpt	78
6.6.3	OsPedDesDukpt.....	79
6.6.4	OsPedGetKsnDukpt	81
6.6.5	OsPedIncreaseKsnDukpt	81
6.7	RSA.....	82
6.7.1	OsPedReadRsaKey	82
6.7.2	OsPedWriteRsaKey	82
6.7.3	OsPedRsaRecover.....	83
6.7.4	OsPedReadCipherRsaKey.....	83
6.7.5	OsPedWriteCipherRsaKey	84
6.8	AES	84

6.8.1	OsPedWriteAesKey	84
6.8.2	OsPedAes	87
6.9	SM Algorithm.....	88
6.9.1	OsPedGenSM2Pair	88
6.9.2	OsPedWriteSM2Key	88
6.9.3	OsPedSM2Sign.....	89
6.9.4	OsPedSM2Verify	90
6.9.5	OsPedSM2Recover.....	91
6.9.6	OsPedSM3.....	92
6.9.7	OsPedSM4.....	93
6.9.8	OsPedGetMacSM	94
6.9.9	OsPedGetPinBlockSM4	95
6.10	DES FIRE.....	96
6.10.1	OsPedDFAuthDiver	96
6.10.2	OsPedDFAuthMerge	97
6.11	RKI	98
6.11.1	OsPedRkiInjectKey	98
7	LCD	99
7.1	OsScrContrast.....	101
7.2	OsScrBrightness	101
7.3	OsScrGetSize	102
8	Keyboard	103

8.1	OsKbBacklight.....	105
9	Touch Screen	107
10	Signature Pad	108
11	Printer	109
11.1	Return Code List	109
11.2	Open and Close	110
11.2.1	OsPrnOpen	110
11.2.2	OsPrnReset.....	110
11.2.3	OsPrnClose	111
11.3	Printer Settings.....	111
11.3.1	OsPrnSetSize.....	111
11.3.2	OsPrnSetDirection.....	111
11.3.3	OsPrnSetGray	112
11.4	Type Setting	112
11.4.1	OsPrnSetSpace.....	112
11.4.2	OsPrnSetReversal.....	113
11.4.3	OsPrnSetIndent.....	113
11.4.4	OsPrnCheck	113
11.4.5	OsPrnGetDotLine	114
11.4.6	OsPrnSetFont.....	114
11.4.7	OsPrnSelectFontSize	114
11.4.8	OsPrnFeed.....	115

11.4.9 OsPrnPrintf.....	116
11.4.10 OsPrnPutImage	116
11.4.11 OsPrnStart	117
11.4.12 OsPrnClrBuf.....	118
11.4.13 OsPrnSetParam.....	118
11.5 POSIX.....	118
11.5.1 Open	119
11.5.2 Read.....	119
11.5.3 Write.....	119
11.5.4 Close	120
12 Font Library.....	121
12.1 Data Structure	121
12.2 Font Operation	122
12.2.1 OsEnumFont	122
12.2.2 OsOpenFont.....	122
12.2.3 OsCloseFont	123
12.2.4 OsGetFontDot	123
13 Code	126
13.1 Code Conversion	126
13.1.1 OsCodeConvert.....	126
14 MSR.....	128
14.1 Return Code List	128

14.2	Data Structure	129
14.3	MSR Control Interface.....	129
14.3.1	OsMsrOpen	129
14.3.2	OsMsrClose.....	130
14.3.3	OsMsrReset	130
14.3.4	OsMsrSwiped	130
14.3.5	OsMsrRead	131
14.3.6	OsMsrReadJIS	131
15	IC Card Reader.....	133
15.1	Return Code List	133
15.2	Data Structure	134
15.2.1	APDU Request Structure.....	134
15.2.2	APDU Response Structure.....	135
15.3	Encapsulated Interfaces.....	136
15.3.1	OslccOpen.....	136
15.3.2	OslccDetect.....	136
15.3.3	OslccInit	137
15.3.4	OslccExchange	139
15.3.5	OslccClose	140
16	RF Reader	141
16.1	Return Code List	141
16.2	Data Structure	142

16.2.1 User Configuration Structure	142
16.3 Encapsulate Interfaces.....	144
16.3.1 OsPiccOpen	144
16.3.2 OsPiccClose.....	144
16.3.3 OsPiccResetCarrier.....	145
16.3.4 OsPiccPoll	145
16.3.5 OsPiccAntiSel.....	145
16.3.6 OsPiccActive	146
16.3.7 OsPiccTransfer.....	147
16.3.8 OsPiccRemove.....	147
16.3.9 OsMifareAuthority.....	147
16.3.10 OsMifareOperate	148
16.3.11 OsPiccInitFelica	149
16.3.12 OsPiccIsoCommand	149
16.3.13 OsPiccSetUserConfig	150
16.3.14 OsPicc GetUserConfig	150
16.3.15 OsPiccApplePoll	150
16.3.16 OsPiccOffCarrier.....	151
16.3.17 OsPiccInitIso15693	151
16.4 Notes of Touch Screen and RF Reader Programming.....	152
17 Communication Port	153
17.1 Data Definition.....	153

17.2	Communication Control.....	154
17.2.1	OsPortOpen	154
17.2.2	OsPortClose	156
17.2.3	OsPortSend.....	156
17.2.4	OsPortRecv	157
17.2.5	OsPortReset.....	158
17.2.6	OsPortCheckTx	158
17.2.7	OsPortCheckRx.....	159
17.3	POSIX Interface	159
17.3.1	Open	160
17.3.2	Read.....	160
17.3.3	Write.....	160
17.3.4	Close	161
17.3.5	Query the Buffer Data of Communication Port	161
17.3.6	Clear the Buffer Data of Communication Port	161
17.3.7	Set the Configuration Parameters of Communication Port.....	161
18	MODEM.....	165
18.1	Return Code List	165
18.2	Data Structure	173
18.3	OsModemOpen.....	177
18.4	OsModemClose	178
18.5	OsModemSwitchPower.....	178

18.6	OsModemConnect	178
18.7	OsModemCheck	180
18.8	OsModemExCmd	181
18.9	OsModemHangup	182
18.10	OsModemSend	182
18.11	OsModemRecv	183
18.12	OsPppomLogin	184
18.13	OsPppomCheck	186
18.14	OsPppomLogout	186
19	Network Communication.....	187
19.1	TCP Programming	187
19.2	UDP Programming	190
20	Network Configuration	193
20.1	Return Code List	193
20.2	Data Definition.....	195
20.2.1	Physical Channel List	195
20.2.2	IPv6 Status Value List	196
20.2.3	Data Structure	196
20.3	IPv4 Network Configuration Interface.....	198
20.3.1	OsNetAddArp	198
20.3.2	OsNetPing	199
20.3.3	OsNetPingEx.....	199

20.3.4 OsNetDns.....	200
20.3.5 OsNetDnsEx.....	201
20.3.6 OsNetSetConfig	202
20.3.7 OsNetGetConfig	203
20.3.8 OsNetStartDhcp	204
20.3.9 OsNetCheckDhcp.....	205
20.3.10 OsNetStopDhcp	205
20.3.11 OsPppoeLogin	206
20.3.12 OsPppoeCheck.....	206
20.3.13 OsPppoeLogout.....	206
20.3.14 OsNetSetRoute.....	207
20.3.15 OsNetGetRoute	207
20.3.16 OsNetSetRouteTable	208
20.3.17 OsNetDelRouteTable	208
20.3.18 OsNetGetRouteTable.....	209
20.3.19 OsNetNat	210
20.4 IPv4/IPv6 Network Configuration Interface.....	211
20.4.1 OsNetSetDns	211
20.5 IPv6 Network Configuration.....	211
20.5.1 OsNetPing6	211
20.5.2 OsNetSetIPv6Addr	212
20.5.3 OsNetGetIPv6Addr.....	213

20.5.4 OsNetGetRouteAdvertise6	214
20.5.5 OsNetStartDhcp6	215
20.5.6 OsNetCheckDhcp6.....	216
20.5.7 OsNetStopDhcp6	216
20.5.8 OsNetSetRouteTable6	217
20.5.9 OsNetGetRouteTable6.....	218
21 GPRS/CDMA	219
21.1 Return Code List	219
21.2 Wireless Module Interface.....	220
21.2.1 OsWILock.....	220
21.2.2 OsWIUnLock	221
21.2.3 OsWIInit.....	221
21.2.4 OsWIInitEx.....	222
21.2.5 OsWISwitchPower.....	223
21.2.6 OsWISwitchSleep.....	224
21.2.7 OsWIGetSignal.....	225
21.2.8 OsWICheck	226
21.2.9 OsWILogin.....	226
21.2.10 OsWILoginEx	229
21.2.11 OsWILogout	231
21.3 Wireless Module Information Settings.....	232
21.3.1 OsWISelSim	232

22	WiFi	233
22.1	Return Code List	233
22.2	Encryption Type List.....	234
22.3	Data Structure	234
22.4	OsWifiOpen.....	237
22.5	OsWifiClose	237
22.6	OsWifiSwitchPower.....	238
22.7	OsWifiScan	238
22.8	OsWifiConnect	239
22.9	OsWifiDisconnect.....	240
22.10	OsWifiCheck	240
22.11	OsWifiCmd.....	242
22.12	OsWifiWpsConnect.....	242
23	GPS	244
23.1	Return Code List	244
23.2	Data Definition.....	244
23.3	OsGpsOpen	245
23.4	OsGpsRead	245
23.5	OsGpsClose.....	246
24	Base	247
24.1	Introduction	247
24.2	Macro Definition	248

24.3 API	248
24.3.1 OsOnBase.....	248
24.3.2 OsBaseOpen.....	248
24.3.3 OsBaseCheck	249
24.3.4 OsBaseClose	250
24.3.5 OsBaseScan	250
24.3.6 OsBaseConnect	251
24.3.7 OsBaseDisconnect.....	252
24.3.8 OsCheckPortStatus.....	253
24.3.9 OsGetBaseInfo.....	253
24.3.10 OsBaseCmd.....	254
25 File System	256
26 System Information.....	257
27 Audio	260
27.1 Return Code List	260
27.2 OsRecordOpen	260
27.3 OsRecordStart	261
27.4 OsRecordStop.....	263
27.5 OsRecordCheck.....	263
27.6 OsRecordClose.....	264
27.7 OsPlayWave	264
27.8 OsStopPlayWave	265

27.9	OsPlayAudio	266
27.10	OsStopPlayAudio.....	267
28	Barcode and Camera.....	268
28.1	General Definition	268
28.2	Data Definition.....	269
28.2.1	Macro Definition of Camera.....	269
28.2.2	Image Resolution Macro Definition of Photography	269
28.2.3	Image Data Format Macro Definition of Photography	270
28.2.4	The Attribute Structure of the Photography Parameter	270
28.2.5	Return Code List	270
28.3	Basic Interface	271
28.3.1	OsScanSetParam.....	271
28.3.2	OsScanOpen.....	272
28.3.3	OsScanRead	272
28.3.4	OsScanClose	273
28.3.5	OsCameraOpen	273
28.3.6	OsCameraClose	273
28.3.7	OsCameraGetParam.....	274
28.3.8	OsCameraSetParam	274
28.3.9	OsCameraCapture	275
28.3.10	OsScanDecodeBuf	276
28.3.11	OsCameraDetectMotion.....	277

29	Power Management.....	279
29.1	Return Code List	279
29.2	Data Structure	279
29.3	OsCheckBattery	281
29.4	OsCheckPowerSupply	282
29.5	OsSysSleep	283
29.6	OsSysSleepEx	284
29.7	OsSysSleepTime	285
29.8	OsReboot.....	285
29.9	OsPowerOff.....	286
29.10	OsPmGetEvent.....	286
29.11	OsPmRequest.....	287
29.12	OsWakeupSource.....	287
29.13	OsCheckPowerStatus.....	288
29.14	OsCheckBMSMode	289
29.15	Get the Power Management Information	290
29.16	Client Sends Request	290
	Appendix 1 PIN Block Format.....	291
	Appendix 2 EPS PINBLOCK Format	295
	Appendix 3 Registry	296
	Appendix 4 Validity of Models and Contents	305

Table List

Table 2.1 Return code classification list	4
Table 2.2 General return code list	5
Table 4.1 System function return code list.....	8
Table 4.2 Definitions of file types.....	8
Table 5.1 Encryption and decryption return code list.....	36
Table 6.1 PED return code list.....	43
Table 6.2 Key types.....	45
Table 6.3 Layout attributes of asterisk.....	46
Table 6.4 Color values of asterisk	46
Table 11.1 Printer return code list	109
Table 14.1MSR return code list	128
Table 15.1 IC Card reader return code list	133
Table 16.1 Return code list.....	141
Table 17.1 Macro definition list of communication ports.....	153
Table 17.2 Return code list of USB port functions.....	153
Table 18.1 Modem return code list	165
Table 18.2 Variable definitions of ST_MODEM_SETUP	174
Table 20.1 Network configuration return code list	193
Table 20.2 Physical channel list	195
Table 20.3 IPv6 status value list	196

Table 21.1 GPRS/CDMA return code list	219
Table 22.1 Return code list.....	233
Table 22.2 Encryption type list	234
Table 27.1 Return code list.....	260
Table 28.1 Camera definition list	269
Table 28.2 Image resolution definition list	269
Table 28.3 Image data format definition list.....	270
Table 28.4 Return code list.....	270

1 Introduction

1.1 Purpose

This documentation introduces the framework of Prolin applications and the key points of GUI and other important frameworks. It provides programming guide for API interfaces compatible with controlling hardware based on Prolin OS and offers corresponding instructions conducive to a successful design for developers.

1.2 Target Readers

This documentation is targeted at Prolin OS developers who are committed to developing Prolin local applications.

Prolin provides a series of interfaces based on Linux system calls and POSIX interfaces. And a set of OSAL interfaces, beginning with the prefix “Os”, are encapsulated for the compatibility demand of some background services and applications. In addition, demo programming codes of POSIX interfaces and system libraries are illustrated in this documentation to guide developers to access device or system function.

In this documentation, all the interfaces beginning with “Os” are defined in *osal.h* of SDK, unless otherwise specified.

Prolin SDK provides necessary tools and resources for local Prolin applications. Different from the background system applications, Prolin SDK can run independently as executive programs on Prolin OS-based devices. Local applications enjoy a great deal of privileges. They can access all Prolin OS features, save data in local file system and even communicate with other installed programs through a special URL.

1.3 Prerequisites

Basic knowledge of Linux is necessary before reading this documentation. Related notions include:

- Process, thread and Linux system functions and their roles in application development;
- Memory management, including how to create and release an application;
- GUI application framework;
- Linux input subsystem and Framebuffer driver interface.

It is necessary to download and install Prolin SDK before developing Prolin applications. With regard to Linux system, Linux 2.6.22 or higher version is required; while for Windows system, Windows XP or higher version is required. For more information about Prolin SDK, please visit PAX Partners Network (<https://support.paxsz.com/>) or send email to support@paxsz.com if you haven't registered.

1.4 Document Conventions

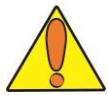
Conventions used in this document and their corresponding meanings are listed below:

NOTE



Giving notes.

CAUTION



Caution for general problems.

WARNING



Warning for crucial problems.

2Return Code and Parameter

2.1 Return Code Classification

Table 2.1 lists the types and value ranges of return codes involved in this document:

Table 2.1 Return code classification list

Type	Value(Decimalism)	Description
General Return Value	-1000~-1999	<i>General error code of API.</i>
System Function	-2200~-2299	
Power Management	-2300~-2399	
Encryption and Decryption	-2400~-2499	
Font Library	-2500~-2599	
LED display	-2600~-2699	
MSR	-2700~-2799	
IC Card Reader	-2800~-2899	

Contactless IC Card Reader	-2900~-2999
Communication Port	-3000~-3099
MODEM	-3100~-3299
IP Network Configuration	-3300~-3399
PED	-3800~-3899

2.2 General Return Code

Table 2.2 lists the macro definitions, values and corresponding descriptions of the general return values in this documentation.

Table 2.2 General return code list

Macro	Value	Description
RET_OK	0	Succeeded.
ERR_INVALID_HANDLE	-1000	Invalid handle.
ERR_TIME_OUT	-1001	Timeout.
ERR_APP_NOT_EXIST	-1002	Application does not exist.
ERR_INVALID_PARAM	-1003	Invalid parameter.
ERR_DEV_NOT_EXIST	-1004	Device does not exist.
ERR_DEV_BUSY	-1005	Device is busy.
ERR_DEV_NOT_OPEN	-1006	Device is not open.
ERR_ACCESS_DENY	-1007	Access denied.
ERR_FONT_NOT_EXIST	-1008	Font does not exist.
ERR_FILE_FORMAT	-1009	File format error.
ERR_USER_CANCEL	-1010	Canceled by user.
ERR_NO_PORT	-1011	No communication port.
ERR_NOT_CONNECT	-1012	Not Connected
ERR_MEM_FAULT	-1013	Memory error.
ERR_SYS_BAD	-1014	System configuration error.
ERR_SYS_NOT_SUPPORT	-1015	System does not

		<i>support this function.</i>
ERR_STR_LEN	-1016	<i>Character string is too long.</i>
ERR_TOO_MANY_HANDLE	-1017	<i>Too many handles</i>
ERR_APP_MODE	-1018	<i>Mode error.</i>
ERR_FILE_NOT_EXIST	-1019	<i>File does not exist.</i>
ERR_TS_DISABLE	-1020	<i>Touch screen is not open.</i>
ERR_FONT_CODE	-1021	<i>Character encoding error.</i>
ERR_VERSION_TOO_LOW	-1022	<i>The version is too low.</i>
ERR_BATTERY_ABSENT	-1023	<i>Battery is absent.</i>
ERR_BATTERY_VOLTAGE_TO_O_LOW	-1024	<i>Battery voltage is too low.</i>

2.3 Parameter

There are two types of API parameters: input parameter and output parameter. They are distinguished with identifiers in this documentation.

All the input and output string parameters end with "\x00" and valid string length must be clarified in parameter description.

3 Thread

Prolin supports *pthread* of POSIX. The header file of *pthread* is located at the installation directory of ProlinBuilder:
toolchains\arm-linux\arm-softfloat-linux-gnu\include\pthread.

4 System Function

4.1 Return Code List

Table 4.1 System function return code list

Macro	Value	Description
ERR_FILE_NOT_FOUND	-2201	<i>File is not found.</i>
ERR_VERIFY_SIGN_FAIL	-2204	<i>Fail to verify signature.</i>
ERR_NO_SPACE	-2205	<i>System space is not enough.</i>
ERR_NEED_ADMIN	-2207	<i>Need higher permission.</i>
ERR_PUK_NOT_EXIST	-2208	<i>PUK doesn't exist.</i>
ERR_OS_VERSION_TOO_LOW	-2209	<i>System version is too low.</i>
ERR_INVALID_PARAM	-1003	<i>Invalid parameter.</i>
ERR_DEV_NOT_EXIST	-1004	<i>Device doesn't exist.</i>
ERR_DEV_BUSY	-1005	<i>Device is busy.</i>
ERR_ACCESS_DENY	-1007	<i>Access denied.</i>
ERR_SYS_NOT_SUPPORT	-1015	<i>System does not support.</i>
ERR_FILE_NOT_EXIST	-1019	<i>File doesn't exist.</i>

Table 4.2 Definitions of file types

Macro	Value	Description
<code>FILE_TYPE_APP</code>	1	<i>Application package.</i>
<code>FILE_TYPE_APP_PARM</code>	2	<i>Application data file.</i>
<code>FILE_TYPE_SYS_LIB</code>	3	<i>System dynamic library file.</i>
<code>FILE_TYPE_PUB_KEY</code>	4	<i>Public key file.</i>
<code>FILE_TYPE_AUP</code>	5	<i>Application upgrade package.</i>

4.2 Data Definition

LOG_T(LOG type enumeration):

LOG_T:

```
typedef enum{
    LOG_DEBUG,      /* Display debugging message */
    LOG_INFO,       /* Display prompt message */
    LOG_WARN,       /* Display warning message */
    LOG_ERROR,      /* Display error message */
} LOG_T;
```

ST_TIME(time structure):

ST_TIME:

```
typedef struct{
    int Year;        /*Year 1970– 2037*/
    int Month;       /*Month 1 –12*/
    int Day;         /*Day 1 –31*/
    int Hour;        /*Hour 0 – 23*/
    int Minute;      /*Minute 0 –59*/
    int Second;      /*Second 0 –59*/
    int DayOfWeek;   /*Monday–Sunday (valid only in read time)*/
} ST_TIME;
```

ST_TIMER(timer structure):

ST_TIMER:

```
typedef struct{
    unsigned long Start;
    unsigned long Stop;
    unsigned long TimeLeft;
} ST_TIMER;
```

ST_APP_INFO(application information structure):**ST_APP_INFO:**

```
typedef struct{
    char Id[64];
    char Name[64];
    char Bin[64];
    char Artwork[64];
    char Desc[64];
    char Vender[32];
    char Version[32] ;
} ST_APP_INFO;
```

ST_OPT_INFO(optional system component information structure):**ST_OPT_INFO:**

```
typedef struct {
    char Name[64];
    char Version[32];
} ST_OPT_INFO;
```

4.3 Time set

4.3.1 OsSetTime

Prototype	int OsSetTime(const ST_TIME *Time);	
Function	Set system time and date. Prolin system does not contain time zone.	
Parameters	Time[Input]	A pointer to ST_TIME structure that contains system time. <i>DayOfWeek</i> is ignored in ST_TIME structure; it can't be NULL.
Return	RET_OK	Succeeded.
	ERR_NEED_ADMIN	Permission denied.

	ERR_INVALID PARA M	Invalid parameter.
Instruction		Only the main application has permission to set time; otherwise, it will fail and return ERR_NEED_ADMIN.

4.3.2 OsGetTime

Prototype	void OsGetTime(ST_TIME *Time);	
Function	Get system time and date.	
Parameters	Time[Output]	A pointer to ST_TIME structure that contains system time. <i>DayOfWeek</i> is ignored in ST_TIME structure; it can't be NULL.
Return	None.	
Instruction		

4.4 Timer

4.4.1 OsTimerSet

Prototype	int OsTimerSet(ST_TIMER *Timer, unsigned long Ms);	
Function	Create a timer with a specified timeout value.	
Parameters	Timer[Output]	A pointer to ST_TIMER structure. It can't be NULL.
	Ms[Input]	Timeout.[unit:ms].
Return	RET_OK	Succeeded.
	ERR_INVALID_PARAM	Invalid parameter.
Instruction	The timer is irrelevant to system time, and it will be paused when system enters sleep mode.	

4.4.2 OsTimerCheck

Prototype	unsigned long OsTimerCheck(ST_TIMER *Timer);	
Function	Check the remaining time of a specified timer.	
Parameters	Timer[Input]	Timer structure ST_TIMER .
Return	>=0	Remaining time. [unit:ms]
	ERR_INVALID_PARAM	Invalid parameter.

	M
Instruction	

4.5 Delay

4.5.1 OsSleep

Prototype	void OsSleep(unsigned int Ms);
Function	Suspend the current process or thread.
Parameters	Ms [Input] Delay time.[unit:ms]
Return	None
Instruction	

4.6 Thread

Prolin thread is implemented by standard linux POSIX interface. Please refer to the following code:

Example:

```
#include <pthread.h>
static pthread_t ntid;
static void *thread_fn(void *arg)
{
    printf("This is child thread\n");
    return ((void *)0);
}
int main()
{
    printf("This is main thread\n");
    if(pthread_create(&ntid,NULL,thread_fn,NULL) != 0)
        printf("can't create thread\n");
    sleep(5);
    return 0;
}
```

4.7 Log

4.7.1 OsLogSetTag

Prototype	void OsLogSetTag(const char *Tag);
Function	Set LOG information tag.
Parameters	Tag[Input] LOG information tag.
Return	None.
Instruction	<ol style="list-style-type: none"> 1. Call this function to set LOG tag; the default tag is NULL. 2. It is recommended to set tag to be an application name. 3. The valid <i>Tag</i> length ranges from 0 to 32 bytes; only the first 32 bytes will be used if <i>Tag</i> length exceeds 32 bytes. 4. OsLog() will not save LOG information for a NULL tag or an empty string.

4.7.2 OsLog

Prototype	int OsLog(LOG_T Prio, const char *fmt, ...);
Function	Save LOG information.
Parameters	Prio[Input] <u>LOG_T</u> structure, type of LOG information. fmt [Input] Format of LOG information.
Return	RET_OK Succeeded. ERR_INVALID_PARAM Invalid parameter.
Instruction	OsLogSetTag() must be called to set tag before calling this function; otherwise, OsLog() will fail to save LOG information.

4.8 Get the Count Value

4.8.1 OsGetTickCount

Prototype	unsigned long OsGetTickCount(void);
Function	Get the cumulative running time from startup to present time, excluding sleeping time.[unit:ms]
Parameters	None
Return	>0 Count value.

Instruction	
-------------	--

4.9 Get Application Information

4.9.1 OsGetAppInfo

Prototype	int OsGetAppInfo(ST_APP_INFO AppInfo[], int InfoCnt);	
Function	Get information of installed applications.	
Parameters	AppInfo[Output]	A pointer to application information structure of ST_APP_INFO array. It cannot be NULL.
	InfoCnt[Input]	The number of App that can be stored in <i>AppInfo</i> array.
Return	>=0	The number of App information that has been obtained.
	ERR_NEED_ADMIN	Permission denied.
	ERR_INVALID_PARAMETER	Invalid parameter.
Instruction	<ol style="list-style-type: none"> Only the main application has permission to get App information.(Note: This limitation is only valid for Prolin-2.4 platform, sub-application on other platform can also get APP information.) When the real number of application is larger than <i>InfoCnt</i>, only the first <i>InfoCnt</i> number of applications will be obtained. 	

4.9.2 OsGetOptInfo

Prototype	int OsGetOptInfo(ST_OPT_INFO OptInfo[], int InfoCnt);	
Function	Get information of installed optional system components.	
Parameters	OptInfo [output]	A pointer to application information structure of ST_OPT_INFO array. It cannot be NULL.
	InfoCnt [input]	The number of optional system components that can be stored in <i>OptInfo</i> array.
Return	>=0	The number of optional system components information that has been

	<p>obtained.</p> <p>ERR_FILE_NOT_FOUND The file path that stores optional system components does not exist.</p> <p>ERR_INVALID_PARAMETER Invalid parameter.</p>
Instruction	When the real number of optional system components is larger than <i>InfoCnt</i> , only the first <i>InfoCnt</i> number will be obtained.

4.10 Buzzer

4.10.1 OsBeep

Prototype	void OsBeep(int ToneType, int DurationMs);																	
Function	Set the frequency and duration of buzzer.																	
Parameters	ToneType [Input]	Tone type. The valid <i>ToneType</i> ranges from 1 to 7.																
	DurationMs [Input]	Duration.[unit:ms] The valid <i>DurationMs</i> ranges from 10 to 10000. If <i>DurationMs</i> <10, it will be set to 10 by default; If <i>DurationMs</i> >10000, it will be set to 10000 by default.																
Return	None.																	
Instruction	<p>Tone type and corresponding frequency:</p> <table border="1"> <thead> <tr> <th>Tone type</th> <th>Buzzer frequency(Hz)</th> </tr> </thead> <tbody> <tr> <td>≤1</td> <td>1680</td> </tr> <tr> <td>2</td> <td>1850</td> </tr> <tr> <td>3</td> <td>2020</td> </tr> <tr> <td>4</td> <td>2130</td> </tr> <tr> <td>5</td> <td>2380</td> </tr> <tr> <td>6</td> <td>2700</td> </tr> <tr> <td>≥7</td> <td>2750</td> </tr> </tbody> </table>		Tone type	Buzzer frequency(Hz)	≤1	1680	2	1850	3	2020	4	2130	5	2380	6	2700	≥7	2750
Tone type	Buzzer frequency(Hz)																	
≤1	1680																	
2	1850																	
3	2020																	
4	2130																	
5	2380																	
6	2700																	
≥7	2750																	

4.11 Key tone

4.11.1 OsSetKeyTone

Prototype	void OsSetKeyTone(int OnOff, int DutyCycle);	
Function	Set the key tone switch and duty cycle.	
Parameters	OnOff [Input]	0 represents turn off the keytone; 1 represents turn on the keytone.
	DutyCycle[Input]	Keytone duty cycle, the ranges is [1,99]. If DutyCycle <1, or DutyCycle >99, keep the same DutyCycle. If DutyCycle=50, the volume is the maximum.
Return	None	
Instruction		

4.12 Run Application

4.12.1 OsRunApp

Prototype	int OsRunApp(char *AppId, char **Argv, void *Data, RUNAPP_CB CbOut, RUNAPP_CB CbErr);	
Function	Switch to a specified sub-application.	
Parameters	AppId[Input]	Sub-application ID.
	Argv[Input]	Input parameter list of sub-application. If not needed, Argv can be set to NULL.
	Data[Input]	Customized data, which will be passed to CbOut() and CbErr() for calling back.
	CbOut[Input]	Standard output message callback function.
	CbErr[Input]	Standard error message callback function.
Return	RET_OK	Succeeded.
	ERR_APP_NOT_EXIST	Sub-application does not exist.

	<p>ERR_ACCESS_DENY Access denied.</p> <p>ERR_APP_MODE Mode error.</p> <p>ERR_INVALID_PARAM Invalid parameter.</p> <p>ERR_NEED_ADMIN Permission denied.</p>
Instruction	<ol style="list-style-type: none"> Only the main application has permission to switch application; otherwise, ERR_NEED_ADMIN will be returned. OsRunApp() will switch to the sub-application specified by <i>AppId</i>; the sub-application cannot be the main application; otherwise, ERR_INVALID_PARAM will be returned. The standard output message and standard error message of sub-application will be exported to CbOut() and CbErr(), respectively; if there are multiple lines of messages, the corresponding callback function will be called for several times. The callback function is defined as: <pre>typedef void (*RUNAPP_CB)(char *appid, char *str, void *data).</pre>

4.12.2 OsGetAppExitCode

Prototype	<code>int OsGetAppExitCode(void);</code>
Function	Get the exit code of sub-application.
Parameters	None
Return	The exit code of sub-application
Instruction	Users can call this function after OsRunApp() to get the exit code of sub-application.

4.13 Set and Read the Registry

4.13.1 OsRegSetValue

Prototype	<code>int OsRegSetValue(const char *Key, const char *Value);</code>	
Function	Set value of the specified key in registry.	
Parameters	Key [Input]	System configuration name, beginning with “persist.sys.” , “rt.sys.” or “rt.app.” and ending with “\0”. The valid string length ranges from 0

		to 31 bytes.
	Value [Input]	Parameter value, ending with “\0”. The valid string length ranges from 0 to 64 bytes, and it cannot be NULL.
Return	RET_OK	Succeeded.
	ERR_INVALID PARA M	Invalid Parameter.
	ERR_NEED_ADMIN	Permission denied.
	ERR_SYS_NOT_SUP PORT	System doesn't support this configuration name.
Instruction	<ol style="list-style-type: none"> 1. This function can only set the system configuration whose name begins with “persist.sys” , “rt.sys” or “rt.app.”, such as “persist.sys.app0.pic”. For more information, refer to Appendix 3 Registry. 2. OsRegSetValue()supports both the keywords listed in Appendix 3 and user-defined configurations. 3. Registry beginning with “rt.sys.” and “rt.app.” will be invalid when POS restarts, if the original state needs to be restored, it requires to be reset after restarting. 4. After setting the key value, it will take about 200 milliseconds for the key value to take effect. 	

4.13.2 OsRegGetValue

Prototype	int OsRegGetValue(const char *Key, char *Value);	
Function	Get value of the specified key in registry.	
Parameters	Key [Input]	System configuration name, ending with“\0”.
	Value [Output]	Parameter value. The valid string length is more than 64 bytes.
Return	>=0	String length.
	ERR_INVALID PARA M	Invalid Parameter.
Instruction	<ol style="list-style-type: none"> 1. This function can only get the system configuration whose name begins “ro.fac.” , “rt.sys.” , “rt.app.” or “persist.sys.”. For more information, refer to Appendix 3 Registry. 2. If the inquired parameter does not exist or is empty, it'll return 0 and the output parameter <i>Value</i> will be “ ”. 	

4.14 Install and Uninstall Files

4.14.1 OsInstallFile

Prototype	<code>int OsInstallFile(const char *Name, const char *FileName, int FileType);</code>	
Function	Install or update application software, application data, OPT package, public user key and firmware of external device (FWP package).	
Parameters	Name[Input] FileName[Input] FileType	<p>Name[Input] Pathname of the specified destination.</p> <p>FileName[Input] File path to be installed. The path includes the file name,it cannot be NULL.</p> <ul style="list-style-type: none"> ▪ FILE_TYPE_APP (application package or AIP) ▪ FILE_TYPE_APP_PARAM (application data file) ▪ FILE_TYPE_SYS_LIB (dynamic library and other optional system components) ▪ FILE_TYPE_PUB_KEY (user public key file) ▪ FILE_TYPE_AUP(application upgrade package) ▪ FILE_TYPE_FWP (device firmware package)
Return	<p>RET_OK Succeeded.</p> <p>ERR_PUK_NOT_EXIST The specified user public key does not exist.</p> <p>ERR_FILE_NOT_FOUND FileName does not exist.</p> <p>ERR_FILE_FORMAT FileName format error.</p> <p>ERR_INVALID_PARAM Invalid parameter</p> <p>ERR_VERIFY_SIGN_FAIL Signature error.</p> <p>ERR_APP_MODE Mode error</p>	
Instruction	<p>1. <i>Name</i> is valid only when <i>FileType</i> is FILE_TYPE_APP_PARAM; otherwise, it is invalid. <i>Name</i> is the application name in “MAINAPP” and other terminals, if</p>	

	<p>the application name doesn't exist, the installation or update will fail.</p> <ol style="list-style-type: none"> 2. When <i>FileType</i> is FILE_TYPE_FWP, if it is the device firmware of wireless module, models with battery require at least 2 bars of power to ensure the normal installation of the firmware. 3. Only the main application has permission to do the installing or update operation. 4. When the <i>FileType</i> is FILE_TYPE_FWP, the return code can be RET_OK, ERR_FILE_NOT_FOUND, ERR_VERIFY_SIGN_FAIL, ERR_ACCESS_DENY, ERR_NO_SPACE or ERR_APP_MODE; the ERR_APP_MODE will be returned when mmap(), fork() or execl() fails, or the updater in the FWP package returns an error code.
--	--

4.14.2 OsUninstallFile

Prototype	<code>int OsUninstallFile(const char *AppName, int FileType);</code>	
Function	Uninstall a specified application or an OPT package.	
Parameters	AppName[Input]	<ul style="list-style-type: none"> ▪ When <i>FileType</i> is FILE_TYPE_APP, <i>AppName</i> refers to the ID of application to be uninstalled. ▪ When <i>FileType</i> is FILE_TYPE_SYS_LIB, <i>AppName</i> refers to the name of OPT package.
	FileType	<ul style="list-style-type: none"> ▪ FILE_TYPE_APP (application package, including all installed files.) ▪ FILE_TYPE_SYS_LIB (OPT package, including all files in OPT package.)
Return	RET_OK ERR_APP_NOT_EXIST	<p>Succeeded.</p> <p>The application or OPT package specified by <i>AppName</i> does not exist.</p>
	ERR_FONT_NOT_EXIST	Font library does not exist.
Instruction	<ol style="list-style-type: none"> 1. This function is only used to uninstall application and OPT package that have been installed by user. For more information, refer to application package management in <i>Prolin2.X User Guide</i>. 2. Only the main application has permission to do the uninstalling operation. 	



CAUTION
After calling this function, it is recommended to prompt user to restart the terminal to complete the uninstallation.

4.15 System Firmware Upgrade

4.15.1 OsFirmwareGetVersion

Prototype	int OsFirmwareGetVersion(char *Version, int Size);	
Function	Get system firmware version.	
Parameters	Version[output]	Firmware version buffer.
	Size[Input]	Length of version buffer, the recommended length is 64 bytes.
Return	RET_OK Succeeded. ERR_INVALID_PARAMETER Invalid parameter.	
Instruction	1. The format of version information is MAIN_VERSION-SVN_VERSION, such as in "2.6.26-r1789", "2.6.26" is the MAIN_VERSION and "r1789" is the SVN_VERSION. 2. Whether the firmware needs to be upgraded or not can be determined by SVN_VERSION.	

4.15.2 OsFirmwareUpgrade

Prototype	int OsFirmwareUpgrade(const char *FwFileName);	
Function	Upgrade system firmware.	
Parameters	FwFileName[Input]	Firmware filename, in ZIP format.
	RET_OK	Update succeeded.
Return	ERR_FILE_NOT_FOUND	<i>FwFileName</i> file doesn't exist.
	ERR_VERIFY_SIGN_FAIL	Signature error.
	ERR_SYS_NOT_SUPPORTED	Incompatible with OS.
Instruction	1. Only the main application has permission to upgrade system firmware.	

2. In Prolin-A.B.C, A means framework version, B means major version and C means minor version. The framework version and major version of firmware to be upgraded must be the same as current firmwares', because Prolin-2.4, Prolin-phoenix-2.5 and Prolin-cygnus-2.6 are not compatible with each other; otherwise, ERR_SYS_NOT_SUPPORT will be returned.
3. This function will keep blocking during the upgrading process. If addition of interface prompt is required, start another process.
4. The power can not be cut off during the upgrading process; otherwise, the device will fail to work normally.
5. The upgrade will take effect only after reboot.
6. Models with battery require at least 2 bars of power to ensure the normal upgrade of the system firmware.

4.16 Verify Signature

Prolin provides signature verification functions.

4.16.1 OsVerifySign

Prototype	<code>int OsVerifySign(const char *FileName, int PUKType);</code>	
Function	Verify the file signature specified by <i>FileName</i> , including the signature data.	
Parameters	<p>FileName [Input]</p> <p>PUKType[Input]</p>	<p>Path of file whose signature needs to be verified. Signature data is contained in this file.</p> <ul style="list-style-type: none"> ▪ PUK_TYPE_M Public key of manufacturers, which is used to verify signature for the firmware released by manufacturer. ▪ PUK_TYPE_US_PUK Public key of user signature certificate, which is used to verify signature for the public key certificate. ▪ PUK_TYPE_USER Public key of users, which is used to verify signature for user's application.
Return	RET_OK	Succeeded.
	ERR_VERIFY_SIGN_FAIL	Invalid signature.

	ERR_FILE_NOT_EXIST	File does not exist.
	ERR_DEV_BUSY	Device is busy.
	ERR_INVALID_PARAM	Invalid parameter.
Instruction	<ol style="list-style-type: none"> This function is only used to verify application parameter files, for example, to verify the root certificate defined by the application. Before installing file, it is not recommended to call this function to verify the legitimacy of the file so that to avoid a repeated verification, as OsInstallFile() will verify the file automatically. 	

4.16.2 OsVerifySignExternal

Prototype	<code>int OsVerifySignExternal(const char *FileName, const void *SignData, int PUKType);</code>	
Function	Verify file signature with specified signature data.	
Parameters	FileName [Input]	Path of file whose signature needs to be verified. The file cannot contain signature data; otherwise, the verification will fail.
	SignData[Input]	Signature data. The valid length is 284 bytes.
	PUKType[Input]	<ul style="list-style-type: none"> ▪ PUK_TYPE_M Public key of manufacturers, which is used to verify signature for the firmware released by manufacturer. ▪ PUK_TYPE_US_PUK Public key of user signature certificate, which is used to verify signature for the public key certificate. ▪ PUK_TYPE_USER The public key of users, which is used to verify signature for user's application.
Return	RET_OK	Succeeded.
	ERR_VERIFY_SIGN_FAIL	Invalid signature.
	ERR_FILE_NOT_EXIST	The file does not exist.

	<p>ERR_DEV_BUSY Device is busy.</p> <p>ERR_INVALID_PARA M Invalid parameter.</p>
Instruction	<ol style="list-style-type: none"> 1. This function is only used to verify the application parameter file, for example, to verify the root certificate defined by the application. 2. Before installing the file, it is not recommended to call this function to verify the legitimacy of the file so that to avoid a repeated verification, as OsInstallFile() will verify the file automatically.

4.17 Get System Version

4.17.1 OsGetSysVer

Prototype	void OsGetSysVer(int VerType, char *Version);	
Function	Get the version information of operating system and firmware version information.	
Parameters	VerType	Version types: TYPE_OS_VER: Operating system version; TYPE_WIRELESS_VER: Wireless firmware version;
	Version[Output]	Buffer of version information The valid length must be not less than 31 bytes.
Return	None	
Instruction	<ol style="list-style-type: none"> 1. If <i>Version[0]</i> is equal to 0x00, the corresponding module does not exist. 2. The buffer size of <i>Version</i> must be more than 31 bytes, and the version information must end with “\0”. 	

4.18 Test Whether on the Base

Prolin can test whether the handset is on the base or not, which is not applicable to models that do not have any base.

4.18.1 OsOnBase

Prototype	int OsOnBase(void);
-----------	----------------------------

Function	Get the status of POS terminal.	
Parameters	None.	
Return	1	On the base
	0	Not on the base
Instruction	This function only applies to POS terminals with a base.	

4.19 Save the Crash Report

Prolin can monitor program state. Once the program crashes, it will save the crash report in the directory “/data/tombstones” after calling OsSaveCrashReport().. .

4.19.1 OsSaveCrashReport

Prototype	void OsSaveCrashReport(<i>int sig</i>);	
Function	Save crash report.	
Parameters	Sig	Signal value.
Return	None.	
Instruction	<p>1. Method one Call signal(SIG_XXX, OsSaveCrashReport) to register OsSaveCrashReport() as signal handler, for example:</p> <pre>signal(SIGILL, OsSaveCrashReport); signal(SIGABRT, OsSaveCrashReport); signal(SIGBUS, OsSaveCrashReport); signal(SIGFPE, OsSaveCrashReport); signal(SIGSEGV, OsSaveCrashReport); signal(SIGSTKFLT, OsSaveCrashReport);</pre> <p>2. Method two Call OsSaveCrashReport (<i>sig</i>) to save the error messages. For example:</p> <pre>void mysighandler(<i>int sig</i>) { do_something(); OsSaveCrashReport(<i>sig</i>); }</pre> <ul style="list-style-type: none"> ● The recommended signals are SIGILL, SIGABRT, SIGBUS, SIGFPE, SIGSEGV and SIGSTKFLT. ● The corresponding signal of <i>sig</i> will be ignored. That is, signal(<i>sig</i>, SIG_IGN) is called inside OsSaveCrashReport(). 	

- The crash report can be exported to USB-disk by Terminal Manager(TM), for complete guide, please refer to “Prolin2.X User guide” chapter xx
- When compiling, GCC parameter `-g -fno-wind-tables` needs to be added to save the debugging information.

4.20 Mount and Unmount the External File System

4.20.1 OsMount

Prototype	<pre><code>int OsMount(const char *Source, const char *Target, const char *FileSystemType, unsigned long MountFlags, const void *Data);</code></pre>	
Function	Mount/install an external storage system, such as USB flash drive.	
Parameters	Source[Input]	The original/source file system that needs to be mounted. It is usually a device and must be in <code>/dev/block/</code> ; the valid length cannot exceed 128 bytes.
	Target[Input]	The destination file directory. It must be in <code>/mnt/</code> ; the valid path length cannot exceed 128 bytes.
	FileSystemType[Input]	Type of the file system to be mounted. The value of the file system type can be “vfat”.
	MountFlags[Input]	Mount flag. Here is a list of flags and their corresponding meanings. The mount flag can be a flag or a combination of the following flags. MS_DIRSYNC: Update synchronous directory. MS_MANDLOCK: Permit mandatory locking on files in this file system. <ul style="list-style-type: none"> ▪ MS_NOATIME: Do not update the access time of file. ▪ MS_NODEV: Do not allow to access device file.

	<ul style="list-style-type: none"> ▪ MS_NODIRATIME: Don't update the access time of directory. ▪ MS_NOEXEC: Do not allow programs to be executed on the mounted file system. ▪ MS_NOSUID: Donot follow the set-user-ID and set-group-ID when executing programs. ▪ MS_RDONLY: Specify the file system as read-only. ▪ MS_RELATIME: Update the last access time (atime) values if the last access time (atime) is not larger than the last modification time (mtime) or the last status change time (ctime). ▪ MS_SILENT: Stop writing warning messages to the system kernel log. ▪ MS_STRICTATIME: Always update the last access time (atime) for each access. MS_SYNCHRONOUS: Synchronize the updates
	Data[Input] User-defined data, it can be NULL.
Return	<p>RET_OK Mount succeeded.</p> <p>ERR_INVALID_PARAM Invalid parameter.</p> <p>ERR_STR_LEN The string is too long.</p> <p>ERR_NEED_ADMIN Permission denied.</p>
Instruction	<ol style="list-style-type: none"> 1. Only the MAINAPP has permission to mount; otherwise, ERR_NEED_ADMIN will be returned. 2. "/mnt" shall be included in the <i>Target</i> when setting the target file directory, such as "/mnt/sdcard". 3. The head file <code>#include <sys/mount.h></code> shall be included in the code, because this function involves macros that are defined in "mount.h", such as MS_DIRSYNC, MS_MANDLOCK, etc. <p>Sample code</p> <p>Mount a SD card:</p> <pre>ret = OsMount("/dev/block/mmcblk0p1", "/mnt/sdcard", "vfat", 0, 0);</pre>

	<p>Folder “/mnt/sdcard” shall be created in advance with either user-defined name or other names. If the mount succeeded, it can conduct read-write operation to the SD card.</p> <p>Mount a U disk:</p> <pre>ret = OsMount("/dev/block/sda1", "/mnt/udisk", "vfat", 0, 0);</pre> <p>Folder “/mnt/udisk” shall be created in advance with either user-defined name or other names. If the mount succeeded, it can read/write USB flash drive.</p> <p>Notes:</p> <ol style="list-style-type: none"> SD card or USB flash drive hot-plugging may change device path. Above instructions only have one partition based on SD card and USB flash drive. Only FAT32 USB flash drive format is supported.
--	---

4.20.2 OsUnmount

Prototype	int OsUnmount(const char *Target, int Flags);	
Function	Unmount file system.	
Parameters	Target[Input]	File system that needs to be unmounted. The path of the target file system must be in the <code>/mnt/</code> directory; the valid path length cannot exceed 128 bytes.
	Flags[Input]	Unmount flag. Here is a list of flags and their corresponding meanings. The unmount flag can be a flag or a combination of the following flags: <ul style="list-style-type: none"> MNT_DETACH: Lazy unmount. The mount point is inaccessible after executing, and it will unmount only when the mount point is not busy. MNT_EXPIRE:Mark the mount point as expired UMOUNT_NOFOLLOW: If the target is a symbolic link, it will not reduce the reference count.
Return	RET_OK ERR_INVALID_PAR AM ERR_STR_LEN ERR_NEED_ADMIN	Succeeded. Invalid parameter. Permission denied. The string is too long.

Instruction	<p>1. Only the main application has permission to unmount; otherwise, it will fail to unmount and return ERR_NEED_ADMIN.</p> <p>2. The head file #include <sys/mount.h> shall be included in the code, because this function involves macros that are defined in "mount.h", such as MS_DIRSYNC, MS_MANDLOCK, etc.</p> <p>Note: Only FAT32 USB flash drive format is supported.</p> <p>Sample code</p> <p>Unmount a SD card: <code>ret = OsUnmount("/mnt/sdcard", 0);</code></p> <p>Unmount a USB flash drive: <code>ret = OsUnmount ("/mnt/udisk", 0);</code></p>
--------------------	---

4.21 Encrypted File System Interface

4.21.1 OsCryptFormat

Prototype	<code>int OsCryptFormat(const char *Dev, const char *Pwd, const char *FsType);</code>						
Function	Format the storage device to be an encrypted file system.						
Parameters	<p>Dev[Input] Device that needs to be formatted. This device is located at /dev/block/ directory, and the path length can not exceed 128 bytes.</p>						
	<p>Pwd[Input] Password of encrypted file system, and it is a visible string ranging from 6 to 32 bytes.</p>						
	<p>FsType[Input] The type of format file system, and this value can be "vfat".</p>						
Return	<table border="0"> <tr> <td style="vertical-align: top;">></td> <td>Succeeded</td> </tr> <tr> <td style="vertical-align: top;">ERR_INVALID PARA M</td> <td>Invalid parameter.</td> </tr> <tr> <td style="vertical-align: top;">ERR_DEV_NOT_EXIS T</td> <td>The file or path doesn't exist</td> </tr> </table>	>	Succeeded	ERR_INVALID PARA M	Invalid parameter.	ERR_DEV_NOT_EXIS T	The file or path doesn't exist
>	Succeeded						
ERR_INVALID PARA M	Invalid parameter.						
ERR_DEV_NOT_EXIS T	The file or path doesn't exist						
Instruction	<ol style="list-style-type: none"> When formatting a non-encrypted storage device to an encrypted file system for the first time, this function needs to be called to format the device. If a storage device has been formatted to an encrypted file system, then it doesn't need to be formatted again when 						

user uses it on another terminal, otherwise, the data will get lost.

CAUTION

When formatting the device to an encrypted file system, it will delete all the data files from the device, please make sure that all the data files have been backed up.

4.21.2 OsCryptMount

Prototype	<code>int OsCryptMount(const char *Dev, const char *Pwd, const char *Target, const char *FsType);</code>	
Function	Mount the encrypted file system.	
Parameters	Dev [Input]	Device that needs to be mounted, this device is located at <code>/dev/block/</code> directory, and the path length cannot exceed 128 bytes.
	Pwd [Input]	Password of encrypted file system, it is a visible string ranging from 6 to 32 bytes.
	Target [Input]	The target file directory, it must be located at <code>/mnt/</code> directory, and the path length cannot exceed 128 bytes.
	FsType [Input]	The type of format file system, this value can be “vfat”.
Return	0	Succeeded.
	ERR_INVALID_PARAMETER	Invalid parameter
	ERR_SYS_NOT_SUPPORTED	System doesn't support (wrong password or encryption format is unsupported)
	ERR_DEV_NOT_EXIST	Device doesn't exist
	ERR_ACCESS_DENY	No permission to access
Instruction	<ol style="list-style-type: none"> When calling this function to mount the non-encrypted storage device for the first time, <code>OsCryptFormat()</code> shall be called to format the device first. When the encrypted SD card password is wrong, the system will not be able to decrypt data or get the encrypted information in correct format, therefore whether it is wrong password or unsupported encryption format, 	

	<p>ERR_SYS_NOT_SUPPORT will be returned.</p> <p>3. User can not mixedly use non-encryption interfaces OsMount()/OsUmount() with encryption interfaces OsCryptMount()/OsCryptUnmount(). For example, after calling OsMount() to mount a device, calling OsCryptUnmount() will not be allowed to unmount this device.</p>
--	---

4.21.3 OsCryptUnmount

Prototype	int OsCryptUnmount(const char *Target);	
Function	Unmount the encrypted file system.	
Parameters	Target [Input]	The directory of encrypted SD card that needs to be unmounted.
Return	0	Succeeded.
	ERR_INVALID_PARA M	Invalid parameter
	ERR_DEV_BUSY	Device is busy.
	ERR_ACCESS_DENY	Permission denied.
	ERR_FILE_NOT_EXIS T	The file directory doesn't exist
Instruction	File related operations such as read/write file system must be stopped when calling OsCryptUnmount(), otherwise, ERR_DEV_BUSY will be returned.	

4.22 LED

4.22.1 OsLed

Prototype	int OsLed(int Id, unsigned int Color, const char *dev);	
Function	Control Led state and color.	
Parameters	Id [Input]	Led number.
	Color [Input]	The color of Led can be set as followings: LED_OFF: off LED_RED: red LED_GREEN: green LED_BLUE: blue

		LED_YELLOW: yellow LED_CYAN: cyan LED_MAGENTA: magenta LED_WHITE: white
	dev [Input]	Device model. When operating the led light of remote device, this parameter needs to be input (such as card reader: "IM700", "IM500" or "IM20"); When operating the led light of the local device, this parameter is NULL.
Return	0	Succeeded.
	ERR_INVALID_PARAM	Invalid parameter
	ERR_SYS_BAD	System error.
	ERR_SYS_NOT_SUPPORT	System does not support.
Instruction	IM500 IM300 IM310	Id = 1: red Id = 2: green
	IM700 QR55 SP200	Id = 1: blue Id = 2: yellow Id = 3: green Id = 4: red
	B920	Id = 1: red or green
	IM20 Q20	Id = 1: red, green, blue, yellow, cyan, magenta or white
	Q28	Id = 1: green Id = 2: red or green
	Q30	Id = 1: red Id = 2: green Id = 3: blue Id = 4: yellow Id = 5: red or green Id = 6: red or green

4.23 Get Terminal Accumulated Usage Information

4.23.1 OsTerminalConsumelInfo

Prototype	int OsTerminalConsumelInfo(const char *Key, int *Value);	
Function	Get accumulated usage information of each module on terminal.	
Parameters	Key [Input]	Device information category.
	Value [Output]	Device usage information
Return	RET_OK	Succeeded.
	ERR_SYS_NOT_SUP PORT	System does not support.
	ERR_INVALID PARA M	Invalid parameter.
Instruction	<ol style="list-style-type: none"> 1. The unit of time is “minute”, the unit of print length is “centimeter”; 2. Only when the return value is RET_OK can the required device information be obtained from the Value. 3. The statistics are rough and not entirely accurate. After erasing the data partition, the statistics will be cleared. 4. Real-time statistics of button times, touch screen clicks and RF card taps; the number of magnetic card swiping, IC card inserting, keystroke backlight time, LCD using time, battery using time and total running time are detected and counted every minute. The statistics will be saved to the file system every half hour. If the POS terminal is powered down directly, it is likely to lose the statistics within half an hour, so it is recommended to long press the shutdown button or call the OsReboot()/OsPowerOff() function to restart/shut down the terminal. 	

The current supported device information is listed below:

Key value	description
msr.swiped.count	counts of swiping magnetic stripe card
msr.swiped.failure_count	Counts of failed swiping magnetic stripe card
icc.inserted.count	counts of inserting card
picc.tapped.count	counts of tapping RF card

key.backlight.time	Cumulative time of backlight on the button
Lcd.display.time	the display time of LED
touchscreen.click.count	counts of clicking on touch screen (Multi-touch only counts one point)
key.pressed.count	keystrokes
boot.count	boot times
running.time	Total run time, excluding sleep time
battery.used.time	battery life, sleep time is not included

4.24 Digital IO

4.24.1 OsDigitalIOGetStat

Prototype	int OsDigitalIOGetStat(int *value);	
Function	Get the input status of all digital IO.	
Parameters	Value[Output]	The status of each IO is represented by bit, 0 represents suspended or high level, 1 represents low level. If the IO is unreadable (write only), the corresponding bit returns 0.
Return	>=0 ERR_INVALID_PARAM ERR_SYS_NOT_SUPPORT	The number of bits of 1 in value. Invalid parameter. System does not support.
Instruction	In the case of a connected handset, if the handset supports digital IO, the interface will get the input status of the handset digital IO by default.	

4.24.2 OsDigitalIOSetStat

Prototype	int OsDigitalIOSetStat(int index, int value);	
Function	Set the output status of the specified IO.	
Parameters	Index[Input]	Specify the sequence number of IO to be set, and if the IO does not support setting (read only), ERR_SYS_NOT_SUPPORT will be returned.

	Value[Input]	0: the corresponding IO output is high level; 1: the corresponding IO output is low level.
Return	RET_OK ERR_INVALID_PARA M ERR_SYS_NOT_SUP PORT	Succeeded. Invalid parameter. System does not support.
Instruction	In the case of a connected handset, if the handset supports digital IO, the interface will get the input status of the handset digital IO by default.	

5 Encryption and Decryption

5.1 Return Code List

Table 5.1 Encryption and decryption return code list

Macro	Value	Description
<i>ERR_DATA_TOO_BIG</i>	-2400	RSA encrypted data is greater than module.
<i>ERR_GEN_RANDOM</i>	-2401	Fail to generate random numbers.
<i>ERR_GEN_FAIL</i>	-2402	Fail to generate RSA key pairs.

5.2 Random Number

Prolin provides true random number interfaces for applications.

5.2.1 OsGetRandom

Prototype	void OsGetRandom(unsigned char *Random,
-----------	--

	int RandomLen);	
Function	Generate true random number.	
Parameters	Random [Output]	A pointer to string that stores random number.
	RandomLen[Input]	Length of random number which needs to be read. The valid length is not more than 4096 bytes.
Return	None.	
Instruction		

5.3 SHA Algorithm

Prolin supports SHA algorithms, including SHA-1, SHA-2(SHA-256, SHA-512) and the truncation form of SHA-2 (SHA-224, SHA-384).

5.3.1 OsSHA

Prototype	void OsSHA(int Mode, const void *Data, int DataLen, unsigned char* ShaOut);	
Function	Calculate the hash value of SHA (Secure Hash Algorithm).	
Parameters	Mode	Algorithm types: <ul style="list-style-type: none">▪ SHA_TYPE_1▪ SHA_TYPE_224▪ SHA_TYPE_256▪ SHA_TYPE_384▪ SHA_TYPE_512
	Data[Input]	Input data buffer.
	DataLen[Input]	Input data length.
	ShaOut[Output]	Output SHA value. The memory space is recommended to be more than 64 bytes. The corresponding relationship between <i>Mode</i> value and the valid length of <i>ShaOut</i> are listed below: <ul style="list-style-type: none">▪ SHA_TYPE_1 20

		<ul style="list-style-type: none"> ▪ SHA_TYPE_224 28 ▪ SHA_TYPE_256 32 ▪ SHA_TYPE_38 48 ▪ SHA_TYPE_512 64
Return	None.	
Instruction		

5.4 DES Algorithm

Prolin supports [DES&TDES](#) algorithms.

5.4.1 OsDES

Prototype	<code>void OsDES(const unsigned char *Input, unsigned char *Output, const unsigned char *DesKey, int KeyLen, int Mode);</code>	
Function	Encrypt or decrypt data with DES / TDES algorithm.	
Parameters	Input[Input]	8-byte input data.
	Output[Output]	8-byte output data.
	DesKey[Input]	DES/TDES key.
	KeyLen[Input]	8, 16 or 24 (bytes).
	Mode	0- Decryption; 1- Encryption.
Return	None.	
Instruction	<ol style="list-style-type: none"> 1. Encryption or decryption is decided by <i>Mode</i> selection. 2. For invalid parameter, it performs no action. 	

5.5 AES Algorithm

Prolin supports [AES](#) algorithm, including AES-128, AES-192 and AES-256.

5.5.1 OsAES

Prototype	<code>void OsAES(const unsigned char *Input, unsigned char *Output, const unsigned char *AesKey, int KeyLen, int Mode);</code>	
Function	Encrypt or decrypt data with AES algorithm	
Parameters	Input[Input]	16-byte input data.
	Output[Output]	16-byte output data.
	AesKey[Input]	Key.
	KeyLen	16, 24 or 32 (bytes).
	Mode	0- Decryption; 1- Encryption.
Return	None.	
Instruction	<ol style="list-style-type: none"> 1. This function supports AES encryption and decryption of 128, 192 or 256 bits. 2. For invalid parameter, it performs no action. 	

5.6 RSA Algorithm

Prolin supports [RSA](#) algorithm, including public/private key-pair generation, RSA encryption and RSA decryption. Prolin supports a maximum modulus length of 4096 bits.

5.6.1 OsRSA

Prototype	<code>int OsRSA(const unsigned char *Modulus, int ModulusLen, const unsigned char *Exp, int ExpLen, const unsigned char *DataIn, unsigned char *DataOut);</code>	
Function	Encrypt or decrypt data with RSA algorithm	
Parameters	Modulus[Input]	A pointer that contains RSA modulus ($n=p \cdot q$). (stored in big)

		endian)
	ModulusLen[Input]	Modulus length[unit:byte].
	Exp[Input]	A pointer that contains RSA operation exponent. (stored in big endian)
	ExpLen	Exponent length.[unit: byte]
	DataIn[Input]	A pointer to the input data buffer. The valid length is equal to the modulus length.
	DataOut[Output]	A pointer to the output data buffer. The valid length is equal to the modulus length.
Return	RET_OK	Succeeded.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_DATA_TOO_BIG	ExpLen is greater than ModulusLen.
Instruction	<ol style="list-style-type: none"> OsRSA() is used to encrypt/decrypt <i>DataIn</i> and its result is stored in <i>DataOut</i>. When (Modulus, Exp) is private key, if <i>DataIn</i> is ciphertext that encrypted by PUK, then <i>DataOut</i> is plaintext of <i>DataIn</i>; otherwise, <i>DataOut</i> is RSA ciphertext of <i>DataIn</i>. When (Modulus, Exp) is public Key, if <i>DataIn</i> is ciphertext that encrypted by private key, then <i>DataOut</i> is plaintext of <i>DataIn</i>; otherwise, <i>DataOut</i> is RSA ciphertext of <i>DataIn</i>. OsRSA() can implement RSA operation whose modulus length is not more than 4096 bits. 	

5.6.2 OsRSAKeyGen

Prototype	Int OsRSAKeyGen(unsigned char *Modulus, unsigned char *PriExp, int ModulusLen, const unsigned char * PubExp);	
Function	Generate RSA key pair of specified exponent and modulus length.	
Parameters	Modulus[Output]	Key modulus. (stored in big endian)
	PriExp[Output]	Private key exponent. (stored in big endian)
	ModulusLen	Modulus length [unit: byte]. The modulus is valid only when it is 64, 128, 256 or 512 bytes.

	PubExp[Input]	Public key exponent. The valid value can only be “\x00\x00\x00\x03” or “\x00\x01\x00\x01”.
Return	RET_OK ERR_INVALID_PARA M ERR_GEN_RANDOM ERR_GEN_FAIL	Succeeded. Invalid Parameters. Fail to generate random data. Fail to generate RSA Key pair.
Instruction		

6PED

Prolin provides a series of PED interfaces, including built-in PED, MK/SK, DUKPT, RSA and other related interfaces.

The architecture of PED is designed to contain three levels of key. From top to bottom, its order is:

- *TLK—Terminal Key Loading Key*

Private key of acquirer or POS operator, which is directly written in by acquirer and POS operator in secure environment.

Each PED terminal only has one TLK. Its index number is 1.

- *TMK—Terminal Master Key=Acquirer Master Key*

Terminal master key or acquirer master key, each PED terminal can have 100 TMKs. The index number is from 1 to 100.

- *TWK—Transaction working key = Transaction Pin Key + Transaction MAC Key + Terminal DES Key*

Transaction working key, which is used to perform PIN, MAC and other operations.

Each PED terminal can have 100 TMKs. The index number is from 1 to 100.

TPK: Calculate PIN Block after inputting PIN.

TAK: Calculate MAC in message communication.

TDK: Transmit or store sensitive data encrypted with DES/TDES algorithm.

Each key has its corresponding index number, length, purpose and tags which are set through API before writing in the key so as to authorize the permission and avoid abuse of the key.

- Mechanism of DUKPT:

DUKPT (Derived Unique Key Per Transaction) is the one-time pad system in which different key (PIN、MAC) is used for each transaction. KSN (Key Serial Number) concept which is the key factor to realize one-time pad is introduced in this system.

Each key corresponding to KSN can generate different keys according to the usage and variant constants listed below.

Key used for	Variant constant
PIN	00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00 FF
MAC, request or both ways	00 00 00 00 00 00 FF 00 00 00 00 00 00 00 00 FF 00
MAC, response	00 00 00 00 FF 00 00 00 00 00 00 00 00 00 FF 00 00 00
Data encryption	00 00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 FF 00 00

Each PED terminal can have 100 DUKPT groups. Selection of group index is needed when writing in TIK; select the corresponding index number when using DUKPT.

6.1 Return Code List

Table 6.1 PED return code list

Macro	Value	Description
<i>ERR_PED_NO_KEY</i>	-3801	<i>Key does not exist.</i>
<i>ERR_PED_KEY_IDX_ERR</i>	-3802	<i>Key index error.</i>
<i>ERR_PED_DERIVE_ERR</i>	-3803	<i>Key level error: Source key level is lower than target key level.</i>

ERR_PED_CHECK_KEY_FAIL	-3804	Key verification failed.
ERR_PED_NO_PIN_INPUT	-3805	No PIN input.
ERR_PED_PIN_INPUT_CANCEL	-3806	PIN input is canceled.
ERR_PED_WAIT_INTERVAL	-3807	The function is called within the minimum time interval (Calculate PIN block/MAC).
ERR_PED_KCV_MODE_ERR	-3808	KCV mode error.
ERR_PED_KEY_TAG_ERR	-3809	Key tag error.
ERR_PED_KEY_TYPE_ERR	-3810	Key type error.
ERR_PED_PIN_LEN_ERR	-3811	PIN length error.
ERR_PED_DSTKEY_IDX_ERR	-3812	Target key index error.
ERR_PED_SRCKEY_IDX_ERR	-3813	Source key index error.
ERR_PED_KEY_LEN_ERR	-3814	Key length error.
ERR_PED_INPUT_PIN_TIMEOUT	-3815	PIN input timeout.
ERR_PED_NO_ICC	-3816	IC card does not exist.
ERR_PED_ICC_INIT_ERR	-3817	IC card initialization error.
ERR_PED_GROUP_IDX_ERR	-3818	DUKPT group index error.
ERR_PED_LOCKED	-3819	PED is locked.
ERR_PED_NOMORE_BUF	-3820	No free buffer.
ERR_PED_NORMAL_ERR	-3821	PED normal error.
ERR_PED_NEED_ADMIN	-3822	Permission denied.
ERR_PED_DUKPT_KSN_OVERFLOW	-3823	DUKPT overflow.
ERR_PED_KCV_CHECK_FAIL	-3824	KCV check failed.
ERR_PED_SRCKEY_TYPE_ERR	-3825	Source key type error.
ERR_PED_UNSPT_CMD	-3826	Unsupported command.
ERR_PED_ADMIN_ERR	-3827	Administration error.
ERR_PED_DOWNLOAD_INACTIVE	-3828	PED download function is not activated.
ERR_PED_KCV_ODD_CHECK_FAIL	-3829	KCV odd parity check failed.
ERR_PED_PED_DATA_RW_FAIL	-3830	Fail to read PED data.
ERR_PED_ICC_CMD_ERR	-3831	IC card operation failed.

ERR_PED_DUKPT_NEED_INC_KSN	-3832	DUKPT KSN needs to increase 1 first.
ERR_PED_DUKPT_NO_KCV	-3833	NO KCV.
ERR_PED_NO_FREE_FLASH	-3834	PED doesn't have enough space.
ERR_PED_INPUT_CLEAR	-3835	Press [CLEAR] key to exit PIN input.
ERR_PED_INPUT_BYPASS_BYFUNCTION	-3836	Press FN/ATM4 to cancel PIN input.
ERR_PED_NO_PIN_MODE	-3837	PIN input mode is not set.
ERR_PED_DATA_MAC_ERR	-3838	Data MAC check error.
ERR_PED_DATA_CRC_ERR	-3839	Data CRC check error.
ERR_PED_KEY_VALUE_INVALID	-3840	The work key value already exists or does not meet requirements.

6.2 Data Definition

6.2.1 Key Type

Table 6.2 Key types

Macro	Value	Description
PED_TLK	0x01	Loading Key
PED_TMK	0x02	Master Key
PED_TPK	0x03	PIN Key
PED_TAK	0x04	MAC Key
PED_TDK	0x05	Data Key
PED_TIK	0x10	DUKPT Initial Key
PED_TRK	0x07	MSR Key
PED_TAESK	0x20	AES Key
PED_SM2_PVT_KEY	0x30	SM2 Private Key
PED_SM2_PUB_KEY	0x31	SM2 Public Key
PED_SM4_TMK	0x32	SM4 Master Key
PED_SM4_TPK	0x33	SM4 PIN Key
PED_SM4_TAK	0x34	SM4 MAC Key

PED_SM4_TDK

0x35

SM4 Data Key

6.2.2 DisplayAttribute of Asterisk

Table 6.3 Layout attributes of asterisk

Macro	Value	Description
PED_ASTERISK_ALIGN_LEFT	0	<i>Left-aligned.</i>
PED_ASTERISK_ALIGN_CENTER	1	<i>Center-aligned.</i>
PED_ASTERISK_ALIGN_RIGHT	2	<i>Right-aligned.</i>

Table 6.4 Color values of asterisk

Macro	Value	Description
RGB(_r, _g, _b)	...	<i>To generate 16-bit color value with the three primary colors.</i>

6.3 Data Structure

6.3.1 Structure of RSA Key

ST_RSA_KEY

```
typedef struct{
    int ModulusLen; /*Modulus length(bits) */
    unsigned char Modulus[512]; /*Modulus. When modulus length is less than 512 bytes, add 0x00 on the left. */
    int ExponentLen; /* Exponent length (bits) */
    unsigned char Exponent [512]; /*Exponent. When exponent is less than 512 bytes, add 0x00 on the left.*/
    unsigned char KeyInfo[128]; /* RSA key information */
}ST_RSA_KEY;
```

6.3.2 RSA Key Structure for Verifying the Ciphertext IC Card PIN

ST_RSA_PINKEY

```

typedef struct{
    int ModulusLen; /*Modulus length of PIN public key(unit: bits) */
    unsigned char Modulus[256]; /*Modulus of PIN public key*/
    unsigned char Exponent [4]; /* Exponent of PIN public key*/
    int IccRandomLen; /*Length of random data from IC card*/
    unsigned char IccRandom[8]; /*Random data from IC card*/
}ST_RSA_PINKEY;

```

6.4 Basic PED

6.4.1 OsPedOpen

Prototype	int OsPedOpen(void);	
Function	Open PED service.	
Parameters	None.	
Return	RET_OK	Succeeded. ERR_DEV_BUSY Device is busy.
Instruction	OsPedOpen() must be called to open PED device; otherwise, other PED related functions won't work.	

6.4.2 OsPedGetVer

Prototype	int OsPedGetVer (unsigned char * PedVer);	
Function	Get current PED version information.	
Parameters	PedVer[Output]	PED version information buffer, the buffer size is 6 bytes.
Return	RET_OK	Succeeded. ERR_DEV_NOT_OPEN PED device is not open. ERR_INVALID_PARAM Invalid parameter.
Instruction		

6.4.3 OsPedSetInterval

Prototype	int OsPedSetInterval(unsigned long TpkIntervalMs);	
Function	Set the minimum time interval of two PIN block calculations.	
Parameters	TpkIntervalMs	= 0 Use the default value(30s) <1000 Automatically set to 1,000(1s)

		>600000 =0xffffffff	Automatically set to 600,000(10 min) Current settings remain unchanged.
Return	RET_OK ERR_DEV_NOT_OPE N	Succeeded. PED device is not open.	
Instruction	<p>Restrictions of PIN input intervals:</p> <p>Each PIN input function can only be called for not more than 4 times during $4 * TpkIntervalMs$. The default value of $TpkIntervalMs$ is 30s which can be set by calling OsPedSetInterval(). For example, if the $TpkIntervalMs$ is 20,000(ms), then the same PIN input function can be called 4 times at most within 80 seconds.</p> <p>PIN input functions include OsPedGetPinBlock(), OsPedGetPinDukpt(), OsPedVerifyPlainPin() and OsPedVerifyCipherPin().</p> <p>$TpkIntervalMs$ will be restored to the default value after reboot.</p>		

6.4.4 OsPedVerifyPlainPin

Prototype	<code>int OsPedVerifyPlainPin(int IccSlot, const char *ExpPinLen, int Mode, unsigned long TimeoutMs, unsigned char *IccRsp);</code>	
Function	Verify offline plaintext PIN.	
Parameters	IccSlot	IC card slot number, IccSlot=0.
	ExpPinLen[Input]	A pointer to valid password length string which is an enumeration set from 0 to 12. Application enumerates all valid password length and separate each length with ",". For example, if no PIN, 4 and 6 digits of PIN are allowed, the string will be set to "0, 4, 6". "0" means no PIN is required and can return directly by pressing "Enter".
	Mode	0x00 : IC card command mode. It supports the IC card command that conforms to EMV2000.
	TimeoutMs	Timeout of PIN input.[unit:ms] The valid timeout ranges from 0 to 300000

		ms. "0" means no timeout, and PED has no timeout control.
	IccRsp[Output]	Card response status code (2 bytes: SWA++SWB) The valid length is 2 bytes.
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPE N	PED device is not open.
	ERR_INVALID PARA M	Invalid parameter.
	Others	Refer to PED Return Code List .
Instruction	Call OsPedVerifyPlainPin() will operate the frame buffer, which might cause the resource conflict. If there is some kind of exception on the interface of application which is running XUI, call XuiSuspend() to suspend XUI before calling OsPedVerifyPlainPin(), after a while, call XuiResume() to resume XUI.	
Internal handling porcess	<p>Internal processing steps to verify PIN block:</p> <ol style="list-style-type: none"> 1. Prompt cardholder to input PIN; 2. Prompt that it is under processing; 3. Convert plaintext PIN to PIN block form. <p>Oslccexchange() is used to verify interaction with card. The process is:</p> <pre> ST_APDU_REQ apdu_s; ST_APDU_RSP apdu_r; memcpy (apdu_s.cmd, icc_command, 4); apdu_s.lc = icc_command[4]; memcpy (apdu_s.data_in, PINBLOCK, apdu_s.lc); apdu_s.le = 0; if (icc_exchange(icc_slot, 0, &apdu_s, &apdu_r)) return CMDERR; icc_resp[0] = apdu_r.swa; icc_resp[1] = apdu_r.swb; </pre>	

6.4.5 OsPedVerifyCipherPin

Prototype	<pre> int OsPedVerifyCipherPin(int IccSlot, const ST_RSA_PINKEY *RsaPinKey,</pre>
-----------	---

	<pre>const char *ExpPinLen, int Mode, unsigned long TimeoutMs, unsigned char *IccRsp);</pre>	
Function	<p>Verify ciphertext PIN, steps are as follows:</p> <ol style="list-style-type: none"> 1. Get the plaintext PIN; 2. Encrypt the plaintext with RsaPinKey provided by application in accordance with EMV specification; 3. Send ciphertext PIN to card according to card command and card slot number provided by application. 	
Parameters	IccSlot	ICC slot number, IccSlot=0.
	RsaPinKey[Input]	A pointer to ST_RSA_PINKEY that needs to be encrypted.
	ExpPinLen[Input]	A pointer to valid password length string, which is an enumeration set from 0 to 12. Application enumerates all valid password length and separate each length with “,”. For example, if no PIN, 4 and 6 digits of PIN are allowed, the string will be set to “0, 4, 6”. “0” means no PIN is required and can return directly by pressing “Enter”.
	Mode	0x00, IC card command mode. It supports the IC card command that conforms to EMV2000.
	TimeoutMs	Timeout of inputting PIN. [unit:ms] The valid timeout value ranges from 0 to 300000 ms. “0” means no timeout, and PED has no timeout control.
	IccRsp[Output]	Status code of card's response. (2 bytes: SWA+SWB)
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAMETER	Invalid parameter.
	Others	Refer to the PED Return Code List .
Instruction	Call OsPedVerifyCipherPin() will operate the framebuffer, which might cause the resource conflict. If there is some kind of exception on the interface of application which is running XUI, call XuiSuspend() to suspend XUI before calling	

	<p>OsPedVerifyCipherPin(), after a while, call XuiResume() to resume XUI.</p> <p>Encryption algorithm:</p> <p>Get ciphertext PIN by using the public key to calculate the data which are listed in following table with RSA algorithm.</p>																				
	<table border="1"> <thead> <tr> <th>Field name</th> <th>Length</th> <th>Description</th> <th>Format</th> </tr> </thead> <tbody> <tr> <td>Header of data</td> <td>1</td> <td>Hexadecimal, the value is “7F”</td> <td>b</td> </tr> <tr> <td>PIN Block</td> <td>8</td> <td>PIN block</td> <td>b</td> </tr> <tr> <td>Random number of IC card</td> <td>8</td> <td>Random number gotten from card, stored in ST RSA PINKEY</td> <td>b</td> </tr> <tr> <td>Random padding bytes</td> <td>NIC-17</td> <td>Random padding bytes gotten from terminal application, stored in ST RSA PINKEY.</td> <td>b</td> </tr> </tbody> </table>	Field name	Length	Description	Format	Header of data	1	Hexadecimal, the value is “7F”	b	PIN Block	8	PIN block	b	Random number of IC card	8	Random number gotten from card, stored in ST RSA PINKEY	b	Random padding bytes	NIC-17	Random padding bytes gotten from terminal application, stored in ST RSA PINKEY .	b
Field name	Length	Description	Format																		
Header of data	1	Hexadecimal, the value is “7F”	b																		
PIN Block	8	PIN block	b																		
Random number of IC card	8	Random number gotten from card, stored in ST RSA PINKEY	b																		
Random padding bytes	NIC-17	Random padding bytes gotten from terminal application, stored in ST RSA PINKEY .	b																		
Internal handling porcess	<p>The internal processing steps are:</p> <ol style="list-style-type: none"> 1. Prompt cardholder to input PIN; 2. Prompt cardholder that it is processing; 3. Convert plaintext PIN to PIN block; 4. Generate data, used for encryption(the data are listed in above table); 5. Use the public key to encrypt data which is generated in step 4 with RSA algorithm, and get the enciphered PIN finally. <p><code>OslccExchange()</code> is used to send verification command to card. The process is as follows:</p> <pre> <i>ST_APDU_REQ apdu_s;</i> <i>ST_APDU_RSP apdu_r;</i> <i>memcpy (apdu_s.cmd, icc_command, 4);</i> <i>apdu_s.LC = icc_command[4];</i> <i>memcpy (apdu_s.data_in, EncipheredPIN, apdu_s.LC);</i> <i>apdu_s.LE = 0;</i> <i>if (OslccExchange(icc_slot, 0, &apdu_s, &apdu_r))</i> <i>return ERR_PED_ICC_CMD_ERR;</i> <i>icc_resp[0] = apdu_r.SWA;</i> </pre>																				

ICC RESP	<code>icc_resp[1] = apdu_r.SWB;</code>
----------	--

6.4.6 OsPedEraseKeys

Prototype	int OsPedEraseKeys(void);	
Function	Clear all key information saved in PED.	
Parameters	None.	
Return	RET_OK ERR_DEV_NOT_OPEN Others	Succeeded. PED device is not open. Refer to PED Return Code List .
Instruction		

6.4.7 OsPedSetFunctionKey

Prototype	int OsPedSetFunctionKey(int KeyFlag);	
Function	Set the function of some function keys:	
Parameters	KeyFlag	<p>0x00: When the PIN has already been cleared or there is no PIN input, pressing CLEAR button will exit from the input status and ERR_PED_INPUT_CLEAR will be returned.</p> <p>0x01: When interfaces (OsPedGetPinBlock(), OsPedGetPinDukpt(), OsPedVerifyPlainPin(), OsPedVerifyCipherPin() etc) are inputting PIN, pressing CLEAR button will clear the latest input PIN digit by digit. Pressing CLEAR button after cleared all PIN will have no response and it will not exit the PIN input function.</p> <p>0x02: Press ATM4 button to end the PIN input. This rule does not apply to the POS terminals without ATM button.</p> <p>0x03: Press FN button to end the PIN input. This rule does not apply to the POS terminals without FN button.</p> <p>0x04: Press INFO button to end the PIN input. This rule does not apply to the POS terminals without INFO button.</p> <p>0x05: When the password input interface is inputting PIN, pressing CANCEL button will clear all input PIN. Pressing CANCEL button after cleared all PIN will have no response and it will not exit the PIN input function.</p> <p>0xff: Restore the default function of function keys</p>

		(press CLEAR button to clear all PIN, press CANCEL button to exit the PIN input function and pressing ATM4/FN/INFO key does not exit from the PIN input).
Return	RET_OK ERR_DEV_NOT_OPE N ERR_INVALID_PARAM Others	Succeeded. PED device is not open. Invalid parameter. Refer to the PED Return Code List .
Instruction	After PED is powered on, the default function of CLEAR button is to clear inputted PIN when the cardholder is inputting the PIN. And call this interface to set other functions of the CLEAR button.	

NOTE

When inputting PIN, if the function of pressing button to exit or to clear the PIN bit by bit is needed, call this function after startup.

6.4.8 OsPedClose

Prototype	void OsPedClose(void);
Function	Disconnect PED service.
Parameters	None.
Return	None.
Instruction	Call this function to close device before exiting program.

6.4.9 OsPedCancelPinEntry

Prototype	int OsPedCancelPinEntry(void);
Function	Terminate the process of PIN input.
Parameters	None.
Return	RET_OK Succeeded.
Instruction	This function does not require PED service, so there is no need to call OsPedOpen() before using it.

6.4.10 OsPedSetOfflinePin

Prototype	int OsPedSetOfflinePin(uchar Mode,
------------------	---

		<code>uchar TpkIdx, uchar *PinBlock, ushort PinBlockLen);</code>
Function	Set verification mode for off-line plain / cipher text.	
Parameters	Mode [Input]	Verification mode: <ul style="list-style-type: none"> ● 0-Input PIN with internal code keypad; ● 1- Input PIN with external code keypad, and the PIN is imported by parameter PinBlock.
	TpkIdx [Input]	TPK index: <ul style="list-style-type: none"> ● When Mode is 0, it is meaningless; ● When Mode is 1, the TPK of this index will be used to decrypt the imported PinBlock, and the result is PIN in plain-text.
	PinBlock [Input]	PIN block: <ul style="list-style-type: none"> ● When Mode is 0, it is meaningless; ● When Mode is 1, it represents PIN in cipher -text encrypted by TPK in ISO9564 Format1.
	PinBlockLen [Input]	Length of PIN block
Return	RET_OK ERR_DEV_NOT_OPEN ERR_INVALID_PARAM Others	Succeeded PED device is not open Invalid parameter Refer to the PED Return Code List
Instruction		

6.4.11 OsPedEndPinEntry

Prototype	<code>int OsPedEndPinEntry(void);</code>	
Function	Send the confirmation key to end PIN input.	
Parameters	None	
Return	RET_OK	Succeeded
Instruction	This function does not require PED service, so there is no need to call OsPedOpen() before using it.	

6.4.12 OsPedPinKeyNotify

Prototype	int OsPedPinKeyNotify(int *KeyCacheCnt, uchar *KeyCache);	
Function	Listen and get the number of PIN keys entered by users in the current state and the sequence of historical keys between this listening and the last listening.	
Parameters	KeyCacheCnt [Output]	The number of obtained historical key values.
	KeyCache [Output]	<ul style="list-style-type: none"> ● The key values are stored into the buffer in chronological order from little-endian to big-endian. And the size of this output buffer must be not less than 64 bytes. ● The maximum of obtained historical key sequence is 64, if the key buffer in the key sequence is more than 64, the latest 64 key sequences will be output. ● The obtained key can only be “PIN”, “ENTER”, “CLEAR”, “CANCEL”, “FN”, and “ATM4”, and “PIN” is replaced by “*”. If there is no key input, the obtained value will be 0.
Return	>= 0 ERR_DEV_NOT_OP EN ERR_INVALID_PAR AM Others	Succeeded, and the returned value represents the number of “*” in the PIN input interface. PED device is not open Invalid parameter Refer to the PED Return Code List
Instruction	<ol style="list-style-type: none"> 1. This function does not support listening for PIN input events by multi-process. 2. When first calling this function, PED will create a listening service for PIN input event, clear the key buffer when PED service is closed, and deregister the PED service. Therefore, users should call this function before calling PIN input function to get the completed key buffer. 3. PED will clear the key buffer when a new PIN input event occurs. 	

6.5 MK/SK

6.5.1 OsPedWriteKey

Prototype	int OsPedWriteKey(const unsigned char *KeyBlock);		
Function	Write in a key, including TLK, TMK, TWK, SM4_TMK; write in and diverge the SM4_TMK key; and verify the validity of keys using KCV.		
Parameters	KeyBlock[Input]	1 byte	Format: 0x03
		1 byte	<p>SrcKeyType:</p> <ul style="list-style-type: none"> ▪ PED_TLK ▪ PED_TMK ▪ PED_TPK/PED_TAK/PED_TDK ▪ PED_SM4_TMK ▪ PED_SM4_TPK/PED_SM_TAK/PED_SM4_TDK
		1 byte	<p>SrcKeyIdx:</p> <ul style="list-style-type: none"> ▪ When SrcKeyType = PED_TLK, SrcKeyIdx = 1; ▪ When SrcKeyType = PED_TMK, SrcKeyIdx = [1~100]; ▪ When ucSrcKeyType=PED_TPK/PE_D_TAK/PED_TDK, ucSrcKeyIdx = [1~100]; ▪ When ucSrcKeyType = PED_SM4_TMK, ucSrcKeyIdx = [1~100]; ▪ When ucSrcKeyType = PED_SM4_TPK/PED_SM4_TAK/PED_SM4_TDK, ucSrcKeyIdx = [1~100];
		1 byte	<p>DstKeyIdx:</p> <ul style="list-style-type: none"> ▪ When DstKeyType = PED_TLK, DstKeyIdx = 1; ▪ When DstKeyType =

		<p>PED_TMK, DstKeyIdx = [1~100];</p> <ul style="list-style-type: none"> ▪ When DstKeyType = PED_TPK or PED_TAK or PED_TDK or PED_TDFK, DstKeyIdx = [1~100].
	7 bytes	Reserved domain; random number.
	1 byte	<p>DstKeyType:</p> <ul style="list-style-type: none"> ▪ PED_TLK ▪ PED_TMK ▪ PED_TPK/PED_TAK/PED_TDK ▪ PED_SM4_TMK ▪ PED_SM4_TPK/PED_SM4_TAK/PED_SM4_TDK ▪ PED_TDFK
	1 byte	<p>DstKeyLen: 8/16/24</p> <ul style="list-style-type: none"> ▪ When DstKeyType is PED_SM4_TMK/PED_SM4_TPK/PED_SM4_TAK/PED_SM4_TDK, DstKeyLen=16.
	8/16/24 bytes	<p>DstKeyValue.</p> <p>Destination key plaintext / ciphertext.</p>
	1 byte	<p>KcvMode:</p> <p>0x00: No verification.</p> <p>0x01: KCV is the first 3 bytes of ciphertext after encrypting the 8-byte 0x00 with DES/TDES algorithm.</p> <p>0x02: KCV is the first 3 bytes of ciphertext after odd parity check on the plaintext of key and DES/TDES encryption on the "\x12\x34\x56\x78\x90\x12\x34\x56".</p> <p>0x03: KCV is an 8-byte MAC value that is a result of</p>

Return		<p>specified mode of MAC calculation on [ciphertext of destination key + input KcvData] with source key pair.</p> <p>0x04: KCV is the first 4 bytes of ciphertext after encrypting the 16-byte 0x00 with SM4 algorithm.</p> <p>Note: Mode 0x01, 0x02 and 0x03 can only be used for checking MK/SK key injection. Mode 0x04 is only used for checking SM4 key injection.</p>
	128 bytes	<p>KcvData:</p> <ul style="list-style-type: none"> ▪ When <i>KcvMode</i> is 0x00/0x01/0x02, padding with random numbers. ▪ When <i>KcvMode</i> is 0x03, the first byte of <i>KcvData</i> is the length of KCV data involved in the calculation, and the rest is KCV data. The first byte after the KCV data represents the mode of MAC calculation.
	8 bytes	<ul style="list-style-type: none"> ▪ When <i>KcvMode</i> = 0x00, padding with random numbers. ▪ When <i>KcvMode</i> = 0x01/0x02/0x03/0x04, <i>KcvValue</i> points to the KCV value.
	10 bytes	Padding with random data.

	ERR_PED_TAMPERED	PED is locked.
	ERR_PED_NO_MORE_BUF	System memory is not enough.
	ERR_PED_NORMAL_ERROR	PED normal error (KeyBlock Format error).
	ERR_PED_DERIVE_ERROR	Key divergence error.
	ERR_PED_KCV_MODE_ERROR	PED KCV verification mode error.
	ERR_PED_KCV_CHECK_FAIL	PED KCV verification failed.
	Others	Refer to the PED Return Code List .
Instruction	<p>When calling this function to write the ciphertext and plaintext of a key to a specific index position of the specific key type area, please pay attention to the following notes:</p> <ol style="list-style-type: none"> 1. When <i>SrcKeyIdx</i>=0, <i>DstKeyValue</i> is regarded as the plaintext of key. And the system will directly write <i>DstKeyValue</i> to <i>DstKeyIdx</i> in <i>DstKeyType</i> area without judging <i>SrcKeyType</i> and <i>SrcKeyIdx</i>. 2. Only when PED_TLK does not exist, plaintext is allowed to be written and key is allowed to be downloaded. 3. When PED_TLK exists, plaintext is not allowed to be written in and key is not allowed to be downloaded. The length of PED_TLK can either be 16 or 24 bytes and the 8-byte key is not allowed to be injected. 4. Before writing in PED_TLK, all keys that have been downloaded must be cleared. 5. If <i>SrcKeyIdx</i> is valid, <i>DstKeyValue</i> will be regarded as the key ciphertext. And the system will decrypt <i>DstKeyValue</i> with <i>SrcKeyIdx</i> key in <i>SrcKeyType</i> key area and write it to <i>DstKeyIdx</i> in <i>DstKeyType</i> area. (<i>DstKeyType</i> >= <i>SrcKeyType</i>) 6. <i>DstKeyLen</i> can only be 8, 16 or 24 bytes. If <i>DstKeyLen</i> = 8 bytes, the key can only be used for DES calculation. If <i>DstKeyLen</i> = 16 or 24 bytes, the key can be used for TDES calculation. (<i>DstKeyLen</i> <= <i>SrcKeyLen</i>) 7. Key type is specified by <i>DstKeyType</i>. When <i>DstKeyType</i>=PED_TPK, the key can only be used for PIN block calculation; when <i>DstKeyType</i>=PED_TAK, the key can only be used for MAC calculation; When 	

	<p><i>DstKeyType</i>=PED_TDK, the key can only be used for DES/TDES encryption and decryption to limit their application field and ensure the uniqueness of their function.</p> <ol style="list-style-type: none"> 8. KCV is used to verify plaintext of key. If plaintext is typed-in directly and the <i>KcvMode</i> of <i>KeyIn</i> is not 0, the system will perform KCV verification on plaintext of key according to the specified <i>KcvMode</i>. 9. Prolin-phoenix-2.5 supports the uniqueness of key injection, while Prolin-2.4 allows the same key to exist in different key slots. 10. When PED_TLK exists and its length isn't 16 bytes, PED_SM4_TMK/PED_SM4_TPK/PED_SM4_TAK/PED_SM4_TDK can not be injected. In SM algorithm key system, PED_TLK must be 16 bytes, too.
--	--

NOTE

1. The valid *KeyBlock* must be 184 bytes.
2. The incoming parameters must be valid; otherwise, an error may occur.

6.5.2 OsPedWriteTIK

Prototype	int OsPedWriteTIK(const unsigned char *KeyBlock);		
Function	Write in a TIK key, and it is optional to verify the key by KCV.		
Parameters	KeyBlock[Input]	1 byte	Format: 0x03
		1 byte	SrcKeyType: ▪ PED_TLK
		1 byte	SrcKeyIdx: ▪ When SrcKeyType = PED_TLK, SrcKeyIdx = 1; ▪ When writing in plaintext, SrcKeyIdx = 0.
		1 byte	DstKeyIdx: DstKeyIdx = [1~100];
		7 bytes	Reserved domain. Random number.
		1 byte	DstKeyType:

		<ul style="list-style-type: none"> ▪ PED_TIK
1 byte	DstKeyLen: 8/16	
24 bytes	DstKeyValue. The destination key plaintext / ciphertext	
1 byte	KcvMode: 0x00: No verification. 0x01: KCV is the first 3 bytes of ciphertext after encrypting the 8-byte 0x00 with DES/TDES algorithm. 0x02: KCV is the first 3 bytes of ciphertext after odd parity check on the plaintext of key and DES/TDES encryption on “\x12\x34\x56\x78\x90\x12\x34\x56”. 0x03: The KCV is an 8-byte MAC value, that is, a result of specified mode of MAC calculation on [ciphertext of destination key + input KcvData] with source key pair.	
128 bytes	KcvData: <ul style="list-style-type: none"> ▪ When the KcvMode is 0x00/0x01/0x02, padding with random numbers. ▪ When the KcvMode is 0x03, the first byte of KcvData is the length of KCV data involved in the calculation, and the rest is KCV data. The first byte after the KCV data represents the mode of MAC calculation. 	
8 bytes	<ul style="list-style-type: none"> ▪ When KcvMode = 0x00, padding with random numbers. ▪ When KcvMode = 0x01/0x02/0x03, KcvValue points to the KCV value. 	
10 bytes	Initialize KSN.	

Return	<p>RET_OK Succeeded.</p> <p>ERR_DEV_NOT_OPE N PED device is not open.</p> <p>ERR_INVALID PARA M Invalid parameter</p> <p>Others Refer to PED Return Code List.</p>
Instruction	<p>When calling this function to write the ciphertext or plaintext of a key to a specific index position of the specific key type area, please pay attention to the following notes:</p> <ol style="list-style-type: none"> 1. When <i>SrcKeyIdx</i>=0, <i>DstKeyValue</i> is regarded as the plaintext of key. The system will directly write the <i>DstKeyValue</i> to <i>DstKeyIdx</i> in <i>DstKeyType</i> area without judging <i>SrcKeyType</i> and <i>SrcKeyIdx</i>. 2. Only when PED_TLK does not exist, plaintext is allowed to be written and key is allowed to be downloaded. 3. When PED_TLK exists, plaintext is not allowed to be written and key is not allowed to be downloaded. 4. If <i>SrcKeyIdx</i> is valid, <i>DstKeyValue</i> is regarded as the ciphertext of the key. The system will decrypt <i>DstKeyValue</i> with <i>SrcKeyIdx</i> key in <i>SrcKeyType</i> area and write the key to <i>DstKeyIdx</i> in <i>DstKeyType</i>. (<i>DstKeyType</i> >= <i>SrcKeyType</i>) 5. KVC is used to verify plaintext. If plaintext is typed-in directly, and the <i>KcvMode</i> of <i>KeyIn</i> is not 0, the system will perform KCV verification on plaintext according to the specified <i>KcvMode</i>.

NOTE

The incoming parameters must be valid; otherwise, an error may occur. The valid *KeyBlock* must be 184 bytes.

6.5.3 OsPedWriteKeyVar

Prototype	<pre>int OsPedWriteKeyVar(int KeyType, int SrcKeyIdx, int DstKeyIdx, const unsigned char *KeyVar);</pre>	
Function	<p>Use the key plaintext specified by source key index and key type to operate with data string and write the result to specified destination key index in the same key type area.</p>	
Parameters	KeyType	Key types:

		<ul style="list-style-type: none"> ▪ PED_TMK ▪ PED_TPK ▪ PED_TAK ▪ PED_TDK
	SrcKeyIdx	Source key index. The valid range is from 1 to 100.
	DstKeyIdx	Destination key index. The valid range is from 1 to 100.
	KeyVar[Input]	A 24-byte string involved in calculation. The string is used to xor with the source key. The string length should be the same as the key's, and it is extensible.
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter
	Others	Refer to PED Return Code List .
Instruction		

NOTE

Please refer to AS2805.6.

6.5.4 OsPedSetAsteriskLayout

Prototype	<pre>int OsPedSetAsteriskLayout(int x, int y, int fontSize, int fontColor, uchar align);</pre>	
Function	Set the layout attributes of asterisk while inputting PIN.	
	x	X-coordinate
	y	Y-coordinate
Parameters	fontSize	Font size of asterisk: <ul style="list-style-type: none"> ▪ fontSize = 16: the character size is 16 dots; ▪ fontSize = 24: the character size is 24 dots; ▪ fontSize = 32: the character size is 32 dots; ▪ fontSize = 48: the character size is 48 dots.

		Note : The asterisk is displayed with PED internal font, which has no relation with the installed system font library.
	fontColor	Font color of asterisk. Macro definition RGB (_r, _g, _b) is used; the 16-bit color value will be generated according to the input three primary colors.
	align	Alignment: <ul style="list-style-type: none">▪ PED_ASTERISK_ALIGN_LEFT: Display asterisk from left to right; the left starting position is fixed.▪ PED_ASTERISK_ALIGN_CENTER: Display asterisk from the middle to both right side and left side symmetrically; the middle position is fixed.▪ PED_ASTERISK_ALIGN_RIGHT: Display asterisk from right to left; the right starting position is fixed.
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OP EN	PED device is not open.
	ERR_INVALID_PAR AM	Invalid parameter.
	Others	Refer to the PED Return Code List .
Instruction	<ol style="list-style-type: none">1. This function is only used to display asterisk, while PIN input interface is displayed by application.2. This function should be called to set the layout of asterisk before calling OsPedVerifyPlainPin(), OsPedVerifyCipherPin() OsPedGetPinBlock() or OsPedGetPinDukpt ().	

6.5.5 OsPedSetPinIconLayout

Prototype	<pre>int OsPedSetPinIconLayout(int X, int Y, int Interval, uchar IconNum, char *PinIcon, int PinIconLen, char *PinIconBG, int PinIconBGLen);</pre>
-----------	---

Function	Set the layout attributes of foreground and background icons.	
Parameters	X [Input]	The initial x coordinate of foreground and background icons.
	Y [Input]	The initial y coordinate of foreground and background icons.
	Interval [Input]	The interval of two adjacent icons.
	IconNum [Input]	The number of background icon
	PinIcon [Input]	Absolute path of foreground icon, BMP and PNG icons are supported.
	PinIconLen [Input]	The length of absolute path of foreground icon, the valid length is not more than 256 bytes.
	PinIconBG [Input]	Absolute path of background icon, BMP and PNG icons are supported, and it can be set to NULL if icon is not needed.
	PinIconBGLen [Input]	The path length of absolute path of background icon, the valid length is not more than 256 bytes, and set it to 0 when background icon is not needed.
Return	RET_OK Succeeded ERR_DEV_NOT_OPE N PED device is not open. ERR_INVALID_PARAM Invalid parameters Others Refer to the PED Return Code List	
Instruction	1. If user calls this function before inputting PIN, the icon specified by the parameter will be used as PIN background and the foreground icon will replace asterisk to prompt PIN input. 2. After calling OsPedSetAsteriskLayout() , the asterisk will replace icon to prompt PIN input and the PIN icon attribute settings will all be cleared; 3. Enable/disable the dynamic center display function of the foreground image by setting the value of "persist.sys.ped.pin.icon.align" in the registry shown as below: <ul style="list-style-type: none"> ● When persist.sys.ped.pin.icon.align=1, while inputting PIN, the foreground image which is passed in after calling OsPedSetPinIconLayout() will be centered and displayed dynamically, and the background image will not be displayed at the same time; 	

- When persist.sys.ped.pin.icon.align=0, the foreground image will restore to normal display, and the default value is "0".

6.5.6 OsPedGetPinBlock

Prototype	<pre>int OsPedGetPinBlock(int KeyIdx, const unsigned char *DataIn, const char *ExpPinLen, int Mode, unsigned long TimeoutMs, unsigned char *PinBlock);</pre>	
Function	Scan input PIN whose length is specified by <i>ExpPinLenIn</i> within specified time and output PIN block which is encrypted with algorithm specified by <i>Mode</i> .	
Parameters	KeyIdx	TPK Index. The value range is from 1 to 100.
	DataIn[Input]	<ul style="list-style-type: none"> When <i>Mode</i>=0x00, <i>DataIn</i> points to the 16-bit primary account number which is generated after shifting card number. When <i>Mode</i>=0x01, <i>DataIn</i> is meaningless, and it can be an arbitrary value. PED will use random number instead of <i>DataIn</i> in the calculation of PinBlock. When <i>Mode</i>=0x02, <i>DataIn</i> is the 16-bit primary account number which is generated after shifting card number. When <i>Mode</i>=0x03, <i>DataIn</i> is the transaction serial number ISN [6 Bytes, ASCII code]. When <i>Mode</i>=0x10, <i>DataIn</i> is meaningless, and it can be an arbitrary value. PED will use random number instead of <i>DataIn</i> in the calculation of PinBlock. When <i>Mode</i>=0x50, <i>DataIn</i> is meaningless, and it can be an arbitrary value.
	ExpPinLen[Input]	A pointer to valid password length string, which is an enumeration set from 0 to 12.

		<p>Application enumerates all valid password length and separate each length with “,”. For example, if no PIN, 4 and 6 digits of PIN are allowed, the string will be set to “0, 4, 6”. “0” means no PIN is required and can return directly by pressing “Enter”.</p> <p>When Mode=0x50, the valid password length string that can be entered is an enumeration set of 0~16.</p>
	Mode	<p>PIN block format:</p> <p>Format of PinBlock encrypted with DES-ECB (3DES-ECB) algorithm:</p> <ul style="list-style-type: none"> ▪ 0x00 0x00 ISO9564 format 0 ▪ 0x01 0x01 ISO9564 format 1 ▪ 0x02 0x02 ISO9564 format 3 ▪ 0x03 0x03 HK EPS proprietary format <p>Format of PinBlock encrypted with DES-CBC (3DES-CBC) algorithm:</p> <ul style="list-style-type: none"> ▪ 0x50 0x50 SIN MCCS proprietary format <p>Format of PinBlock encrypted with AES-ECB algorithm:</p> <ul style="list-style-type: none"> ▪ 0x10: the first 8 bytes of plaintext PinBlock is in ISO9564 format 1, the last 8 bytes will be filled in #PKCS7 which is a 8-byte of 0x08.
	TimeoutMs	<p>Timeout of PIN input [unit:ms]</p> <p>The valid timeout ranges from 0 to 300000 ms. “0” means no timeout. PED has no timeout control.</p>
	PinBlock[Output]	<p>Point to the generated PIN block.</p> <p>The valid length is 8 or 16 bytes.</p> <p>When Mode=0x10, the length of PinBlock is 16 bytes.</p>
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to PED Return Code List .
Instruction	<ol style="list-style-type: none"> 1. Press [CANCEL]key to cancel input. 2. Call OsPedGetPinBlock() will operate the framebuffer, 	

	<p>which might cause the resource conflict. If there is some kind of exception on the interface of application which is running XUI, call XuiSuspend() to suspend XUI before calling OsPedGetPinBlock(), after a while, call XuiResume() to resume XUI. If the background of PIN input is set by calling OsPedSetPinBg() before the PIN input interface is called, XuiSuspend() and XuiResume() cannot be called.</p> <p>3. The key which is used for encrypting PinBlock with DES(3DES) algorithm is PED_TPK, and the key which is used for encrypting PinBlock with AES algorithm is PED_AES_TPK.</p>
--	---

6.5.7 OsPedUpdatePinBlock

Prototype	<pre>int OsPedUpdatePinBlock(int UpdateFlag, const unsigned char *KeyInfo, const unsigned char *DataIn, unsigned char *PinBlock, int Mode);</pre>								
Function	Recalculate PIN block and replaced TPK according to <i>UpdateFlag</i> .								
Parameters	<table border="1"> <tr> <td style="background-color: #4f81bd; color: white;">UpdateFlag</td><td>0: Do not replace TPK, Others: Replace TPK</td></tr> <tr> <td style="background-color: #4f81bd; color: white;">KeyInfo[Input]</td><td>Please refer to the definition of <i>KeyBlock</i> in OsPedWriteKey(). The valid length is 184 bytes. When <i>UpdateFlag</i> is 0, only <i>DstKeyIdx</i> is valid in <i>KeyBlock</i>. PIN block is recalculated with TPK specified by <i>DstKeyIdx</i>.</td></tr> <tr> <td style="background-color: #4f81bd; color: white;">DataIn[Input]</td><td> <ul style="list-style-type: none"> When <i>UpdateFlag</i> is 0 and <i>Mode</i>=0x03, <i>DataIn</i> is the transaction serial number ISN [6 Bytes, ASCII code]. When <i>UpdateFlag</i> is 0 and <i>Mode</i>=0x00, the first 16 bytes of <i>DataIn</i> is the old PAN, and the last 16 bytes of <i>DataIn</i> is the new PAN. PAN is the 16-bit primary account number which is generated after shifting card number. When <i>UpdateFlag</i> is not 0, <i>DataIn</i> can be NULL. </td></tr> <tr> <td style="background-color: #4f81bd; color: white;">PinBlock[Output]</td><td>Input original PIN block data and output new PIN block. The valid value is 8 bytes.</td></tr> </table>	UpdateFlag	0: Do not replace TPK, Others: Replace TPK	KeyInfo[Input]	Please refer to the definition of <i>KeyBlock</i> in OsPedWriteKey(). The valid length is 184 bytes. When <i>UpdateFlag</i> is 0, only <i>DstKeyIdx</i> is valid in <i>KeyBlock</i> . PIN block is recalculated with TPK specified by <i>DstKeyIdx</i> .	DataIn[Input]	<ul style="list-style-type: none"> When <i>UpdateFlag</i> is 0 and <i>Mode</i>=0x03, <i>DataIn</i> is the transaction serial number ISN [6 Bytes, ASCII code]. When <i>UpdateFlag</i> is 0 and <i>Mode</i>=0x00, the first 16 bytes of <i>DataIn</i> is the old PAN, and the last 16 bytes of <i>DataIn</i> is the new PAN. PAN is the 16-bit primary account number which is generated after shifting card number. When <i>UpdateFlag</i> is not 0, <i>DataIn</i> can be NULL. 	PinBlock[Output]	Input original PIN block data and output new PIN block. The valid value is 8 bytes.
UpdateFlag	0: Do not replace TPK, Others: Replace TPK								
KeyInfo[Input]	Please refer to the definition of <i>KeyBlock</i> in OsPedWriteKey(). The valid length is 184 bytes. When <i>UpdateFlag</i> is 0, only <i>DstKeyIdx</i> is valid in <i>KeyBlock</i> . PIN block is recalculated with TPK specified by <i>DstKeyIdx</i> .								
DataIn[Input]	<ul style="list-style-type: none"> When <i>UpdateFlag</i> is 0 and <i>Mode</i>=0x03, <i>DataIn</i> is the transaction serial number ISN [6 Bytes, ASCII code]. When <i>UpdateFlag</i> is 0 and <i>Mode</i>=0x00, the first 16 bytes of <i>DataIn</i> is the old PAN, and the last 16 bytes of <i>DataIn</i> is the new PAN. PAN is the 16-bit primary account number which is generated after shifting card number. When <i>UpdateFlag</i> is not 0, <i>DataIn</i> can be NULL. 								
PinBlock[Output]	Input original PIN block data and output new PIN block. The valid value is 8 bytes.								

	Mode	0x00: ISO9564 format 0; 0x03: HK EPS proprietary format [Refer to Appendix 2 EPS PINBLOCK Format].
Return	RET_OK ERR_DEV_NOT_OPEN ERR_INVALID_PARAM Others	Succeeded. PED device is not open. Invalid parameter. Refer to the PED Return Code List .
Instruction	This function only applies to EPS.	

6.5.8 OsPedGetMac

Prototype	<code>int OsPedGetMac(int KeyIdx, const unsigned char *DataIn, int DataInLen, unsigned char *Mac, int Mode);</code>	
Function	Calculate data with specified <i>Mode</i> and MAC key.	
Parameters	KeyIdx	TAK index. The valid <i>KeyIdx</i> ranges from 1 to 100.
	DataIn[Input]	Data that needs to do MAC operation. The valid length is not more than 8192 bytes.
	DataInLen	The length of data package. If the length is not a multiple of 8, 0x00 will be padded automatically.
	Mac[Output]	MAC output. The valid length is 8 bytes
	Mode	<i>DataIn</i> is blocked into 8-byte as BLOCK1, BLOCK2, BLOCK3, etc. 0x00: Perform DES/TDES encryption on BLOCK1 with MAC key and bitwise xor the encryption result with Block2. Then perform DES/TDES encryption on the xor result with TAK key to get an 8-byte encrypted result. 0x01: Bitwise xor BLOCK1 with BLOCK2; use the xor result to bitwise xor with BLOCK3; continue in this way, we can get an 8-byte xor result. Then perform DES/TDES encryption with TAK on the final xor result.

		0x02: According to ANSIX9.19 standard, perform DES encryption on BLOCK1 with TAK (only take the first 8 bytes of the key). Then bitwise xor the encrypted result with BLOCK2, and perform DES encryption on the xor result with TAK to get an 8 bytes encryption result. Finally perform DES/TDE encryption on the 8-byte result.
Return	RET_OK ERR_DEV_NOT_OPE N ERR_INVALID_PARA M Others	Succeeded. PED device is not open. Invalid parameter. Refer to PED Return Code List .
Instruction		

6.5.9 OsPedDes

Prototype	<code>int OsPedDes(int KeyIdx, unsigned char * InitVector, const unsigned char *DataIn, int DataInLen, unsigned char *DataOut, int Mode);</code>	
Function	Use DES/TDES algorithm and TDK to encrypt or decrypt data whose length is specified by <i>DataInLen</i> and output ciphertext or plaintext into <i>DataOut</i> . A specified TDK can be used either in encryption or decryption.	
Parameters	KeyIdx	TDK index. The valid <i>KeyIdx</i> ranges from 1 to 100.
	InitVecto[Input]	Initialization vector. The valid length is 8 bytes. When <i>Mode</i> =0x02/0x03/0x04/0x05, the initialization vector is needed in encryption or decryption; when <i>InitVecto</i> is NULL, the initialization vector is set to “\x00\x00\x00\x00\x00\x00\x00\x00” by default. When <i>Mode</i> =0x00/0x01, the initialization vector is not needed and can be set to NULL.
	DataIn[Input]	A pointer to the data that needs to be

		calculated.
	DataInLen	Length of data that needs to be calculated.[unit: byte] The valid data length should be less than or equal to 1024 bytes. When Mode=0x00~0x05, the data length must be a multiple of 8.
	DataOut[Output]	A pointer to the data that has been calculated.
	Mode	<ul style="list-style-type: none"> ▪ 0x00: ECB Decryption ▪ 0x01: ECB Encryption ▪ 0x02: CBC Decryption ▪ 0x03: CBC Encryption ▪ 0x04: OFB Decryption ▪ 0x05: OFB Encryption ▪ 0x06: CFB8 Decryption ▪ 0x07: CFB8 Encryption
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to PED Return Code List .
Instruction		

NOTE

The key length decides the encryption or decryption algorithm is DES or TDES.Whether to use DES or TDES algorithm depends on the key length.

6.5.10 OsPedGetKcv

Prototype	<pre>int OsPedGetKcv(int KeyType, int KeyIdx, int KcvMode, int KcvDataLen, unsigned char *KcvData, unsigned char *Kcv);</pre>
Function	Get KCV value to verify the key for both sides of session: when the <i>KeyType</i> is not TIK, KCV is the first 3-byte of data encrypted

	with specified key and algorithm; when the <i>KeyType</i> is TIK, KCV is an 8-byte calculation result which is injected together with TIK.	
Parameters	KeyType	<p>Key type:</p> <ul style="list-style-type: none"> ▪ PED_TLK ▪ PED_TMK ▪ PED_TAK ▪ PED_TPK ▪ PED_TDK ▪ PED_TIK ▪ PED_SM4_TMK ▪ PED_SM4_TPK ▪ PED_SM4_TAK ▪ PED_SM4_TDK
	KeyIdx	<p>Key index number, for example:</p> <ul style="list-style-type: none"> ▪ TLK can only be 1. ▪ TMK ranges from 1 to 100. ▪ TWK ranges from 1 to 100. ▪ TIK ranges from 1 to 100.
	KcvMode	<p>KCV check mode.</p> <p>0x00: Calculate the KCV of the key with DES algorithm ;</p> <p>0x04: Calculate the KCV of SM4 key with SM4 algorithm, and the key types can only be SM4 series keys (KeyType can only be PED_SM4_TMK/PED_SM4_TPK/PED_SM4_TAK/PED_SM4_TDK).</p>
	KcvDataLen	<p>Length of data that needs to be calculated.</p> <p>The valid data length ranges from 0 to 128bytes and must be a multiple of 8. When <i>KeyType</i> is TIK, <i>KcvDataLen</i> can be set to 0; When the KCV check mode is 0x04, the data length must be divisible by 16.</p>
	KcvData[Input]	<p>A pointer to the data that needs to be calculated.</p> <p>NULL pointer is valid when <i>KeyType</i> is TIK.</p>
	Kcv[Output]	<p>KCV.</p> <p>The valid length of <i>Kcv</i> is 8 bytes when <i>KeyType</i> is TIK, and 3 bytes for rest <i>KeyType</i>.</p>
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OP	PED device is not open.

	EN	
	ERR_INVALID_PAR	Invalid parameter.
	AM	
	Others	Refer to the PED Return Code List .
Instruction		

6.5.11 OsPedDeriveKey

Prototype	<code>int OsPedDeriveKey(int SrcKeyType, int SrcKeyIdx, int DstKeyType, int DstFromKeyIdx, int DstToKeyIdx, int Mode);</code>	
Function	Use the key specified by <i>SrcKeyIdx</i> to encrypt or decrypt the key specified by <i>DstFromKeyIdx</i> , derive a new key and save it at the destination key specified by <i>DstToKeyIdx</i> .	
Parameters	SrcKeyType	Source key types: <ul style="list-style-type: none">▪ PED_TLK▪ PED_TMK▪ PED_TAK▪ PED_TPK▪ PED_TDK
	SrcKeyIdx	Index number of source key, for example: <ul style="list-style-type: none">▪ TLK can only be 1.▪ TMK ranges from 1 to 100.▪ TWK ranges from 1 to 100.
	DstKeyType	Destination key types : <ul style="list-style-type: none">▪ PED_TLK▪ PED_TMK▪ PED_TAK▪ PED_TPK▪ PED_TDK
	DstFromKeyIdx	Source index of destination key.
	DstToKeyIdx	Destination index of destination key.
	Mode	0x00: DES/TDES decryption; 0x01: DES/TDES encryption.
Return	RET_OK Succeeded.	

	<p>ERR_DEV_NOT_OPEN PED device is not open.</p> <p>ERR_INVALID_PARAM Invalid parameter.</p> <p>Others Refer to PED Return Code List.</p>
Instruction	The level of source key type cannot be lower than destination key.

6.5.12 OsPedSetPinBg

Prototype	<code>int OsPedSetPinBg(uchar Mode, const uchar *Bg, int BgLen);</code>	
Function	Set the background of the PIN input interface using <i>Framebuffer</i> data or image data.	
Parameters	Mode[Input]	<p>The setting mode of the PIN input background.</p> <ol style="list-style-type: none"> 1. If <i>Mode</i>=0x00, the setting of this function will be cleared. 2. If <i>Mode</i>=0x01, the background of the PIN input interface will be set using <i>Framebuffer</i> data. 3. If <i>Mode</i>=0x02, the background of the PIN input interface will be set using image data.
	Bg[Input]	The buffer to store <i>Framebuffer</i> data or image data.
	BgLen[Input]	The buffer length of the stored <i>Framebuffer</i> data or image data.
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to PED Return Code List .
Instruction	<ol style="list-style-type: none"> 1. When <i>Mode</i>=0x00, parameter <i>Bg</i> and <i>BgLen</i> are meaningless; 2. When <i>Mode</i> is not equal to 0x00, parameter <i>Bg</i> cannot be null and <i>BgLen</i> > 0; 3. The size of the <i>Framebuffer</i> data which is provided by users must be consistent with the <i>Framebuffer</i> data size of the screen, and the image resolution must be consistent with the resolution of the screen; 4. The image data represents the data which is read from image file by users and cached into memory, and only 	

- “bmp” and “png” format images are supported;
5. The PIN input background set in this interface will be cached in PED until it is cleared or the terminal is powered off;
 6. XUI provides *XuilmgToFrameBuffer()* to get the *Framebuffer* data of the whole screen, and it can be used with this interface. The detailed description and corresponding sample code are in the 10.2 section of “Prolin Application Development Manual”.

6.5.13 OsPedCustomKeypad

Prototype	int OsPedCustomKeypad(char *IconPath, uchar *KeypadColor, uchar Mode);	
Function	Customize the PED soft keypad.	
Parameters	IconPath [Input]	The absolute path of keypad icons, and relative path is not supported. It cannot be more than 256 bytes and it can be NULL. When <i>IconPath</i> is NULL, the default icon will be used.
	KeypadColorBg [Input]	RGB value of the keypad background color, it has 4 bytes and it can be NULL. When <i>KeypadColorBg</i> is NULL, the default keypad background will be used.
	Mode [Input]	When <i>Mode</i> = 0x00, the default icon and keypad background color will be used. When <i>Mode</i> = 0x01, the icon and keypad background color specified by users will be used.
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_FILE_NOT_EXIST	Resource file of keypad icon does not exist.
	ERR_VERIFY_SIGN_FAILED	Failed to verify the resource file of keypad icon.
	Others	Refer to PED Return Code List .
Instruction	1. Before inputting PIN, call this function to set the icon in the specified path as the PIN input keypad icon, and to set the canvas background color as the PIN input keypad	

	<p>background color.</p> <ol style="list-style-type: none"> 2. Support replacing parts of keypad icons. The system will only obtain the icon resource file which corresponds to relevant model in <i>KeypadPath</i>, if no corresponding keypad icon resource file is found, the system will use the default resource file. 3. Each keypad icon provided by users needs to be signed by MF_PVK. If the signature verification is failed, it indicates the setting is failed, and <i>ERR_VERIFY_SIGN_FAIL</i> will be returned. 4. When Mode = 0x01, <i>KeypadPath</i> and <i>KeypadColorBg</i> cannot be NULL at the same time. If none of keypad icon in <i>KeypadPath</i> meets the requirement, this API will cancel the setting and return <i>ERR_FILE_NOT_EXIST</i>. 5. Please refer to section 10.3 in “Prolin Application Development Manual” for the usage and requirements of relevant models’ keypad icon resource. 6. Soft keypad background color does not support transparency temporarily, so only <i>RGB</i> in <i>KeypadColorBg</i> takes effect, and A (Alpha) has no effect.
--	---

6.6 DUKPT

6.6.1 OsPedGetPinDukpt

Prototype	<pre>int OsPedGetPinDukpt(int GroupIdx, const unsigned char *DataIn, const char *ExpPinLen, int Mode, unsigned long TimeoutMs, unsigned char *Ksn, unsigned char *PinBlock);</pre>		
Function	Scan input PIN and output PIN block which is a result of DUKPT PIN key calculation within specified time.		
Parameters	GroupIdx	DUKPT group index number. The valid <i>GroupIdx</i> ranges from 1 to 100.	
	DataIn[Input]	<ol style="list-style-type: none"> 1. If <i>Mode</i>=0x20, <i>DataIn</i> points to the 16-bit primary account number after shifting card number. 2. If <i>Mode</i>=0x21, <i>DataIn</i> has no meaning. 3. If <i>Mode</i>=0x22, <i>DataIn</i> points to the 16-bit primary account number after shifting card 	

	<p>number.</p> <p>4. If <i>Mode</i>=0x23, <i>DataIn</i> is ISN [6 Bytes, ASCII code]</p>
ExpPinLen[Input]	<p>A pointer to valid password length string which is an enumeration set from 0 to 12. Application enumerates all valid password length and separates each length with ",". For example, if no PIN, 4 and 6 digits of PIN are allowed, the string will be set to "0, 4, 6". "0" means no PIN is required and can return directly by pressing "Enter".</p>
Mode	<p>Choose the format of PIN block:</p> <p>0x20 ISO9564 format 0, KSN does not increase 1 automatically.</p> <p>0x21 ISO9564 format 1, KSN does not increase 1 automatically.</p> <p>0x22 ISO9564 format 3, KSN does not increase 1 automatically.</p> <p>0x23 HK EPS format, KSN does not increase 1 automatically.</p>
TimeoutMs	<p>Timeout of PIN input[unit:ms]</p> <p>The valid timeout ranges from 0 to 300000ms. "0" means no timeout, and PED has no timeout control.</p>
Ksn[Output]	<p>A pointer to the current KSN.</p> <p>The valid length of KSN is 10 bytes.</p>
PinBlock[Output]	<p>A pointer to the generated PIN block result.</p> <p>The valid length of PIN block is 8 bytes.</p>
Return	<p>RET_OK Succeeded.</p> <p>ERR_DEV_NOT_OPEN PED device is not open.</p> <p>ERR_INVALID_PARAM Invalid parameter.</p> <p>Others Refer to PED Return Code List .</p>
Instruction	<ol style="list-style-type: none"> When KSN does not increase 1, a DUKPT PIN key can only calculate the PIN block once. Calling OsPedGetPinDukpt() will operate the framebuffer, which might cause the resource conflict. If there is some kind of exception on the interface of application which is running XUI, call XuiSuspend() to suspend XUI before calling OsPedGetPinDukpt(), after a while, call XuiResume() to resume XUI.

6.6.2 OsPedGetMacDukpt

Prototype	<pre>int OsPedGetMacDukpt(int GroupIdx, const unsigned char *DataIn, int DataInLen, unsigned char *Mac, unsigned char *Ksn, int Mode);</pre>												
Function	Calculate MAC with DUKPT key.												
Parameters	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%;">GroupIdx</td><td>DUKPT group index number. The value range is from 1 to 100.</td></tr> <tr> <td>DataIn[Input]</td><td>A pointer to the data that needs to calculate MAC.</td></tr> <tr> <td>DataInLen</td><td>Data length. The valid length is not more than 8192 bytes and must be a multiple of 8; otherwise, padding with "x00" automatically.</td></tr> <tr> <td>Mac[Output]</td><td>A pointer to the generated MAC.</td></tr> <tr> <td>Ksn[Output]</td><td>A pointer to the current KSN.</td></tr> <tr> <td>Mode</td><td> <p><i>DataIn</i> is blocked into 8-byte BLOCK1 , BLOCK2, BLOCK3, etc.</p> <p>0x20: Perform TDES encryption on BLOCK1 with MAC key and bitwise xor the encryption result with Block2. Then perform TDES encryption on the xor result with TAK key to get an 8-byte encrypted result.</p> <p>0x21: Bitwise xor BLOCK1 with BLOCK2; Use the xor result to bitwise xor with BLOCK3; Continue in this way, we can get an 8-byte xor result. Then perform DES/TDES encryption with TAK on the 8-byte xor result.</p> <p>0x22: According to ANSI X9.19 standard, perform DES encryption on BLOCK1 with TAK (only take the first 8 bytes of the key). Then bitwise xor the encrypted result with BLOCK2, and perform DES encryption on the xor result with TAK to get an 8 bytes encryption result. Finally perform TDE encryption on the 8-byte result.</p> </td></tr> </table>	GroupIdx	DUKPT group index number. The value range is from 1 to 100.	DataIn[Input]	A pointer to the data that needs to calculate MAC.	DataInLen	Data length. The valid length is not more than 8192 bytes and must be a multiple of 8; otherwise, padding with "x00" automatically.	Mac[Output]	A pointer to the generated MAC.	Ksn[Output]	A pointer to the current KSN.	Mode	<p><i>DataIn</i> is blocked into 8-byte BLOCK1 , BLOCK2, BLOCK3, etc.</p> <p>0x20: Perform TDES encryption on BLOCK1 with MAC key and bitwise xor the encryption result with Block2. Then perform TDES encryption on the xor result with TAK key to get an 8-byte encrypted result.</p> <p>0x21: Bitwise xor BLOCK1 with BLOCK2; Use the xor result to bitwise xor with BLOCK3; Continue in this way, we can get an 8-byte xor result. Then perform DES/TDES encryption with TAK on the 8-byte xor result.</p> <p>0x22: According to ANSI X9.19 standard, perform DES encryption on BLOCK1 with TAK (only take the first 8 bytes of the key). Then bitwise xor the encrypted result with BLOCK2, and perform DES encryption on the xor result with TAK to get an 8 bytes encryption result. Finally perform TDE encryption on the 8-byte result.</p>
GroupIdx	DUKPT group index number. The value range is from 1 to 100.												
DataIn[Input]	A pointer to the data that needs to calculate MAC.												
DataInLen	Data length. The valid length is not more than 8192 bytes and must be a multiple of 8; otherwise, padding with "x00" automatically.												
Mac[Output]	A pointer to the generated MAC.												
Ksn[Output]	A pointer to the current KSN.												
Mode	<p><i>DataIn</i> is blocked into 8-byte BLOCK1 , BLOCK2, BLOCK3, etc.</p> <p>0x20: Perform TDES encryption on BLOCK1 with MAC key and bitwise xor the encryption result with Block2. Then perform TDES encryption on the xor result with TAK key to get an 8-byte encrypted result.</p> <p>0x21: Bitwise xor BLOCK1 with BLOCK2; Use the xor result to bitwise xor with BLOCK3; Continue in this way, we can get an 8-byte xor result. Then perform DES/TDES encryption with TAK on the 8-byte xor result.</p> <p>0x22: According to ANSI X9.19 standard, perform DES encryption on BLOCK1 with TAK (only take the first 8 bytes of the key). Then bitwise xor the encrypted result with BLOCK2, and perform DES encryption on the xor result with TAK to get an 8 bytes encryption result. Finally perform TDE encryption on the 8-byte result.</p>												

		<ol style="list-style-type: none"> 1. 0x20/0x21/0x22/0x40/0x41/0x42: KSN does not increase1 automatically. 2. 0x40/0x41/0x42: MAC calculation method is the same as 0x20/0x21/0x22's. 3. 0x40/0x41/0x42: Choose to respond to MAC key. 0x20/0x21/0x22: KSN choose to request or respond to MAC key 4. Other values are reserved for extended MAC algorithm.
Return	RET_OK ERR_DEV_NOT_OPEN ERR_INVALID_PARAM Others	Succeeded. PED device is not open. Invalid parameter. Refer to PED Return Code List .
Instruction	If KSN does not increase 1, both the response MAC key and the request MAC key can calculate MAC for unlimited times.	

6.6.3 OsPedDesDukpt

Prototype	<pre>int OsPedDesDukpt(int GroupIdx, int KeyVarType, unsigned char *InitVector, int DataInLen, unsigned char *DataIn, unsigned char *DataOut, unsigned char *Ksn, int Mode);</pre>	
Function	Encrypt or decrypt input data with DUKPT key.	
Parameters	GroupIdx KeyVarType[Input]	DUKPT group index. The valid range is from 1 to 100. 0x00: Use response or request MAC key. 0x01: Use DUKPT DES key 0x02: Use PIN variant to encrypt data. Only ECB and CBC encryption is available, which means <i>Mode</i> can be 0x01 or 0x03. When the length of DUPKT key is 8 bytes, the standard DES algorithm is not adopted but the special DES algorithm defined in ANSI9.24-1998 is adopted.

		0x03: Use response MAC key, only the encryption mode is supported, which means <i>Mode</i> can only be 0x01, 0x03 or 0x05. 0x04: Use DES response key, only the encryption mode is available, that means <i>Mode</i> can only be 0x01, 0x03 or 0x05.
	InitVector[Input]	Initialization vector for encryption/decryption. The valid length is 8 bytes. When <i>Mode</i> =0x02/0x03/0x04/0x05, the initialization vector is needed in encryption or decryption; If <i>InitVector</i> is NULL, the initialization vector is set to “\x00\x00\x00\x00\x00\x00\x00\x00” by default. When <i>Mode</i> =0x00/0x01, initialization vector is not needed and can be set to NULL.
	DataInLen	Length of data that needs to be encrypted or decrypted. The valid length is not more than 8192 bytes.
	DataIn [Input]	A pointer to the input data that needs to be calculated.
	DataOut[Output]	A pointer to the data that has been calculated.
	Ksn[Output]	A pointer to the current KSN. The valid length of KSN is 10 bytes.
	Mode	Encryption/decryption mode: <ul style="list-style-type: none">▪ 0x00: ECB decryption▪ 0x01: ECB encryption▪ 0x02: CBC decryption▪ 0x03: CBC encryption▪ 0x04: OFB decryption▪ 0x05: OFB encryption
Return	RET_OK	Succeeded
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to PED Return Code List .
Instruction	If KSN does not increase 1 automatically, a DUKPT group can	

perform DES operations for 256 times at most when *KeyVarType* is 0x00/0x01; a DUKPT group can only do one DES operation when *KeyVarType* is 0x02.

6.6.4 OsPedGetKsnDukpt

Prototype	int OsPedGetKsnDukpt(int GroupIdx, unsigned char *Ksn);	
Function	Read the current KSN value.	
Parameters	GroupIdx	DUKPT group index. The value range is from 1 to 100.
	Ksn[Output]	A pointer to the current KSN. The valid length of KSN is 10 bytes.
Return	RET_OK Succeeded. ERR_DEV_NOT_OPEN PED device is not open. ERR_INVALID_PARAM Invalid parameter Others Refer to PED Return Code List .	
Instruction		

6.6.5 OsPedIncreaseKsnDukpt

Prototype	int OsPedIncreaseKsnDukpt(int GroupIdx);	
Function	Increase the KSN value of the specified DUKPT group.	
Parameters	GroupIdx	DUKPT group index. The value range is from 1 to 100.
	RET_OK Succeeded. ERR_DEV_NOT_OPEN PED device is not open. ERR_INVALID_PARAM Invalid parameter. Others Refer to PED Return Code List .	
Instruction		

6.7 RSA

6.7.1 OsPedReadRsaKey

Prototype	int OsPedReadRsaKey(int RsaKeyIdx, ST_RSA_KEY *RsaKey);	
Function	Read the RSA public key.	
Parameters	RsaKeyIdx	RSA Key index, the value range is from 1 to 10.
	RsaKey[Output]	A pointer to ST_RSA_KEY structure.
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to PED Return Code List .
Instruction	OsPedReadRsaKey() can only be used to read RSA public key; When reading private key, an error will be returned.	

6.7.2 OsPedWriteRsaKey

Prototype	int OsPedWriteRsaKey(int RsaKeyIdx, ST_RSA_KEY *RsaKey);	
Function	Inject RSA key into PED.	
Parameters	RsaKeyIdx	RSA Key index. The value range is from 1 to 10.
	RsaKey[Input]	A pointer to ST_RSA_KEY structure that will be injected into PED.
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to PED Return Code List .
Instruction	<ol style="list-style-type: none"> The type of RSA key depends on <i>ExponentLen</i> and <i>ModulusLen</i> in <i>RsaKey</i>. It is a private key if <i>ExponentLen</i> is equal to <i>ModulusLen</i> (<i>ExponentLen=ModulusLen</i>); otherwise, it is public key. For a RSA public key, <i>ExponentLen</i> in <i>RsaKey</i> must be 32 which means its exponent length is 4 bytes; if its length is less than 4 bytes, pad with 0x00 at the left side of <i>Exponent</i>. 	



1. The supported RSA key length is not more than 512 bytes.
 2. RSA key can be rewritten at any time.

6.7.3 OsPedRsaRecover

Prototype	<pre><code>int OsPedRsaRecover(int KeyIdx, int DataInLen, unsigned char *DataIn, unsigned char *DataOut, unsigned char *KeyInfo);</code></pre>
Function	Calculate RSA data with RSA key stored in PED.
Parameters	RsaKeyIdx RSA Key index. The value range is from 1 to 10.
	DataInLen Length of data that needs to be calculated. It is equal to the modulus length of RSA key. The valid length ranges from 64 to 512 and must be a multiple of 8.
	DataIn[Input] A pointer to the data that needs to be calculated.
	DataOut[Output] A pointer to the data that has been calculated.
	KeyInfo[Output] A pointer to key information.
Return	RET_OK Succeeded.
	ERR_DEV_NOT_OPEN PED device is not open.
	ERR_INVALID_PARAM Invalid parameter.
	Others Refer to PED Return Code List .
Instruction	

6.7.4 OsPedReadCipherRsaKey

Function	Read the ciphertext of RSA key.	
Parameters	RsaKeyIdx	RSA key index. The valid range is from 1 to 10.
	CipherRsaKey[Output]	A pointer to the ciphertext data of RSA key.
Return	>0	Ciphertext length of RSA key.
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to PED Return Code List .
Instruction		

6.7.5 OsPedWriteCipherRsaKey

Prototype	<code>int OsPedWriteCipherRsaKey(int RsaKeyIdx, int CipherRsaKeyLen, unsigned char *CipherRsaKey);</code>	
Function	Write the ciphertext of RSA key.	
Parameters	RsaKeyIdx	RSA key index. The value range is from 1 to 10.
	CipherRsaKeyLen	Ciphertext length of RSA key. The valid length is not more than 1024 bytes.
	CipherRsaKey[Input]	A pointer to ciphertext of RSA key.
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to PED Return Code List .
Instruction		

6.8 AES

6.8.1 OsPedWriteAesKey

Prototype	<code>intOsPedWriteAesKey(const unsigned char *KeyBlock);</code>
------------------	--

Function	Write AES key's ciphertext or plaintext into a position of specified index within the AES key area, and it is optional to use KCV to verify the validity of the key.		
Parameters	KeyBlock[Input]	1 byte	Format: 0x03
		1 byte	SrcKeyType: ▪ PED_TLK ▪ PED_TMK
		1 byte	SrcKeyIdx: ▪ When <i>SrcKeyType</i> = PED_TLK, <i>SrcKeyIdx</i> = 1; ▪ When <i>SrcKeyType</i> = PED_TMK, <i>SrcKeyIdx</i> = [1~100]; ▪ If <i>ucSrcKeyIdx</i> = 0, key will be written in PED as plaintext.
		1 byte	DstKeyIdx: [1-100].
		7 bytes	Reserved field, random number.
		1 byte	DstKeyType: ▪ PED_TAESK ▪ PED_AES_TPK ▪ PED_AES_TAK ▪ PED_AES_TDK
		1 byte	DstKeyLen: 16/24/32
		32 bytes	DstKeyValue: Destination key plaintext or ciphertext.
		1 byte	KcvMode: 0x00: No verification. 0x01: KCV is the first 3 bytes of ciphertext after encrypting the 16-byte 0x00 with AES/ECB algorithm. 0x02: KCV is the first 3 bytes of ciphertext after odd parity check on the plaintext of key and AES/ECB encryption on “\x12\x34\x56\x78\x90\x12\x34\x56”. 0x03: KCV is an 8-byte MAC value,

			that is, a result of specified mode of MAC calculation on [ciphertext of destination key + input KcvData] with source key.
	128 bytes	KcvData:	<ul style="list-style-type: none"> ▪ When <i>KcvMode</i> is 0x00/0x01/0x02, pad with random numbers. ▪ When <i>KcvMode</i> is 0x03, the first byte of <i>KcvData</i> is the length of KCV data involved in the calculation, and the rest is KCV data. The first byte after the KCV data represents the MAC operation mode.
	8 bytes		<ul style="list-style-type: none"> ▪ When <i>KcvMode</i> = 0x00, pad with random numbers. ▪ When <i>KcvMode</i> =0x01/0x02/0x03, <i>KcvValue</i> points to the KCV value.
	2 bytes		Padding with random number.
Return	RET_OK Succeeded.		
	ERR_DEV_NOT_OPEN	Device is not open.	
	ERR_INVALID_PARAM	Invalid parameter.	
	Others	Refer to PED Return Code List .	
Instruction	<ol style="list-style-type: none"> 1. When <i>SrcKeyIdx</i>=0, <i>DstKeyValue</i> is regarded as the plaintext of key. The system will directly write <i>DstKeyValue</i> into <i>DstKeyIdx</i> position within <i>DstKeyType</i> area without judging <i>SrcKeyType</i> and <i>SrcKeyIdx</i>. 2. Only when PED_TLK does not exist, plaintext is allowed to be written and key is allowed to be downloaded. 3. When <i>SrcKeyIdx</i> is valid, <i>DstKeyValue</i> is regarded as the ciphertext of the key. The system will decrypt <i>DstKeyValue</i> with <i>SrcKeyIdx</i> key in <i>SrcKeyType</i> area and write the key to <i>DstKeyIdx</i> position within <i>DstKeyType</i> area. 4. PED_TAESK is the same as PED_AES_TDK, it is used for encrypting/decrypting with AES algorithm. 		

NOTE

The input parameter must be valid; otherwise, an error may occur. Valid *KeyBlock* length must be 184 bytes.

6.8.2 OsPedAes

Prototype	intOsPedAes(intKeyIdx, unsigned char *InitVector, const unsigned char *DataIn, intDataInLen, unsigned char *DataOut, int Mode);	
Function	Use TAESK or PED_AES_TDK to do AES encryption or decryption for DataIn whose length is specified by DataInLen.	
Parameters	KeyIdx[Input]	TAESK index. The value range is from 1 to 100.
	InitVector[Input/Output]	Initialization vector. When <i>Mode</i> =0x02/0x03/0x04/0x05, initialization vector is needed in encryption/decryption. If <i>InitVector</i> is NULL, the initialization vector is set to 16-byte 0x00 by default. When <i>Mode</i> =0x00/0x01, the initialization vector is not needed and can be set to NULL. When <i>Mode</i> =0x06/0x07, the initialization vector represents a 16-byte temporary counter which is required for calculation, and the counter will be updated after calculated successfully.
	DataIn[Input]	A pointer to the data that needs to be calculated.
	DataInLen[Input]	Length of data that needs to be calculated. The valid length is not more than 1024 bytes and must be a multiple of 16. When the calculation mode is in CTR mode, there is no limit to the data length.
	DataOut[Output]	A pointer to the data that has been calculated.
	Mode [Input]	0x00: ECB Decryption

		0x01: ECB Encryption 0x02: CBC Decryption 0x03: CBC Encryption 0x04: OFB Decryption 0x05: OFB Encryption 0x06: CTR Decryption 0x07: CTR Encryption
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	Device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	Others	Refer to PED Return Code List .
Instruction		

6.9 SM Algorithm

6.9.1 OsPedGenSM2Pair

Prototype	<code>int OsPedGenSM2Pair(uchar *PvtKey, uchar *PubKey, int KeyLenBit);</code>	
Function	Generate SM2 key pair.	
Parameters	PvtKey[Output]	A pointer to SM2 private key. The valid length is 32 bytes.
	PubKey[Output]	A pointer to SM2 public key. The valid length is 64 bytes.
	KeyLenBit[Input]	Private key bit, 256 bits.
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	Device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_SYS_BAD	System error
Instruction	256 bits of SM2 private key and 512 bits of public key are valid.	

6.9.2 OsPedWriteSM2Key

Prototype	<code>int OsPedWriteSM2Key(int KeyIdx, int KeyType,</code>
------------------	--

	uchar *KeyValue);	
Function	Inject SM2 private key or public key into PED.	
Parameters	KeyIdx[Input]	SM2 key index. The valid range is from 1 to 20.
	KeyType[Input]	Key types: PED_SM2_PVT_KEY 0x30 private key PED_SM2_PUB_KEY 0x31 public key
	KeyValue[Input]	When KeyType=0x30, the valid length of KeyValue is 32 bytes. When KeyType=0x31, the valid length of KeyValue is 64 bytes.
Return	RET_OK Succeeded. ERR_DEV_NOT_OPEN Device is not open. ERR_INVALID_PARAM Invalid parameter. ERR_SYS_BAD System error ERR_PED_KEY_IDX_ERR Key index error ERR_PED_KEY_TYPE_ER R Key type error Others Refer to PED Return Code List .	
Instruction		

6.9.3 OsPedSM2Sign

Prototype	int OsPedSM2Sign(int PubKeyIdx, int PvtKeyIdx, uchar *Uid, int UidLen, uchar *Input, int InputLen, uchar *Signature);	
Function	Use SM2 algorithm to get signature information.	
Parameters	PubKeyIdx[Input]	SM2 public key index. The value range is from 1 to 20.
	PvtKeyIdx[Input]	SM2 private key index. The value range is from 1 to 20.
	Uid[Input]	User ID. The length of user ID is 16 bytes. The default value of Uid is 0x31, 0x32,

		0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, unless otherwise specified.
	UidLen[Input]	Length of user ID, the maximum length is 512 bytes.
	Input[Input]	Data that needs to be signed.
	InputLen[Input]	Data length, the maximum length is 2048 bytes.
	Signature[Input]	64-byte signature value.
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	Device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_SYS_BAD	System error.
	ERR_PED_KEY_IDX_ERR	Key index error.
	ERR_PED_KEY_TYPE_ERR	Key type error.
	ERR_PED_NO_KEY	Key doesn't exist.
	ERR_PED_TAMPERED	PED is tampered.
	Others	Refer to PED Return Code List .
Instruction		

6.9.4 OsPedSM2Verify

Prototype	<code>int OsPedSM2Verify(int PubKeyIdx, uchar *Uid, int UidLen, uchar *Input, int InputLen, const uchar *Signature);</code>	
Function	Use SM2 public key to verify signature.	
Parameters	PubKeyIdx[Input]	SM2 public key index. The value range is from 1 to 20.
	Uid[Input]	User ID. The length of user ID is 16 bytes. The default value of Uid is 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, unless otherwise specified.

	UidLen[Input]	Length of user ID, the maximum length is 512 bytes.
	Input[Input]	Data that needs to be signed.
	InputLen[Input]	Data length, the maximum length is 2048 bytes.
	Signature[Input]	64-byte signature value.
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	Device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_SYS_BAD	System error.
	ERR_VERIFY_SIGN_FAIL	Verification failed.
	ERR_PED_KEY_IDX_ERR	Key index error.
	ERR_PED_KEY_TYPE_ERR	Key type error.
	ERR_PED_NO_KEY	Key doesn't exist.
	ERR_PED_TAMPERED	PED is tampered
	Others	Refer to PED Return Code List .
Instruction		

6.9.5 OsPedSM2Recover

Prototype	<code>int OsPedSM2Recover(int KeyIdx, uchar *Input, int InputLen, uchar *Output, int *OutputLen, int Mode);</code>	
Function	Encrypt data with SM2 public key or decrypt data with private key.	
Parameters	KeyIdx[Input]	SM2 private key index. The value range is from 1 to 20.
	Input[Input]	Data that needs to be encrypted or decrypted.
	InputLen[Input]	Data length. For encryption: The maximum length is (2048-96) bytes, For decryption: The maximum length is

		2048 bytes.
	Output[Output]	Encrypted or decrypted data.
	OutputLen[Output]	Length of encrypted or decrypted data. The length of encrypted data is equal to the original data length +96 bytes. The length of decrypted data is equal to the original data length -96 bytes.
	Mode[Input]	PED_DECRYPT 0x00 : use SM2 private key to decrypt data. PED_ENCRYPT 0x01 : use SM2 public key to encrypt data.
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	Device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_SYS_BAD	System error.
	ERR_PED_KEY_IDX_ERR	Key index error.
	ERR_PED_KEY_TYPE_ERR	Key type error.
	ERR_PED_NO_KEY	Key doesn't exist.
	ERR_PED_TAMPERED	PED is tampered.
	Others	Refer to PED Return Code List .
Instruction		

6.9.6 OsPedSM3

Prototype	<code>int OsPedSM3(uchar *Input, int InputLen, uchar *Output, int Mode);</code>	
Function	Use SM3 algorithm to calculate the hash value.	
Parameters	Input[Input]	Input data
	InputLen[Input]	Length of input data
	Output[Output]	32-byte hash value
	Mode[Input]	Mode 0x00 is supported. Other values are reserved.
Return	RET_OK	Succeeded.

	ERR_DEV_NOT_OPEN	Device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_SYS_BAD	System error.
Instruction		

6.9.7 OsPedSM4

Prototype	<code>int OsPedSM4(int KeyIdx, uchar *InitVector, uchar *Input, int InputLen, uchar *Output, int Mode);</code>	
Function	Use SM4 algorithm to encrypt or decrypt data.	
Parameters	KeyIdx[Input]	PED_SM4_TDK index, the value range is from 1 to 100.
	InitVector[Input]	16-byte initialization vector. For ECB mode, it is a NULL pointer.
	Input[Input]	Data that needs to be encrypted or decrypted.
	InputLen[Input]	Data length, the data length is not more than 1024 and must be a multiple of 16.
	Output[Output]	Encrypted or decrypted data. The data length is the same as the length of input data.
	Mode[Input]	PED_SM4_ECB_DECRYPT 0x00: SM4 ECB decryption PED_SM4_ECB_ENCRYPT 0x01: SM4 ECB encryption PED_SM4_CBC_DECRYPT 0x02: SM4 CBC decryption PED_SM4_CBC_ENCRYPT 0x03: SM4 CBC encryption
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	Device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_SYS_BAD	System error.
	ERR_PED_KEY_IDX_ERR	Key index error.

	ERR_PED_KEY_TYPE_ERR	Key type error.
	ERR_PED_NO_KEY	Key doesn't exist.
	ERR_PED_TAMPERED	PED is tampered.
	ERR_PED_KEY_LEN_ERR	Key length error.
	Others	Refer to PED Return Code List .
Instruction		

6.9.8 OsPedGetMacSM

Prototype	<code>int OsPedGetMacSM(int KeyIdx, uchar *InitVector, uchar *Input, int InputLen, uchar *MacOut, int Mode);</code>	
Function	Use SM4 algorithm to calculate MAC.	
Parameters	KeyIdx[Input]	PED_SM4_TAK key index, the value range is from 1 to 100.
	InitVector[Input]	16-byte initialization vector.
	Input[Input]	Input data that needs to perform MAC calculation.
	InputLen[Input]	Length of data that needs to be calculated, the valid length is not more than 1024 and must be a multiple of 16.
	MacOut[Output]	MAC value
	Mode[Input]	0x00: Use SM4 CBC algorithm to calculate MAC value. Firstly, bitwise xor <i>InitVecotor</i> with BLOCK1, and use SM4 algorithm to encrypt the xor result with TAK. Then, bitwise xor the encrypted result with BLOCK2, and use SM4 algorithm to encrypt the xor result with TAK. MacOut is the 16-byte encryption result. 0x01: Calculate SM3 Hash of the input data with SM4-TAK key and get the 32-byte <i>MacOut</i> .
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	Device is not open.

	ERR_INVALID_PARAM	Invalid parameter.
	ERR_SYS_BAD	System error.
	ERR_PED_KEY_IDX_ERR	Key index error.
	ERR_PED_KEY_TYPE_ERR	Key type error.
	ERR_PED_NO_KEY	Key doesn't exist.
	ERR_PED_TAMPERED	PED is tampered.
	ERR_PED_KEY_LEN_ERR	Key length error.
	Others	Refer to PED Return Code List .
Instruction		

6.9.9 OsPedGetPinBlockSM4

Prototype	<code>int OsPedGetPinBlockSM4(int KeyIdx, const char *ExpPinLenIn, uchar *DataIn, uchar *PinBlockOut, int Mode, ulong TimeoutMs);</code>	
Function	Within specified TimeoutMs, scan the input PIN on the keyboard and output PIN block which is encrypted by SM4 algorithm.	
Parameters	KeyIdx[Input]	PED_SM4_TPK key index, the value range is from 1 to 100.
	ExpPinLenIn[Input]	A pointer to valid password length string, which is an enumeration set from 0 to 12. Application lists all the valid passwords and separates them with ",". For example, if 4 and 6 digits of passwords are allowed, then the string should be set to "4, 6".
	DataIn[Input]	When Mode=0x00, DataIn points to the 16-bit primary account number generated after shifting the card number.
	PinBlockOut[Output]	Encrypted PIN block The valid length is 16 bytes.
	Mode[Input]	Format of PIN block. 0x00 ISO9564 format 0
	TimeoutMs[Input]	Timeout of inputting PIN.[unit: ms] The maximum timeout is 300,000ms; the

		default timeout is 30,000 ms; 0 means using the default timeout.
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	Device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_SYS_BAD	System error.
	ERR_PED_KEY_IDX_ERR	Key index error.
	ERR_PED_KEY_TYPE_ERR	Key type error.
	ERR_PED_NO_KEY	Key doesn't exist.
	ERR_PED_TAMPERED	PED is tampered.
	ERR_PED_KEY_LEN_ERR	Key length error.
	ERR_PED_NO_PIN_INPUT	No input.
	ERR_PED_PIN_INPUT_CANCEL	Cancel input.
	ERR_PED_WAIT_INTERVAL	Interval is too short.
	Others	Refer to PED Return Code List .
Instruction	Call OsPedGetPinBlockSM4() will operate the framebuffer, which might cause the resource conflict. If there is some kind of exception on the interface of application which is running XUI, call XuiSuspend() to suspend XUI before calling OsPedGetPinBlockSM4(), after a while, call XuiResume() to resume XUI.	

6.10 DES FIRE

6.10.1 OsPedDFAuthDiver

Prototype	<code>int OsPedDFAuthDiver(int SrcKeyIdx, int KeySetVer, int DivType, uchar Mode, uchar *Uid, uchar *EncRndB, uchar *EncSessionKey);</code>
Function	Check session key B passed from DESFire card, and generate session key A. Merge A and B into a complete session key, then encrypt and output it.

Parameters	SrcKeyIdx[Input]	The index of DESFire key, it can be valued from 1 to 100..
	KeySetVer[Input]	Key version, it is used to check the DESFire version.
	DivType[Input]	<p>Key divergent mode:</p> <ul style="list-style-type: none"> ● When DivType = 0x00, it represents non-divergent, and DESFire key will be used to encrypt the session key; ● When DivType = 0x01, combine with Uid to diverge key, the divergent key will be used to encrypt the session key.
	Mode[Input]	Encryption mode of the session key: 0x02: 3DES encryption of 16 bytes
	Uid[Input]	User data, the data length is fixed to 8 bytes. And it is used to diverge the session key.
	EncRndB[Input]	<p>The session key B generated by DESFire card:</p> <ul style="list-style-type: none"> ● When the length of session key is 8 or 16 bytes, the length of EncRndB is 8 bytes; ● When the length of session key is 24 bytes, the length of EncRndB is 16 bytes.
	EncSessionKey[Output]	The encrypted (RndA + RndB').
Return	RET_OK	Succeeded
	ERR_DEV_NOT_OPEN	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter
	Others	Refer to PED Return Code List
Instruction		

6.10.2 OsPedDFAuthMerge

Prototype	<code>int OsPedDFAuthMerge(uchar *EncRndA, int DataLen);</code>
Function	Check the cipher-text session key A' passed from DESFire card.

Parameters	EncRndA[Input]	Cipher-text of A'.
	DataLen[Input]	<p>The length of <i>EncRndA</i>.</p> <ul style="list-style-type: none"> ● When the length of session key is 8 or 16 bytes, the length of <i>EncRndA</i> is 8 bytes; ● When the length of session key is 24 bytes, the length of <i>EncRndA</i> is 16 bytes.
Return	RET_OK	Succeeded
	ERR_DEV_NOT_OPE_N	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter
	Others	Refer to PED Return Code List
Instruction		

6.11 RKI

6.11.1 OsPedRkiInjectKey

Prototype	<code>int OsPedRkiInjectKey(int KeyBlkLen, uchar *KeyBlk, uchar DstKeyIdx);</code>	
Function	Inject RKI key.	
Parameters	KeyBlkLen [Input]	Length of the RKI key
	KeyBlk [Input]	RKI key data
	DstKeyIdx [Input]	Index of the destination key, currently, it is meaningless and it can be an arbitrary value.
Return	RET_OK	Succeeded
	ERR_DEV_NOT_OPE_N	PED device is not open.
	ERR_INVALID_PARAM	Invalid parameter
	Others	Refer to PED Return Code List
Instruction		

7LCD

Prolin supports Minogui, QT and other GUI support system to manage the display on LCD. Prolin provides interfaces that can directly access physical device of LCD for applications to configure light, set contrast and retrieve screen size of LCD.

Prolin's Framebuffer provides display function. Framebuffer supports XUI, a kind of GUI that is developed by PAX (please refer to "Prolin XUI Programming Guide"). The popular GUI, such as Minogui and QT, and other GUIs based on Framebuffer are supported too. Applications can use XUI graphical interfaces provided by PAX or develop GUI system.

Details about basic FrameBuffer operations are as follows:

1. Open the FrameBuffer device, the device node is "/dev/fb";
2. Get the fixed screen information through *ioctl*;
3. Get the variable screen information through *ioctl*;
4. Map device memory to the process space through *mmap*;
5. Write framebuffer.

```

int open_screen(void)
{
    char vname[128];
    int fd, nr;
    unsigned y, addr;
    struct fb_fix_screeninfo fix;
    sb = (screen_buffer*)malloc(sizeof(screen_buffer));
    if ((sb->dev_fd = open(FB_DEV_PATH, O_RDWR)) == -1) {
        perror("open");
        return -1;
    }

    int ret = ioctl(sb->dev_fd, FBIOGET_VSCREENINFO, &fb_vinfo);

    if (ret) {
        sb->width = FB_WIDTH;
        sb->height = FB_HEIGHT;
        sb->bytes_per_pixel = FB_BYTES_PER_PIXEL;
        fprintf(stderr, "in %s line %d", __FUNCTION__, __LINE__);
    } else {
        sb->width = fb_vinfo.xres;
        sb->height = fb_vinfo.yres;
        sb->bytes_per_pixel = fb_vinfo.bits_per_pixel / 8;
    }
    if (sb->bytes_per_pixel == 3)
        sb->bytes_per_pixel = 4;

    if (ioctl(sb->dev_fd, FBIOGET_FSCREENINFO, &fix) < 0) {
        close(sb->dev_fd);
        return -1;
    }

    fbmemlen = sb->width * sb->height * sb->bytes_per_pixel;

    if ((sb->buffer = (uint8_t *) mmap(NULL, fbmemlen, PROT_READ | PROT_WRITE, MAP_FILE | MAP_SHARED, sb->dev_fd, 0)) == (uint8_t *) -1)
    {
        fprintf(stderr, "rw_sd_inand.c: Can't mmap frame buffer ++\n");
        exit(1);
    }
}

```

```

    }
    memset(sb->buffer, 0, fbmemlen);
    return 0;
}

```

6. Close the FrameBuffer Device.

```

void close_screen(screen_buffer *sb)
{
    if(!sb)
        return;
    // Unmap the framebuffer
    munmap(sb->buffer, fbmemlen);
    // Close framebuffer device
    close(sb->dev_fd);
    free(sb);
}

```

7.1 OsScrContrast

Prototype	void OsScrContrast(int Contrast);	
Function	Set LCD contrast.	
Parameters	Contrast	Contrast level. The value range is [0~7]. 0: darkest 7: brightest The default value is 4. Other values: no action.
Return	None.	
Instruction	OsScrContrast() only applies to monochrome LCD.	

7.2 OsScrBrightness

Prototype	void OsScrBrightness(int Brightness);	
Function	Set screen brightness.	
Parameters	Brightness	Brightness level [0~10]. 0: turn off the backlight

		10: brightest The default value is 8. Other values: no action.
Return	None.	
Instruction	The settings will become invalid after exiting the application.	

7.3 OsScrGetSize

Prototype	<code>void OsScrGetSize(int *Width, int *Height);</code>	
Function	Get the LCD physical screen size.	
Parameters	Width[Output]	Width. (unit: pixel)
	Height[Output]]	Height. (unit: pixel)
Return	None.	
Instruction	<ol style="list-style-type: none"> The screen size is read-only. OsScrGetSize() is only applicable to applications that do not use GUI. 	

8Keyboard

Prolin's keyboard input is managed by GUI.

Definitions of key value

Macro	Value	Description
KEY1	2	<i>KEY“1”</i>
KEY2	3	<i>KEY “2”</i>
KEY3	4	<i>KEY“3”</i>
KEY4	5	<i>KEY“4”</i>
KEY5	6	<i>KEY“5”</i>
KEY6	7	<i>KEY“6”</i>
KEY7	8	<i>KEY“7”</i>
KEY8	9	<i>KEY“8”</i>
KEY9	10	<i>KEY“9”</i>
KEY0	11	<i>KEY“0”</i>
KEYCANCEL	223	<i>KEY“CANCEL”</i>
KEYCLEAR	14	<i>KEY“CLEAR”</i>
KEYENTER	28	<i>KEY“ENTER”</i>
KEYALPHA	69	<i>KEY“ALPHABET”</i>
KEYF1	59	<i>The first key from left to right</i>

		<i>at the bottom of S800 LCD.</i>
KEYF2	60	
KEYF3	61	
KEYF4	62	
KEYFUNC	102	<i>Key“Function”</i>
KEYUP	103	<i>The second key from left to right at the bottom of S800 LCD.</i>
KEYDOWN	108	<i>The third key from left to right at the bottom of S800 LCD.</i>
KEYPOWER	116	<i>Power key of D200 on the upper right corner.</i>
KEYMENU	139	<i>The fourth key from left to right at the bottom of S800 LCD.</i>
KEYCAMERA	212	<i>Independent key of Q90s Camera.</i>

The input subsystem can directly be used to test keyboard drivers or transplant other GUI systems. The device node of keyboard is "/dev/keydown".

Details of calling the input subsystem are shown as follows:

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <linux/input.h>
static int keypad_fd = -1;
struct input_event ev0[64];
//for handling/key/event
static int handle_event0() {
    int button = 0, realx = 0, realy = 0, i, rd;
    rd = read(keypad_fd, ev0, sizeof(struct input_event) * 64);
    if (rd < sizeof(struct input_event)) return 0;
    for (i = 0; i < rd / sizeof(struct input_event); i++) {
        printf("%d, %d, %d, %d, %d\n", ev0[i].type, ev0[i].code, ev0[i].value);
        if (ev0[i].type == 3 && ev0[i].code == 0)
            realx = ev0[i].value;
```

```

else if (ev0[i].type == 3 && ev0[i].code == 1)
    realy = ev0[i].value;
else if (ev0[i].type == 1) {
    if (ev0[i].code == 158) {
        //if key esc then exit
        return 0;
    }
}
else if (ev0[i].type == 0 && ev0[i].code == 0 && ev0[i].value == 0) {
    realx = 0, realy = 0;
}
printf("event(%d): type: %d; code: %3d; value: %3d; realx: %3d;
realy: %3d\n", i,
    ev0[i].type, ev0[i].code, ev0[i].value, realx, realy);
}
return 1;
}

int main(void) {
    int done = 1;
    printf("sizeof(struct input_event) = %d\n", sizeof(struct input_event));
    keypad_fd = open("/dev/keydown", O_RDWR);
    if (keypad_fd < 0)
        return -1;
    while (done) {
        printf("begin handle_event0...\n");
        done = handle_event0();
        printf("end handle_event0...\n");
    }
    if (keypad_fd > 0) {
        close(keypad_fd);
    }
    keypad_fd = -1;
}
return 0;
}

```

8.1 OsKbBacklight

Prototype	void OsKbBacklight(int OnOff);
-----------	---------------------------------------

Function	Switch on/off the keyboard backlight.	
Parameters	OnOff	0: Turn off the backlight. Non-zero: Turn on the backlight.
Return	None.	
Instruction	When the application exits, the setting will be resumed to the state before the application is started.	

9 Touch Screen

Prolin's touch screen input is managed by GUI.

Application developers can directly use the input subsystem to test touchscreen drivers or transplant other GUI systems.

About input subsystem calling, please refer to Chapter 8 Keyboard. The node of touch screen is “/dev/tp”.

10Signature Pad

For more details, please refer to the “*Prolin XUI Programming Guide*”.

11Printer

Prolin supports both physical printing and virtual printing, and provides developers with uniform printing interfaces.

Physical printer supports POSIX interface of Linux.

Virtual printer generates BMP images and saves the printing result in local disk space.

11.1 Return Code List

Table 11.1 Printer return code list

Macro	Value	Description
<i>ERR_PRN_BUSY</i>	-3701	<i>Printer is busy.</i>
<i>ERR_PRN_PAPEROUT</i>	-3702	<i>Out of paper.</i>
<i>ERR_PRN_WRONG_PACKAGE</i>	-3703	<i>Data package format error.</i>
<i>ERR_PRN_OVERHEAT</i>	-3704	<i>Printer is overheated.</i>
<i>ERR_PRN_OUTOFMEMORY</i>	-3705	<i>The print data is too large and exceeds the buffer length.</i>
<i>ERR_PRN_OVERTENSION</i>	-3706	<i>Voltage is too high.</i>

11.2 Open and Close

This part includes three functions: opening, resetting and closing printer.

11.2.1 OsPrnOpen

Prototype	<code>int OsPrnOpen(unsigned int printertype, const char*targetname);</code>													
Function	Open a printer device, including physical or virtual printer.													
Parameters	printertype[Input]	<p>Printer Types:</p> <ul style="list-style-type: none"> • PRN_REAL: Physical printer. • PRN_BMP: Virtual printer, the printing result is saved in bmp format in local position. 												
	targetname[Input]	<p>For physical printer, targetname must be NULL; For virtual printer, targetname points to the local bmp file. If the specified file has already existed, overwrite it.</p>												
Return	<table> <tr> <td>RET_OK</td> <td>Succeeded.</td> </tr> <tr> <td>ERR_DEV_NOT_EXIST</td> <td>Device does not exist.</td> </tr> <tr> <td>ERR_INVALID_PARAM</td> <td>Invalid parameter.</td> </tr> <tr> <td>ERR_DEV_BUSY</td> <td>Device is busy.</td> </tr> <tr> <td>ERR_BATTERY_ABSENT</td> <td>Battery is absent.</td> </tr> <tr> <td>ERR_BATTERY_VOLTAGE_TOO_LOW</td> <td>Battery voltage is too low.</td> </tr> </table>		RET_OK	Succeeded.	ERR_DEV_NOT_EXIST	Device does not exist.	ERR_INVALID_PARAM	Invalid parameter.	ERR_DEV_BUSY	Device is busy.	ERR_BATTERY_ABSENT	Battery is absent.	ERR_BATTERY_VOLTAGE_TOO_LOW	Battery voltage is too low.
RET_OK	Succeeded.													
ERR_DEV_NOT_EXIST	Device does not exist.													
ERR_INVALID_PARAM	Invalid parameter.													
ERR_DEV_BUSY	Device is busy.													
ERR_BATTERY_ABSENT	Battery is absent.													
ERR_BATTERY_VOLTAGE_TOO_LOW	Battery voltage is too low.													
Instruction	<ol style="list-style-type: none"> 1. OsPrnOpen() must be called successfully to open printer device; otherwise, other related functions of printer will fail to work. 2. S920 mobile terminal must be powered by battery; otherwise, printer cannot work and returns ERR_BATTERY_ABSENT. 3. When the battery level is 0, the module cannot work and returns ERR_BATTERY_VOLTAGE_TOO_LOW. 													

11.2.2 OsPrnReset

Prototype	<code>void OsPrnReset(void);</code>
Function	Reset printer parameters and clear printing buffer.
Parameters	None.

Return	None.
Instruction	

CAUTION

Font library settings will be restored to default font status (only support English).

11.2.3 OsPrnClose

Prototype	void OsPrnClose(void);
Function	Close printer.
Parameters	None.
Return	None.
Instruction	Call OsPrnClose() to close printer device when program exits.

11.3 Printer Settings

11.3.1 OsPrnSetSize

Prototype	int OsPrnSetSize(unsigned int Width, unsigned int Height);	
Function	Set parameters for virtual printer.	
Parameters	Width[Input]	Width
	Height[Input]	Height
Return	RET_OK Succeeded. ERR_INVALID_PAR Invalid parameter. AM	
Instruction	1. OsPrnSetSize() is only applicable to virtual printer. 2. The default size is 384×5000 and the maximum size is 600 ×5000.(unit: dot) 3. Only the first calling of this function is valid and the settings will remain unchanged even it is called again later.	

11.3.2 OsPrnSetDirection

Prototype	int OsPrnSetDirection(unsigned char Mode);
------------------	---

Function	Set printing direction.	
Parameters	Mode [Input]	0: print horizontally. 1: print vertically.
Return	RET_OK	Succeeded. ERR_INVALID_PARAM Invalid parameter.
Instruction	This function applies to both physical printer and virtual printer.	

11.3.3 OsPrnSetGray

Prototype	void OsPrnSetGray(int Level);	
Function	Set printing gray level.	
Parameters	Level	<ul style="list-style-type: none"> • Level =0: reserved. • Level =1: normal printing; default level. • Level =2: reserved. • Level =3: two-layer thermal printing. • Level =4: two-layer thermal printing, higher gray level than 3. • The default level is 1. • Invalid parameter won't change current settings.
Return	None.	
Instruction	This function just applies to physical printer.	

11.4 Type Setting

11.4.1 OsPrnSetSpace

Prototype	void OsPrnSetSpace(int CharSpace, int LineSpace);	
Function	Set printing space.	
Parameters	CharSpace	Character space. (unit: pixel) (It is invalid to the mandatory non-monospaced fonts, such as Arabic fonts, Thai fonts.)
	LineSpace	Line space. (unit: pixel)
Return	None.	
Instruction	1. After a successful call of this function, the settings will remain valid until OsPrnSetSpace() is called again or	

	<p>OsPrnReset() is called.</p> <ol style="list-style-type: none"> 2. The default character space is 0 pixel. 3. The default line space for thermal printer and stylus printer is 0 pixel and 2 pixels, respectively. 4. The maximum line space is 255 pixels. 5. The maximum character space is 255 pixels. 6. Invalid parameter will not change current settings.
--	--

11.4.2 OsPrnSetReversal

Prototype	int void OsPrnSetReversal(int Attr);	
Function	Set reverse attribute of font. The default setting is to print normally.	
Parameters	Attr	0: normal printing. 1: reversal printing
Return	None.	
Instruction	<ol style="list-style-type: none"> 1. This function applies to both physical printer and virtual printer. 2. The reverse attribute doesn't work for printing images. 	

11.4.3 OsPrnSetIndent

Prototype	int OsPrnSetIndent(unsigned int Left, unsigned int Right);	
Function	Set left and right margins.	
Parameters	Left[Input]	Left margin. The valid range is from 0 to 100 and default value is 0.
	Right[Input]	Right margin. The valid range is from 0 to 100 and default value is 0.
Return	RET_OK Succeeded. ERR_INVALID_PARAM Invalid parameter.	
Instruction	If physical printer is set to print vertically, left margin corresponds to the top page margin, while right margin corresponds to the bottom page margin.	

11.4.4 OsPrnCheck

Prototype	int OsPrnCheck(void);
------------------	------------------------------

Function	Check the current status of printer. It only applies to physical printer.	
Parameters	None.	
Return	RET_OK	Succeeded.
	ERR_PRN_BUSY	Printer is busy.
	ERR_PRN_PAPEROUT	Out of paper.
	ERR_PRN_OVERHEAT	Printer is overheated.
Instruction		

11.4.5 OsPrnGetDotLine

Prototype	int OsPrnGetDotLine(void);	
Function	Get the number of printed dot lines.	
Parameters	None.	
Return	>=0	The number of printed dot lines.
Instruction	Used for slip alignment (seldom used). This function applies to both physical printer and virtual printer.	

11.4.6 OsPrnSetFont

Prototype	int OsPrnSetFont(const char *fontname);	
Function	Select font.	
Parameters	fontname[Input]	Font (file) name.
Return	RET_OK	Succeeded.
	ERR_FONT_NOT_EXIST	Font does not exist.
	ERR_INVALID_PARAMETER	Invalid parameter.
Instruction	1. This function is used to choose different font styles and sizes for printing. 2. The built-in font (file) name can be obtained by calling OsEnumFont().	

11.4.7 OsPrnSelectFontSize

Prototype	void OsPrnSelectFontSize(int SingleCodeWidth,
------------------	--

	<code>int SingleCodeHeight, int MultiCodeWidth, int MultiCodeHeight);</code>		
Function	Set font size.		
Parameters	SingleCodeWidth	Width control of single code font. (For non-monospaced font, the real width of each character may not meet the setting). The value ranges from 8 to 64.	
	SingleCodeHeight	Height control of single code font. The value ranges from 8 to 64.	
	MultiCodeWidth	Width control of multiple code font. The value ranges from 12 to 64.	
	MultiCodeHeight	Height control of multiple code font. The value ranges from 12 to 64.	
Return	None.		
Instruction	<ol style="list-style-type: none"> After the first calling of OsPrnOpen(), the font width and height are both set to the default values, for single-code, it is (12×24) , for multi-code, it is (24×24). This function applies to both physical printer and virtual printer. 		



CAUTION For signle-code font, it is recommended the width should be half of the height. For multi-code font, The height and width should be the same, otherwise, font may be displayed abnormally.

11.4.8 OsPrnFeed

Prototype	<code>void OsPrnFeed(int Pixel);</code>		
Function	Feed printing paper for some number of pixels in print buffer.		
Parameters	Pixel	Number of pixels >0: feed paper. <0: unreeling. =0: no action.	
Return	None.		
Instruction	1. This function applies to both physical and virtual printer.		



CAUTION This is a one-time action which will become invalid once implemented.

11.4.9 OsPrnPrintf

Prototype	<code>void OsPrnPrintf(const char *Str, ...);</code>
Function	Format and output string to print buffer.
Parameters	Str[Input] A pointer to string that needs to be printed.
Return	None.
Instruction	<ul style="list-style-type: none"> 1. Support variable-length parameters. 2. Support '\n' (new line) and '\f' (form feed) control characters in the string. 3. If the package of printing data is too long, the program will overflow. 4. If printing string is beyond print range, it will automatically change line and continue printing. 5. The maximum buffer size is 2048 bytes. 6. This function is used to store str in printing buffer. After calling OsPrnStart(), data will be printed in the same sequence as it is written into the buffer.

11.4.10 OsPrnPutImage

Prototype	<code>void OsPrnPutImage(const unsigned char *Logo);</code>
Function	Output image to print buffer.
Parameters	Logo[Input] A pointer to logo information; The valid length is not more than 20,000 bytes.
Return	None.
Instruction	<p>Steps of generating bitmap data are as follows:</p> <ul style="list-style-type: none"> 1. Draw a logo: use paintbrush program under Windows to draw a bitmap, and save the bitmap drawn at the previous step as a "monochromatic, bmp format" file. 2. Use "Bitmap Converter" provided by PAX to convert the bmp file into a header file, for instance, Logo.h. (After conversion, the number of arrays in the head file will be same as the number of selected .bmp files. The definition of array name is related to BMP filename.)

NOTE

Printing bitmap size limit: For thermal printer, up to 384 pixels in width are allowed; for stylus printer, 180 pixels are allowed. The height is unlimited.

The generated array can be directly used as the input parameter of this function.

If the bitmap width is larger than the limit of printer, the redundant data on the right will be removed.

If the size of data packet is too large, the redundant LOGO message will be removed.

NOTE

Format description of image array in header file:

First byte [1 byte]: row numbers of the bitmap;

Byte size of the first bitmap line [2 Bytes, MSB ahead (most significant byte)];

Bitmap data of the first bitmap line;

Byte size of the second bitmap line [2 Bytes, MSB ahead];

Bitmap data of the second bitmap line;

So on and so forth.

This function only stores logo into printing buffer. After calling OsPrnStart (), the system begins to print data in printing buffer in the same sequence as they are written into the buffer.

11.4.11 OsPrnStart

Prototype	int OsPrnStart(void);	
Function	Start printing the data in the buffer.	
Parameters	None.	
Return	RET_OK ERR_PRN_BUSY ERR_PRN_PAPEROUT ERR_PRN_WRONG_PACKAGE ERR_PRN_OVERHEAT ERR_PRN_OUTOFMEMORY	Succeeded. Printer is busy. Out of paper. Data package format error. Printer is overheated. Data size is too large.

Instruction	<ol style="list-style-type: none"> 1. OsPrnStart() will begin printing task and will not return until the task is completed. 2. After completing printing task, OsPrnStart() will return the printer status in return value. Therefore, it is not necessary to check printer status. 3. After the printing task is finished, slip will be reprinted if this function is called again. 4. After setting parameter <code>persist.sys.continue.print</code> in the registry to enable the function of continuing to print after a shutdown, if event like lacking paper or the printer is overheated occurs during the printing process, after adding some more paper or waiting for the machine to cool down, call this function again will continue the printing from where it left off. But it doesn't support continue to print after the terminal has been rebooted.
-------------	--

11.4.12 OsPrnClrBuf

Prototype	void OsPrnClrBuf(void);
Function	Clear print buffer.
Parameters	None
Return	None
Instruction	

11.4.13 OsPrnSetParam

Prototype	int OsPrnSetParam(unsigned int cmd);	
Function	Set whether to pre-feed printing paper.	
Parameters	cmd [Input]	1: No pre-feeding printing paper 2: Pre-feeding printing paper
Return	RET_OK	Succeeded
	ERR_INVALID_PARAM	Invalid parameter
Instruction	<ol style="list-style-type: none"> 1. This setting only affect the pre-feeding printing paper function, and pre-feeding is enabled by default. 2. This function only applies to thermal printer. 	

11.5 POSIX

Prolin physical printer driver provides application developers with POSIX interface.

11.5.1 Open

Open physical printer.

The device name is “/dev/printer”.

```
int handle = open("/dev/printer", O_RDWR).
```

11.5.2 Read

Read printer status.

The first byte of Read buffer is *buf[0]*. The format is:

buf[0]	Description
<i>0x00</i>	<i>Normal</i>
<i>0x01</i>	<i>Printer is busy.</i>
<i>0x02</i>	<i>Out of paper.</i>
<i>0x03</i>	<i>Printer is overheated.</i>
<i>0x04~0xFF</i>	<i>Reserved.</i>

Sample code:

```
unsigned char buf[10];
int ret = read(handle, buf, 2);
if (ret > 0) {
    //buf[0]
}
```

11.5.3 Write

Configure printer and start printing buffer.

The first two bytes of buffer are gray setting and reserved bit. Suppose that the buffer is char buf[50], bit0~bit2 in the buf[0] represent the printing gray control values. bit3~bit7 are reserved.

bit0~2 in buf[0]	Description
<i>000</i>	<i>Reserved</i>

001	<i>Default gray level</i>
010	<i>Reserved</i>
011	<i>Two-layer thermal printing A</i>
100	<i>Two-layer thermal printing B</i>
101	<i>Reserved</i>
110	<i>Reserved</i>
111	<i>Reserved</i>

The second byte buf[1]: reserved.

From buf[2] on, every line is composed of 48 characters (384 dots), if less than 48 characters, the driver will be padded with 0x00.

Sample code:

```
unsigned char buf[50];

buf[0] = 0x01;
memset(buf + 2, 0xff, 48);

int ret = write(handle, buf, 50);
if (ret < 0) {
    /*Error handling.....*/
}
```

The maximum data length is described as below:

1. For 384 dots thermal printer, when printing horizontally, the driver can deal with 5000 dot lines each time at most; otherwise, the printer will not work.
2. When printing vertically, the driver can deal with 384 lines each time at most, and each line can print 5000 dots at most; otherwise, the printer will not work.
3. The maximum length of a `write` buffer should be $384 \times 5000/8 + 2 = 240,002$ bytes.

11.5.4 Close

Close file handles of printer.

```
close (handle);
```

12Font Library

Prolin uses [Freetype](#) as the system font library. Therefore, the system supports a series of vector font and bitmap font.

12.1 Data Structure

FT_FONT font structure:

```
FT_FONT
typedef struct {
    char FileName[64]; /* Font file name */
    char FontName[64]; /* Font name */
}FT_FONT;
```

FT_DOT matrix structure:

```
FT_DOT
typedef struct {
    unsigned char Left; /*Left base-line shift*/
    unsigned char Top; /* Top base-line shift*/
    unsigned char Width; /* Font width */
    unsigned char Height; /* Font height */
    unsigned char Dot[3072]; /*Valid font data */
```

```
 }FT_DOT;
```

12.2 Font Operation

12.2.1 OsEnumFont

Prototype	int OsEnumFont(FT_FONT **FontList);	
Function	Get the vector font list from FreeType library.	
Parameters	FontList[Output]	Vector font list of FT_FONT structure.
Return	>=0 ERR_INVALID_PARAMETER ERR_FONT_NOT_EXIST	The number of vector fonts. Invalid parameter. Font library does not exist.
Instruction	<p><i>Example:</i></p> <pre>int i, num; FT_FONT *FontList; num = OsEnumFont(&FontList); if(num <= 0) return -1; for(i=0; i<num; i++) printf("[%d]file name: %s, font name : %s\n", i, FontList[i].FileName, FontList[i].FontName);</pre>	

12.2.2 OsOpenFont

Prototype	int OsOpenFont(const char *FileName);	
Function	Load vector fonts.	
Parameters	FileName[Input]	Font file name.
Return	>=0 ERR_INVALID_PARAMETER ERR_FONT_NOT_EXIST	Font handle Invalid parameter. Font library does not exist.
Instruction	Dot-matrix data needs to be cached after the fonts were	

opened. It is recommended to wait 3 seconds before calling OsGetFontDot().

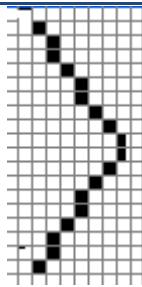
12.2.3 OsCloseFont

Prototype	void OsCloseFont(int Handle);
Function	Close vector fonts.
Parameters	Handle[Input] Font handle
Return	None.
Instruction	After using vector fonts, please close it to release the system resources.

12.2.4 OsGetFontDot

Prototype	int OsGetFontDot(int Handle, const char *Utf8Code, const int Width, const int Height, const int Style, FT_DOT *FtDot);																		
Function	Get the type matrix which conforms to UTF-8 encoding standards.																		
Parameters	Handle [Input]	Font handle																	
	Utf8Code [Input]	Character which conforms to UTF-8 encoding standards																	
	Width [Input]	Font width. The value range is from 8 to 128.																	
	Height [Input]	Font height. The value range is from 8 to 128.																	
	Style [Input]	Font styles: <table> <tr> <td>FONT_STYLE_NOR</td> <td>0</td> <td>No style</td> </tr> <tr> <td>E</td> <td></td> <td></td> </tr> <tr> <td>FONT_STYLE_BOL</td> <td>0x000000</td> <td>Bold</td> </tr> <tr> <td>D</td> <td>01</td> <td></td> </tr> <tr> <td>FONT_STYLE_ITAL</td> <td>0x000000</td> <td>italic</td> </tr> <tr> <td>IC</td> <td>02</td> <td></td> </tr> </table>	FONT_STYLE_NOR	0	No style	E			FONT_STYLE_BOL	0x000000	Bold	D	01		FONT_STYLE_ITAL	0x000000	italic	IC	02
FONT_STYLE_NOR	0	No style																	
E																			
FONT_STYLE_BOL	0x000000	Bold																	
D	01																		
FONT_STYLE_ITAL	0x000000	italic																	
IC	02																		

		“ ” can be used to combine different font styles together when several styles are needed, for example, FONT_STYLE_BOLD FONT_STYLE_ITALIC means bold and italic font.
	FtDot [Output]	Output FT DOT structure.
Return	RET_OK	Succeeded.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_FILE_NOT_EXIST	File does not exist.
	ERR_FONT_CODE	Font code error.
	ERR_INVALID_HANDLE	Invalid handle
Instruction	<p>Utf8Code input:</p> <ol style="list-style-type: none"> 1. UTF-8 code is of variable-length and shall end with “\0”; 2. When the code is composed of letters, <i>Utf8Code</i> requires two bytes in which <i>Utf8Code[0]</i> represents letter and <i>Utf8Code[1]</i> represents “\ 0”; 3. When the code is composed of Chinese, <i>Utf8Code</i> requires four bytes in which <i>Utf8Code[0-2]</i> represents Chinese and <i>Utf8Code[3]</i> represents “\ 0”. <p>The italic style dot matrix:</p> <ol style="list-style-type: none"> 1. Please pay special attention to the layout in type setting as the obtained dot matrix width may be wider than the set value when using italics effect. 2. Dot matrix size is not recommended to be less than 24; otherwise, the dot matrix may fail to be displayed in italic effect. For example, when Song typeface dot matrix is less than 19 and Black dot matrix is less than 21, italic effect cannot be applied to all the Chinese characters, but it is available for English letters. <p>Font data format:</p> <ol style="list-style-type: none"> 1. All the font dot matrixes are arranged in horizontal mode; 2. A byte contains all dots of a row. The MSB (0x80) and LSB (0x01) of the byte correspond the most left and most right dot, respectively; 3. If the character width is not an integer multiple of 8, each line of dot matrix is (width+7)/8 bytes. <p>For example: For the character “>”, with 10(width)x20(height)</p>	



The font data is:

0x00, 0x00,
0x20, 0x00,
0x10, 0x00,
0x10, 0x00,
0x08, 0x00,
0x04, 0x00,
0x04, 0x00,
0x02, 0x00,
0x01, 0x00,
0x00, 0x80,
0x00, 0x80,
0x01, 0x00,
0x02, 0x00,
0x04, 0x00,
0x04, 0x00,
0x08, 0x00,
0x10, 0x00,
0x10, 0x00,
0x20, 0x00,
0x00, 0x00

The following characters will be returned:

Width = 10

Height = 20

5
Dot data:

```
0x00,0x00,0x20,0x00,0x10,0x00,0x10,0x00,  
0x08,0x00,0x04,0x00,0x04,0x00,0x02,0x00,  
0x01,0x00,0x00,0x80,0x00,0x80,0x01,0x00,  
0x02,0x00,0x04,0x00,0x04,0x00,0x08,0x00,  
0x10,0x00,0x10,0x00,0x20,0x00,0x00,0x00
```

13Code

Prolin uses UTF-8 as the default code and provides interfaces to converse code.

13.1 Code Conversion

13.1.1 OsCodeConvert

Prototype	<code>int OsCodeConvert (const char *FromCharset, const char *ToCharset, const char *InBuf, char *OutBuf, unsigned int LenOut);</code>	
Function	Convert character encoding.	
Parameters	FromCharset[Input]	Original character encoding
	ToCharset[Input]	Target character encoding
	InBuf [Input]	Original encoding string, ending with“\0”. “Unicode” should end with“\0\0”.
	OutBuf [Output]	Target encoding string that has been converted.
	LenOut [Input]	Size of <i>OutBuf</i> array. It should be at least 1.5 times of the <i>InBuf</i>

		array.
Return	>=0	Conversion succeeded. Return the length of converted string.
Instruction	ERR_INVALID_PARAM	Invalid parameter

The system supports conversions among the following codes.
/ISO-8859-(1,2,3,4,5,6,7,8,9,10,11, 13,14,15,16)
cp(850,874,932,1250,1251,1252,1253,1254,1255,1256,1257,1258)
 GBK/GB18030(2 bytes part)
 BIG5
 SHIFT_JIS
 EUC-KR
 UNICODE
 UTF-8

Notes:

1. The above codes are only suggested to convert with UTF-8 code, conversions between other codes might fail.
2. UNICODE uses the Little-Endian mode.

14MSR

Prolin provides interfaces to read data from MSR. Additionally, the customized MSR decoding logic can be implemented by application which can access the driver of card reader and directly get the bit-stream of magnetic stripe through POSIX interface.

14.1 Return Code List

Table 14.1MSR return code list

Macro	Value	Description
ERR_MSR_FAILED	-2701	<i>Fail to operate MSR.</i>
ERR_MSR_HEADERERR	-2702	<i>Head mark is not found.</i>
ERR_MSR_ENDERR	-2703	<i>End mark is not found.</i>
ERR_MSR_LRCERR	-2704	<i>LRC check error.</i>
ERR_MSR_PARERR	-2705	<i>Check error of a certain bit of MSR.</i>
ERR_MSR_NOT_SWIPED	-2706	<i>No card is swiped.</i>
ERR_MSR_NO_DATA	-2707	<i>No data in Mag card.</i>
ERR_MSR_END_ZEROERR	-2708	<i>Magnetic stripe card format error.</i>
ERR_MSR_PED_DECRYPT	-2709	<i>PED decryption failed.</i>
ERR_MSR_NO_TRACK_ER	-2710	<i>The corresponding magnetic track in</i>

R

the magnetic card has not been detected.

14.2 Data Structure

MSR data structure: Record information and status of each magnetic track.

ST_MSR_DATA:

```
typedef struct {
    unsigned char TrackData[256]; /* Decoded bit stream*/
    int DataLen; /* Track data length*/
    int Status; /*Track data status, 0 means succeeded, other value means failed*/
}ST_MSR_DATA;
```



When track data status is 0, it means reading track data successfully, pay attention to the following two situations:

1. If data format is correct or there is no data in track, judge according to the valid data length *DataLen* ;
2. If OsMsrRead() isn't called to read track data within 30 seconds after swiping card successfully, the data will be cleared automatically.

14.3 MSR Control Interface

14.3.1 OsMsrOpen

Prototype	int OsMsrOpen(void);	
Function	Open magnetic stripe reader.	
Parameters	None.	
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_EXIST	Device does not exist.
	ERR_DEV_BUSY	Device is busy.
	ERR_DEV_NOT_OPEN	Fail to open device.
Instruction	OsMsrOpen() shall be called to open the device; otherwise, the	

related functions will not work.



NOTE Magnetic stripe reader works in interrupt mode. Once the magnetic stripe reader is opened, it can read the magnetic track data as long as card is swiped even if no function is called to read card. So it is better to close magnetic stripe reader when it is not in use.

14.3.2 OsMsrClose

Prototype	void OsMsrClose(void);
Function	Close magnetic stripe reader.
Parameters	None.
Return	None.
Instruction	OsMsrClose() must be called to close device when program exits.

14.3.3 OsMsrReset

Prototype	void OsMsrReset(void);
Function	Soft reset magnetic stripe reader and clear the obtained magnetic card data.
Parameters	None.
Return	None.
Instruction	

14.3.4 OsMsrSwiped

Prototype	int OsMsrSwiped(void);
Function	Detect a swiping action.
Parameters	None.
Return	TRUE Swiped. FALSE No swiping. ERR_DEV_NOT_OPE N Device is not open.
Instruction	1. OsMsrSwiped() will return immediately no matter whether

	<p>there is a card-swiping action or not.</p> <p>2. OsMsrOpen(), OsMsrRead() or OsMsrReset() can clear the swiped status of magnetic card.</p>
--	--

14.3.5 OsMsrRead

Prototype	<code>int OsMsrRead(ST_MSR_DATA *Track1, ST_MSR_DATA *Track2, ST_MSR_DATA *Track3);</code>
Function	Read MSR data.
Parameters	Track1[Output] Output Track1 data
	Track2[Output] Output Track2 data
	Track3[Output] Output Track3 data
Return	<p>RET_OK Succeeded.</p> <p>ERR_MSR_NOT_SWIP ED No card is swiped.</p> <p>ERR_INVALID_PARAM Invalid parameter.</p> <p>ERR_DEV_NOT_OPEN Device is not open.</p>
Instruction	<p>1. If any track's data is not needed, set the corresponding pointer to NULL, then the data will not be outputted.</p> <p>2. All track data must be read out within 30 seconds after swiping card successfully; otherwise, all track data will be cleared automatically and output data are all 0x00.</p>

14.3.6 OsMsrReadJIS

Prototype	<code>int OsMsrReadJIS(ST_MSR_DATA *Track1, ST_MSR_DATA *Track2, ST_MSR_DATA *Track3, ST_MSR_DATA *Track4);</code>
Function	Read the MSR data of general single side magnetic stripe card or double side magnetic stripe card decoded with JIS.
Parameters	Track1[Output] Output Track1 data
	Track2[Output] Output Track2 data
	Track3[Output] Output Track3 data
	Track4[Output] Output Track4 data
Return	<p>RET_OK Succeeded.</p> <p>ERR_MSR_NOT_SWIP No card is swiped.</p>

	<p>ED</p> <p>ERR_INVALID_PARAM Invalid parameter.</p> <p>ERR_DEV_NOT_OPEN Device is not open.</p>
Instruction	<ol style="list-style-type: none"> 1. If any track's data is not needed, set the corresponding pointer to NULL, then the data will not be outputted. 2. All track data must be read out within 30 seconds after swiping card successfully; otherwise, all track data will be cleared automatically and output data are all 0x00. 3. Track 4 is used to store the JCB (JIS II) data, and track 1-3 is used to store the data in other formats. 4. If the magnetic head of one side has not detected any tracks, ERR_MSR_NO_TRACK_ERR will be returned to the <i>Status</i> of the corresponding <i>Track</i>.



CAUTION Call OsMsrSwiped() first to make sure there is a card swiping action before calling OsMsrRead() to obtain the data of magnetic track. Otherwise, when the function returns, the data in the buffer is invalid.



NOTE When reading magnetic card conforming to ISO7812 standard:
The length of Track1 is 79 bytes.
The length of Track2 is 37 bytes.
The length of Track3 is 107 bytes.

15IC Card Reader

In this chapter, all protocol interfaces for IC card are based on ISO7816/EMV.

15.1 Return Code List

Table 15.1 IC Card reader return code list

Macro	Value	Description
<i>ERR_SCI_HW_NOCARD</i>	-2800	<i>No card.</i>
<i>ERR_SCI_HW_STEP</i>	-2801	<i>No initialization when exchanging data; no power when warm reseting.</i>
<i>ERR_SCI_HW_PARITY</i>	-2802	<i>Parity check error.</i>
<i>ERR_SCI_HW_TIMEOUT</i>	-2803	<i>Timeout.</i>
<i>ERR_SCI_TCK</i>	-2804	<i>TCK error.</i>
<i>ERR_SCI_ATR_TS</i>	-2810	<i>ATR TS error.</i>
<i>ERR_SCI_ATR_TA1</i>	-2811	<i>ATR TA1 error.</i>
<i>ERR_SCI_ATR_TD1</i>	-2812	<i>ATR TD1 error.</i>
<i>ERR_SCI_ATR_TA2</i>	-2813	<i>ATR TA2 error.</i>
<i>ERR_SCI_ATR_TB1</i>	-2814	<i>ATR TB1 error.</i>

ERR_SCI_ATR_TB2	-2815	ATR TB2 error.
ERR_SCI_ATR_TC2	-2816	ATR TC2 error.
ERR_SCI_ATR_TD2	-2817	ATR TD2 error.
ERR_SCI_ATR_TA3	-2818	ATR TA3 error.
ERR_SCI_ATR_TB3	-2819	ATR TB3 error.
ERR_SCI_ATR_TC3	-2820	ATR TC3 error.
ERR_SCI_T_ORDER	-2821	Protocol is not T0 or T1.
ERR_SCI_PPS_PPSS	-2830	PPSS error in PPS.
ERR_SCI_PPS_PPS0	-2831	PPS0 error in PPS.
ERR_SCI_PPS_PCK	-2832	ATRPCK error in PPS.
ERR_SCI_T0_PARAM	-2840	Transmitted data is too long in T0.
ERR_SCI_T0_REPEAT	-2841	Too many character repetitions in T0.
ERR_SCI_T0_PROB	-2842	Procedure byte error in T0.
ERR_SCI_T1_PARAM	-2850	Transmitted data is too long in T1.
ERR_SCI_T1_BWT	-2851	BWT is oversize in T1.
ERR_SCI_T1_CWT	-2852	CWT is oversize in T1.
ERR_SCI_T1_BREP	-2853	Too many block repetitions in T1.
ERR_SCI_T1_LRC	-2854	LRC error in T1.
ERR_SCI_T1_NAD	-2855	NAD error in T1.
ERR_SCI_T1_LEN	-2856	LEN error in T1.
ERR_SCI_T1_PCB	-2857	PCB error in T1.
ERR_SCI_T1_SRC	-2858	SRC error in T1.
ERR_SCI_T1_SRL	-2859	SRL error in T1.
ERR_SCI_T1_SRA	-2860	SRA error in T1.
ERR_SCI_PARAM	-2880	Invalid parameter.

15.2 Data Structure

15.2.1 APDU Request Structure

ST_APDU_REQ

```
typedef struct
{
    Unsigned char Cmd[4]; /*CLA, INS, P1, P2*/
    int LC;                /* The valid length of DataIn sent to IC card */
    unsigned char DataIn[512];/* The data sent to ICC */
    int LE;                /* The expected length of returned data*/
}ST_APDU_REQ;
```

ST_APDU_REQ structure:

1. LE is the expected length of returned data. The actual length is related to specific command. LE is only an expected length and the actual length can be obtained by LenOut in ST_APDU_RSP response structure.
2. LE and LC can be used in combination:

NOTE



- LC=0, LE=0. No data is sent or returned.
- LC=0, LE>0. No data is sent, but returned expected data. If the expected data length is unknown, set LE to 256; otherwise, it will be a constant.
- LC>0, LE=0. Send data, but no expected data is returned.
- LC>0, LE>0. Send data and returned expected data. If expected data length is unknown, set LE to 256; otherwise, it will be a constant.

15.2.2 APDU Response Structure

ST_APDU_RSP:

```
typedef struct
{
    Int LenOut;           /* Data length returned from ICC*/
    unsigned char DataOut[512]; /*Data pointer returned from ICC */
    unsigned char SWA;      /*status word 1 of ICC */
```

```
unsigned char SWB;           /* status word 2 of ICC */
}ST_APDU_RSP;
```

15.3 Encapsulated Interfaces

15.3.1 OslccOpen

Prototype	int OslccOpen(int Slot);		
Function	Open IC card reader.		
Parameters	Slot	IC card channel number:	
		<ul style="list-style-type: none"> ▪ ICC_USER_SLOT User card ▪ ICC_SAM1_SLOT SAM card slot 1 ▪ ICC_SAM2_SLOT SAM card slot 2 ▪ ICC_SAM3_SLOT SAM card slot 3 ▪ ICC_SAM4_SLOT SAM card slot 4 	
Return	RET_OK	Succeeded.	
	ERR_DEV_NOT_EXIST	Device does not exist.	
	ERR_DEV_BUSY	Device is busy.	
Instruction			

- CAUTION**
- 
1. Other functions can be used only after successfully opening IC card device.
 2. Slot number and type may be different for different models, please refer to manual or consult professional staff for specific slot number.

15.3.2 OslccDetect

Prototype	int OslccDetect(int Slot);
-----------	-----------------------------------

Function	Test whether there is a card in specified slot.		
Parameters	Slot	IC card channel number:	
		<ul style="list-style-type: none"> ▪ ICC_USER_SLOT User card ▪ ICC_SAM1_SLOT SAM card slot 1 ▪ ICC_SAM2_SLOT SAM card slot 2 ▪ ICC_SAM3_SLOT SAM card slot 3 ▪ ICC_SAM4_SLOT SAM card slot 4 	
Return	RET_OK Others	Card is inserted. Please refer to IC Card Return Code List .	
Instruction	<ol style="list-style-type: none"> 1. OslccDetect () will return immediately no matter whether there is a card in slot or not. 2. For USER_SLOT, if card is inserted or pulled out, system will send SIG/CC message to application to open the device. This mechanism doesn't apply to SAM card. 		

CAUTION

This interface will power off the SAM card. Please ensure this interface is called before resetting SAM card.

15.3.3 OslccInit

Prototype	int OslccInit(int Slot, unsigned long Option, unsigned char *Atr);		
Function	Initialize IC card device.		
Parameters	Slot	IC card channel number:	
		<ul style="list-style-type: none"> ▪ ICC_USER_SLOT User card ▪ ICC_SAM1_SLOT SAM card slot 1 ▪ ICC_SAM2_SLOT SAM card slot 2 ▪ ICC_SAM3_SLOT SAM 	

		<p>card slot 3</p> <ul style="list-style-type: none"> ▪ ICC_SAM4_SLOT SAM <p>card slot 4.</p>
	Option	<p>(Bit 0~1) Card voltage options: 00 - 5V, 01 - 1.8V, 10 - 3V</p> <p>(Bit 2)Support PPS protocol: 0 – Unsupported; 1 – Supported.</p> <p>(Bit 3~4)Rate used in power-onreset: 00 – Standard rate 9600; 10 – Four times rate 38400.</p> <p>The rate mentioned here is a reference value in typical frequency (3.57MHz).</p> <p>The communication rate between IC card and card reader is closely related to working clock frequency provided by a specific machine.</p> <p>(Bit 5) Supported standards: 0 – EMV; 1 - ISO7816.</p> <p>If EMV mode is specified, the power on rate will be marked as invalid and it will use the standard rate by default.</p> <p>(Bit 6 ~31)Reserved “Option” is set to 0 by default (that is, 5V, non-PPS, standard rate, and conforming to EMVx).</p>
	Atr [Output]	<ol style="list-style-type: none"> 1. Answer To Reset (ATR). The card will return response data of 34 bytes at most. 2. Contents: Length of ATR (1 byte) and ATR.
Return	RET_OK	Succeeded.
	Others	Please refer to IC Card Return Code List .
Instruction	<ol style="list-style-type: none"> 1. ATR output buffer should be allocated at least 34 bytes. 2. Whether PPS communication parameters negotiation protocol is supported or not depends on the specific card. 3. Some terminals can only work with one SAM card at a time. If several cards need to be operated at the same time, 	

initialize cards one by one: (OslccInit ()) -> operation (OslccExchange()) -> close (OslccClose()).

CAUTION

Most SAM cards only support ISO7816, so the “Option” should be set to 0x20 instead of 0x00.

15.3.4 OslccExchange

Prototype	<pre>int OslccExchange(int Slot, int CtrlFlag, constST_APDU_REQ *ApduReq, ST_APDU_RSP *ApduRsp);</pre>	
Function	Interact with IC card using commands.	
Parameters	Slot	IC card channel number: <ul style="list-style-type: none"> ▪ ICC_USER_SLOT User card ▪ ICC_SAM1_SLOT SAM card slot 1 ▪ ICC_SAM2_SLOT SAM card slot 2 ▪ ICC_SAM3_SLOT SAM card slot 3 ▪ ICC_SAM4_SLOT SAM card slot 4
	CtrlFlag	<ol style="list-style-type: none"> 1. Bit0: Whether to send "GET RESPONSE" command automatically under T=0 protocol. 1 Yes 0 No 2. Bit1~Bit31: Reserved
	ApduReq [Input]	A pointer to APDU request structure ST_APDU_REQ : terminal request message.
	ApduRsp [Output]	A pointer to APDU response structure ST_APDU_RSP :card response message.
Return	RET_OK	Succeeded.
	Others	Please refer to IC Card Return Code

	List.
Instruction	

15.3.5 OslccClose

Prototype	int OslccClose(int Slot);	
Function	Close IC card device.	
Parameters	Slot	IC card channel number: <ul style="list-style-type: none"> ▪ ICC_USER_SLOT User card ▪ ICC_SAM1_SLOT SAM card slot 1 ▪ ICC_SAM2_SLOT SAM card slot 2 ▪ ICC_SAM3_SLOT SAM card slot 3 ▪ ICC_SAM4_SLOT SAM card slot 4
Return	RET_OK	Succeeded.
	Others	Please refer to IC Card Return Code List.
Instruction		

16RF Reader

This chapter mainly describes application programming interface of RF reader, which is conforming to the ISO14443 and “*EMV Contactless Book D V2.1*” regulation.

16.1 Return Code List

Table 16.1 Return code list

Macro	Value	Description
<code>PCD_ERR_PAR_FLAG</code>	-2901	Parity error.
<code>PCD_ERR_CRC_FLAG</code>	-2902	CRC error.
<code>PCD_ERR_WTO_FLAG</code>	-2903	Timeout or no card.
<code>PCD_ERR_COLL_FLAG</code>	-2904	Multiple cards collision.
<code>PCD_ERR_ECD_FLAG</code>	-2905	Frame format error.
<code>PCD_ERR_EMD_FLAG</code>	-2906	Interference.
<code>PCD_ERR_COM_FLAG</code>	-2907	Chip error, fail to communicate correctly.
<code>PCD_ERR_AUT_FLAG</code>	-2908	M1 authentication error.
<code>PCD_ERR_TRANSMIT_FLAG</code>	-2909	Transmission error.
<code>PCD_ERR_PROTOCOL_FLAG</code>	-2910	Protocol error.
<code>PCD_ERR_PARAMFILE_FLAG</code>	-2911	Configuration file does

		<i>not exist.</i>
PCD_ERR_USER_CANCEL	-2912	<i>Transaction is cancelled.</i>
PCD_ERR_CARRIER_OBTAIN_FLAG	-2913	<i>No obtained carrier</i>
PCD_ERR_CONFIG_FLAG	-2914	<i>Fail to configure register.</i>
PCD_ERR_RXLEN_EXCEED_FLAG	-2915	<i>The returned data length exceeds the set receiving length</i>
PCD_ERR_NOT_ALLOWED_FLAG	-2951	<i>Parameter error or invalid value.</i>
PCD_CHIP_ABNORMAL	-2952	<i>Chip is abnormal or does not exist.</i>
PCD_CHIP_NOT_OPENED	-2953	<i>Module is not open.</i>
PCD_CHIP_CARDEXIST	-2954	<i>Card is not removed.</i>
PCD_ERR_NOT_IDLE_FLAG	-2955	<i>Card is not in idle state.</i>
PCD_ERR_NOT_POLLING_FLAG	-2956	<i>No polling</i>
PCD_ERR_NOT_WAKEUP_FLAG	-2957	<i>Card does not wakeup.</i>
PCD_ERR_NOT_ACTIVE_FLAG	-2958	<i>Card is not activated.</i>
PCD_ERR_NOT_SUPPORT	-2959	<i>Chip is unsupported.</i>
ERR_BATTERY_VOLTAGE_TO_O_LOW	-1024	<i>Battery voltage is too low.</i>

16.2 Data Structure

For more information about the request data structure and response data structure, please refer to Chapter IC Card Reader [15.2.1](#) and [15.2.2](#).

16.2.1 User Configuration Structure

PCD_USER_ST

```
typedef struct pcd_user_t{
    unsigned char wait_retry_limit_w; /* S(WTX) responds to write permission of sending times*/
    unsigned int wait_retry_limit_val; /*S(WTX)responds to the maximum repetition times.*/
    unsigned char check_cancel_key_w; /*respond to write permission of the cancel key*/
}
```

```

unsigned char check_cancel_key_val; /* 0: No response to the cancel
key; 1: Respond to the cancel key
*/
int (*check_cancel_key_function)(void);/*Check whether the cancel key is
pressed. If set
check_cancel_key_w=1 and
check_cancel_key_val=1,
check_cancel_key_function () will
be called during RF card
transaction process. If
check_cancel_key_function ()
returns 0, it means the cancel key
is not pressed. If the function
returns non-zero, it means the
cancel key is pressed and will exit
transaction by force,*/
unsigned char os_picc_transfer_set_w; /*1 means
“os_picc_transfer_set_val”
value is valid, 0 means
“os_picc_transfer_set_val”
value is invalid*/
unsigned char os_picc_transfer_set_val; /* Set OsPiccTransfer
receive/send: Bit0=0: send
disable CRC; Bit0=1: send
enable CRC; Bit1=0: receive
disable CRC; Bit1=1: receive
enable CRC*/
unsigned char anti_interference_flag; /*Anti-interference setting when
searching the card; 1- enable
anti-interference, 0- disable
an-interference.*/
unsigned char protocol_layer_fwt_set_w; /*1 indicates that the value of
protocol_layer_fwt_set_val is
valid, 0 indicates that the
value of
protocol_layer_fwt_set_val is
invalid */
unsigned int protocol_layer_fwt_set_val; /* Set the frame wait time for
the protocol layer (value of
FWT)*/
unsigned char os_picc_transfer_rxlen_set_w; /*1 indicates that the
value of
os_picc_transfer_rxlen_set_v
al is valid, 0 indicates that the
value of
os_picc_transfer_rxlen_set_v
al is invalid */

```

```

unsigned int os_picc_transfer_rxlen_set_val; /*Set the maximum
                                             allowable data length in the
                                             half-duplex          block
                                             (OsPiccTransfer) transfer*/
unsigned char os_picc_transfer_direct_transmit_set_w; /* 1 means the
                                                       data is transmitted in the way
                                                       of data stream, that is, the
                                                       half-duplex          block
                                                       transmission protocol is not
                                                       applicable;0 represents data
                                                       transfer using the half duplex
                                                       block transfer protocol */
unsigned char reserved[36]; /* Reserved byte, for future use.
                           sizeof(PCD_USER_ST)= 76*/
} PCD_USER_ST;

```

16.3 Encapsulate Interfaces

16.3.1 OsPiccOpen

Prototype	int OsPiccOpen(void);								
Function	Power on PCD module and make it prepared for work.								
Parameters	None.								
Return	<table> <tr> <td>0</td> <td>Succeeded.</td> </tr> <tr> <td>Others</td> <td>Fail to open device. (Details refer to Return Code List.)</td> </tr> <tr> <td>ERR_BATTERY_VOLTAGE_TOO_LOW</td> <td>Battery voltage is too low.</td> </tr> <tr> <td>ERR_BATTERY_ABSENT</td> <td>Battery is absent.</td> </tr> </table>	0	Succeeded.	Others	Fail to open device. (Details refer to Return Code List .)	ERR_BATTERY_VOLTAGE_TOO_LOW	Battery voltage is too low.	ERR_BATTERY_ABSENT	Battery is absent.
0	Succeeded.								
Others	Fail to open device. (Details refer to Return Code List .)								
ERR_BATTERY_VOLTAGE_TOO_LOW	Battery voltage is too low.								
ERR_BATTERY_ABSENT	Battery is absent.								
Instruction	<ol style="list-style-type: none"> D200 and S920 POS models must be powered by battery to operate RF module; otherwise, ERR_BATTERY_ABSENT will be returned. When the battery voltage is 0, ERR_BATTERY_VOLTAGE_TOO_LOW is returned. 								

16.3.2 OsPiccClose

Prototype	int OsPiccClose(void);
Function	Power off PCD module.

Parameters	None.	
Return	0 Others	Succeeded. Failed.(Details refer to Return Code List .)
Instruction		

16.3.3 OsPiccResetCarrier

Prototype	int OsPiccResetCarrier(void);	
Function	Reset the carrier wave.	
Parameters	None.	
Return	0 Others	Succeeded. Failed. (Details refer to Return Code List .)
Instruction	This function implements the carrier reset for RF reader and the statue of card in the RF field will be changed to idle status.	

16.3.4 OsPiccPoll

Prototype	int OsPiccPoll(char *pcPiccType, unsigned char *pucATQx);	
Function	Poll cards, only support the roll polling of A card and B card.	
Parameters	pcPiccType[Output]	Card type: ▪ “A” - card A ▪ “B” - card B
	pucATQx [Output]	Response data: The valid length of A card is 2 bytes. The valid length of B card is 12 bytes.
Return	0 Others	Succeeded. Failed. (Details refer to Return Code List .)
Instruction	1. Mifare card is a special kind of A card. 2. The returned card type of M card is type A.	

16.3.5 OsPiccAntiSel

Prototype	int OsPiccAntiSel(const char pcPiccType, unsigned char *pucUID,
------------------	---

	<code>const unsigned char ucATQ0, unsigned char *pucSAK);</code>	
Function	Do anti-collision and selection operations for cards.	
Parameters	pcPiccType[Input]	Card type: <ul style="list-style-type: none">▪ “A” - card A▪ “B” - card B
	pucUID[Output]	Unique identifier of card: <ul style="list-style-type: none">▪ Card A-- 4, 7 or 10 bytes.▪ Card B—4 bytes.
	ucATQ0[Input]	The parameter is unused.
	pucSAK[Output]	Response data to card selection. SAK is the response data to the last selected command of card A. This parameter is ignored by card B. The valid length is 1 byte.
Return	0	Succeeded.
	Others	Failed. (Details refer to Return Code List .)
Instruction	The value of <i>pucSAK</i> can be used to differentiate card A and Mifare card.	

16.3.6 OsPiccActive

Prototype	<code>int OsPiccActive(const char pcPiccType, unsigned char *pucRATS);</code>	
Function	Activate card.	
Parameters	pcPiccType[Input]	Card type: <ul style="list-style-type: none">▪ “A”– A card▪ “B”– B card
	pucRATS[Output]	Response data to card activation: <ul style="list-style-type: none">▪ For A card, pucRATS is the response data to ATS command.▪ For B card, PucRATS is the response data to ATTRIB command.
Return	0	Succeeded.
	Others	Fail to activate card. (Details refer to Return Code List .)
Instruction		



NOTE Data outputted by pucRATS, mainly includes card frame waiting time, buffer size, transmission rate etc. For details, see the section 5.7 and 6.4 of “*EMV Contactless Book D V2.1*”.

16.3.7 OsPiccTransfer

Prototype	<code>int OsPiccTransfer(const unsigned char*pucTxBuff, int iTxLen, unsigned char* pucRxBuff, int *piRxLen);</code>	
Function	Implement transparent transmission/reception function in accordance with the half-duplex communication protocol of ISO14443-4.	
Parameters	pucTxBuff[Input]	Transmision data buffer.
	iTxLen [Input]	Transmission data length. (unit: byte)
	pucRxBuff[Output]	Response data buffer.
	piRxLen [Output]	Response data length.
Return	0	Succeeded.
	Others	Failed. (Refer to Return Code List .)
Instruction		

16.3.8 OsPiccRemove

Prototype	<code>int OsPiccRemove(void);</code>	
Function	Remove card in accordance with EMV mode.	
Parameters	None.	
Return	0	Succeeded.
	Others	Failed. (Refer to Return Code List .)
Instruction		

16.3.9 OsMifareAuthority

Prototype	<code>int OsMifareAuthority(unsigned char *uid, unsigned char blk_no, unsigned chargroup, unsigned char *psw);</code>
------------------	---

Function	VerifyMifare card.	
Parameters	uid [Input]	Card ID. The valid length is 4 bytes.
	blk_no[Input]	Block number.
	group [Input]	Password types: "A"/"B".
	psw[Input]	Authentication password. The valid length is 6 bytes.
Return	0	Succeeded.
	Others	Failed. (Refer to Return Code List .)
Instruction		

16.3.10 OsMifareOperate

Prototype	<code>int OsMifareOperate(unsigned char ucOpCode, unsigned char ucSrcBlkNo, unsigned char* pucVal, unsigned char ucDesBlkNo);</code>	
Function	Read or write a specific block of Mifare card; Increase, decrease or backup the data in specified block of Mifare card, and update it into another specified data block.	
Parameters	ucOpCode [Input]	<ul style="list-style-type: none"> ▪ "r"or"R":read; ▪ "w"or"W": write; ▪ "+": increase; ▪ "-": decrease; ▪ ">": transfer/backup.
	ucSrcBlkNo[Input]	Specify the block number that will be accessed.
	pucVal[Input/Output]	<ul style="list-style-type: none"> ▪ For reading operation, pucVal outputs block contents. The buffer size of pucVal is 16 bytes. ▪ For writing operation, pucVal inputs block contents. The buffer size of pucVal is 16 bytes. ▪ For "+"or "-"operation, pucVal is buffer head address. The buffer size of pucVal is 4 bytes. ▪ For transfer operation, pucVal has no practical meaning, but the incoming pointer cannot be NULL.

	ucUpdateBlkNo [Input]	Specify the block number which is used to write in the operation result. (When reading or writing block, <i>ucUpdateBlkNo</i> is a NULL pointer.)
Return	0 Others	Succeeded. Failed. (Refer to Return Code List .)
Instruction		

16.3.11 OsPiclInitFelica

Prototype	int OsPiclInitFelica(unsigned char ucSpeed, unsigned char ucModInvert);	
Function	Initialize the configuration of Felica card.	
Parameters	ucSpeed[Input]	Set transmission rate of interaction with card: ▪ 1: 424Kbp ▪ Others: 212Kbps
	ucModInvert[Input]	Set modulation mode of Felica. ▪ 1: positive modulation output; ▪ Others: reverse modulation output.
Return	0	Succeeded.
Instruction		

16.3.12 OsPiclIsoCommand

Prototype	int OsPiclIsoCommand(int cid, ST_APDU_REQ *ApduReq, ST_APDU_RSP *ApduRsp);	
Function	Send APDU data to card and receive response in specified channel.	
Parameters	cid [Input]	Specify logical channel number of card. The valid value of cid ranges from 0 to 14, the value is set to 0 currently.
	ApduReq [Input]	Command data structure ST_APDU_REQ sent to PICC card.
	ApduRsp [Output]	Response data structure ST_APDU_RSP returned from PICC card.
Return	0 Others	Succeeded. Failed. (Refer to Return Code List .)

Instruction	
-------------	--

16.3.13 OsPiccSetUserConfig

Prototype	<code>int OsPiccSetUserConfig(PCD_USER_ST *pcd_user_config);</code>	
Function	Set user configuration.	
Parameters	pcd_user_config [Input]	A pointer to user configuration structure PCD_USER_ST .
Return	0	Succeeded. Others Failed. (Refer to Return Code List .)
Instruction		

16.3.14 OsPiccGetUserConfig

Prototype	<code>int OsPiccGetUserConfig(PCD_USER_ST *pcd_user_config);</code>	
Function	Get user configuration.	
Parameters	pcd_user_config [Output]	A pointer to user configuration structure PCD_USER_ST .
Return	0	Succeeded. Others Failed. (Refer to the Return Code List .)
Instruction		

16.3.15 OsPiccApplePoll

Prototype	<code>int OsPiccApplePoll(char *pcPiccType, unsigned char *pucATQx, unsigned char *pucSendData);</code>	
Function	Detect cards, including the roll polling of type A card, type B card and type V card.	
Parameters	pcPiccType[Output]	Card types: <ul style="list-style-type: none">▪ A – type A card▪ B – type B card▪ V – type V card
	pucATQx [Output]	Response data of the detected card: <ul style="list-style-type: none">▪ The response data length of type A card is 2 bytes.▪ The response data length of type B

		<p>card is 12 bytes.</p> <ul style="list-style-type: none"> ▪ The response data length of type V card is 2 bytes.
	pucSendData[Input]	<p>When the first byte is 0x01, enter a fixed four-byte length:</p> <ul style="list-style-type: none"> ▪ "\x01\x00\x00\x00" represents Terminal in VAS App OR Payment Mode. ▪ "\x01\x00\x00\x01" represents Terminal in VAS App AND Payment Mode. ▪ "\x01\x00\x00\x02", represents Terminal in VAS App Mode Only. ▪ "\x01\x00\x00\x03", represents Terminal in Payment Mode Only. <p>When the first byte is 0x02, enter a variable-length byte, the lower nibble of the second byte X represents the length of Terminal Data:</p> <ul style="list-style-type: none"> ▪ "\x02\x8X\x00\x00"+X-byte Terminal Data.
Return	0	Succeeded.
	Others	Failed. (Refer to the Return Code List .)
Instruction	<ol style="list-style-type: none"> 1. Mifare card is a special kind of A card. 2. The returned card type of M card is type A. 	

16.3.16 OsPiccOffCarrier

Prototype	int OsPiccOffCarrier(void);	
Function	Close the carrier.	
Parameters	None	
Return	0	Succeeded.
	Others	Failed. (Refer to the Return Code List .)
Instruction	Close the carrier for the Non-contact IC card reader, and then all cards in RF field will be in power-off state.	

16.3.17 OsPiccInitIso15693

Prototype	int OsPiccInitIso15693(unsigned char ucDataCodeMode);
------------------	--

Function	Initialize the ISO15693 card.	
Parameters	ucDataCodeMode[1] nput]	The transmission rate that used to interact with the card: <ul style="list-style-type: none"> ● 0: 26.48 kbits/s ● Other values: reserved
Return	0	Succeeded. Others Failed. (Refer to the Return Code List .)
Instruction	<pre> int iRet = 0; unsigned char aucDataSend[256], aucDataResp[256]; int iSendLen = 0, iRespLength = 0; memset(aucDataSend, 0, sizeof(aucDataSend)); memset(aucDataResp, 0, sizeof(aucDataResp)); memcpy(aucDataSend, "\x26\x01\x00", 3); iSendLen = 3; iRet = OsPiccOpen(); iRet = OsPiccInitIso15693(0); iRet = OsPiccTransfer(aucDataSend, iSendLen, aucDataResp, &iRespLength); OsPiccClose(); </pre>	

16.4 Notes of Touch Screen and RF Reader Programming

As model S900 and S920 are equipped with both touch screen and RF reader, application developers should pay special attention to the following notes:

1. Touch screen cannot be used in the whole process A/B transaction.
2. After finishing the transaction, [OsPiccRemove\(\)](#) must be called to remove the card.
3. For Mifare card, [OsPiccRemove\(\)](#) or [OsPiccClose\(\)](#) must be called at last.
4. For FeliCa card, OsPiccClose() must be called at last.

17 Communication Port

17.1 Data Definition

Table 17.1 Macro definition list of communication ports

Macro	Value	Description
PORT_COM1	0	<i>Serial port 1UART.</i>
PORT_COM2	1	<i>Serial port2.</i>
PORT_COM3	2	<i>Serial port 3.</i>
PORT_PINPAD	3	<i>Built-out PinPad.</i>
PORT_USBDEV	11	<i>USB device mode port.(Host port 0)</i>
PORT_USBHOST	12	<i>USB host mode port.</i>
PORT_USBHID	13	<i>USB HID device port.</i>
PORT_USBHOST1	14	<i>USB host mode port.(Host port 1)</i>
PORT_USBDEV1	19	<i>USB device mode port</i>

Table 17.2 Return code list of USB port functions

Macro	Value	Description
USB_ERR_NOT_OPEN	-3403	<i>Channel is not open.</i>

USB_ERR_BUF	-3404	<i>Send buffer error.</i>
USB_ERR_NOT_FREE	-3405	<i>No free port.</i>
USB_ERR_NO_CONF	-3411	<i>Enumeration and configuration process are not completed.</i>
USB_ERR_DISCONNECT	-3412	<i>Disconnected with the host.</i>
USB_ERR_MEM_SYSTEM	-3413	<i>System memory is abnormal.</i>
USB_ERR_BUSY	-3414	<i>USB system is busy.</i>
USB_ERR_RC_SYSTEM	-3415	<i>Fail to apply for system resources.</i>
USB_ERR_DEV_ABSENT	-3416	<i>The device is absent.</i>
USB_ERR_INVALID	-3417	<i>USB communication state is invalid.</i>

17.2 Communication Control

17.2.1 OsPortOpen

Prototype	int OsPortOpen(int Channel, const char *Attr);	
Function	Open communication port and set port attributes.	
Parameters	Channel	<p>Please refer to the Macro definition list of communication ports.</p> <p>For S800, PORT_COM2 and PORT_PINPAD are multiplex, and only one of them can be used at a time.</p> <p>S920 only has two ports: PORT_USBDEV and PORT_USBHOST, please refer to Appendix 4.</p>
	Attr[Input]	<p>A pointer to communication port attributes.</p> <p>When <i>Channel</i> is PORT_USBDEV, PORT_USBHOST or PORT_USBHID, parameter <i>Attr</i> is ignored and can be NULL.</p> <p>When <i>Channel</i> is UART port, its format is: "baud rate, data bits, parity check bit, stop bit", each of which are separated by a comma.</p> <ol style="list-style-type: none"> Baud rate: 1200, 2400, 4800, 9600, 19200, 38400, 57600, or 115200.

		<p>2. Data bit: 7 or 8.</p> <p>3. Parity check method: "o" - odd parity; "e" - even parity; "n" - no parity.</p> <p>4. Stop bit: 1 or 2.</p> <p>For example, "9600, 8, n, 1\0".</p> <p>1. attr ="9600, 8, n, 1", it represents that the baud rate is 9600bps; 8 data bits; no parity; 1 stop bit.</p>
Return		<p>RET_OK Succeeded.</p> <p>ERR_DEV_BUSY Device is busy.</p> <p>ERR_DEV_NOT_EXIS_T Port does not exist.</p> <p>ERR_INVALID_PARA_M Invalid parameter.</p>
Instruction		<p>1. When program starts, OsPortOpen() must be called successfully to open communication port; otherwise, the related functions will fail to work.</p> <p>2. Soft flow control and hard flow control will be closed.</p> <p>3. Only P^X7 and P^X5 support 921,600 baud rate, while other models don't .</p> <p>4. For device which conforms to HID Category Specification and connects to terminal with USB device, PORT_USBHID port can be used to read data. For example, it supports the common USB barcode scanner device.</p> <p>5. For terminal with only one USB HOST port, channel PORT_USBHOST should be used to open the USB HOST port; For terminal with two USB HOST ports, channel PORT_USBHOST and PORT_USBHOST1 should be used to open their corresponding USB HOST port, respectively.</p>

NOTE

1. Prolin-2.4 system will start the XCB service with USB by default. OsRegSetValue("persist.sys.xcb.enable", "0") should be called in main() to close the XCB service if application needs to use USB or serial port so as to avoid resource conflict.
2. XCB service can be opened in following ways:
 - Call OsRegSetValue("persist.sys.xcb.enable", "1") in the main application;
 - Select a connection mode among COM, USB and Network in

TM.

17.2.2 OsPortClose

Prototype	void OsPortClose(int Channel);	
Function	Close specified port.	
Parameters	Channel	Please refer to Macro definition list of communication ports .
Return	None.	
Instruction	This function shall be called to close device while program exits.	

17.2.3 OsPortSend

Prototype	int OsPortSend(int Channel, const void *SendBuf, int SendLen);	
Function	Send data to specified communication port.	
Parameters	Channel	Please refer to Macro definition list of communication ports .
	SendBuf[Input]	Send buffer.
	SendLen	Length of data that will be sent. The valid data length is not more than 8*1024 bytes.
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPEN	Port is not open.
	ERR_INVALID_PARAM	Invalid parameter.
Instruction	<ol style="list-style-type: none"> 1. The size of send buffer for every port is allocated 8k bytes by Prolin. If <i>SendLen</i> is less than the free space of send buffer, this function will return immediately and all the send data is only stored in the send buffer. 2. When calling OsPortClose(), the system will keep blocking until the data in send buffer has been sent out. 3. When using PORT_USBHID port, sending data is not supported by this function. 	

17.2.4 OsPortRecv

Prototype	<code>int OsPortRecv(int Channel, void *RecvBuf, int RecvLen, int TimeoutMs);</code>	
Function	Receive data from specified communication port.	
Parameters	Channel	Please refer to the Macro definition list of communication ports .
	RecvBuf[Output]	Received buffer.
	RecvLen	Expected length of received data; RecvLen=0 means clearing the receive buffer.
	TimeoutMs	Timeout[unit:ms] The minimum timeout is 100ms. The maximum value is 25,500 ms. When TimeoutMs exceeds this value, ERR_INVALID_PARAM will be returned.
Return	>=0	Actual length of received data.
	ERR_DEV_NOT_OPEN	Port is not open.
	ERR_INVALID_PARAMETER	Invalid parameter
Instruction	<ol style="list-style-type: none"> The function will return immediately if received data length equals to <i>RecvLen</i>. The function will wait until timeout if length of received data is less than <i>RecvLen</i>. When using PORT_USBHID port to receive data from HID device, system buffer can cache data with 500 characters at most. If the cached data exceeds the limit and this function has not been called, the cached data will be overwritten, and it will result in incomplete data. When using PORT_USBHID port to receive data from HID device, the following two scanner are supported: <ol style="list-style-type: none"> A scanner whose scanned data ending with an end character: with this scanner, the RecvLen can be greater than or equal to 500 characters and all the system buffer data can be read at once. A scanner whose scanned data ending without an end character: with this scanner, set RecvLen to 1 (that is reading in single-byte) and timeout to 100ms. If a value is scanned, an additional loop is added to continuously 	

	<p>call OsPortRecv(), and the data of OsPortRecv() is appended after each result until the timeout, the last obtained string data is the result of the scan. In the process of calling OsPortRecv(), the return value needs to be determined, if the return value is greater than RecvLen, the return value is subtracted from ReveLen to get a difference, and append the difference value of the single character on the data that is received by the character. For example, if the string obtained by the previous loop is "123", and the character '0' is received this time, and the return value is 3, then "000" needs to be appended to the total number of characters obtained by the previous loop, and the result is "123000".</p> <p>3) Note: the scanner only needs to scan once before the timeout. If the scanner is performed several times before the timeout, the resulting string will be superimposed.</p>
--	---

17.2.5 OsPortReset

Prototype	int OsPortReset(int Channel);	
Function	Reset communication port and clear all the data in send and receive buffers.	
Parameters	Channel [Input]	Please refer to Macro definition list of communication ports .
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_OPE_N	Port is not open.
	ERR_INVALID_PARAM	Invalid parameter.
Instruction		

17.2.6 OsPortCheckTx

Prototype	int OsPortCheckTx(int Channel);	
Function	Check the remaining bytes in sending buffer of specified communication port.	
Parameters	Channel [Input]	Please refer to Macro definition list of communication ports .
Return	>=0	The remaining data size in sending buffer.
	ERR_DEV_NOT_OPE_N	Port is not open.

	ERR_INVALID_PARAM Invalid parameter.
Instruction	

17.2.7 OsPortCheckRx

Prototype	int OsPortCheckRx(int Channel);	
Function	Check the remaining bytes in receiving buffer of specified communication port.	
Parameters	Channel [Input]	Please refer to Macro definition list of communication ports .
Return	>=0 ERR_DEV_NOT_OPE N ERR_INVALID_PARAM ERR_SYS_NOT_SUPP ORT	The remaining data size in receiving buffer. Port is not open. Invalid parameter. System does not support.
Instruction	<ol style="list-style-type: none"> When the return value is greater than 0, OsPortRecv() can receive data; Since OsPortRecv() can continue to receive data which is updated in the buffer during the timeout until the length of the received data is RecvLen, the actual length of the received data is based on the return value of OsPortRecv(). 	

17.3 POSIX Interface

Prolin serial ports are allowed to be accessed by POSIX interfaces. Only P^X7 and P^X5 series models support soft flow control and hard flow control.

For Prolin-2.4 and Prolin-phoenix-2.5 platforms, since there is no soft link for serial port, the corresponding relation between device nodes and serial ports can refer to the following table. For Prolin-cygnus-2.6 and later platforms, the device nodes corresponding to each serial port need to be determined by the soft links of /dev/ttycom1 and /dev/ttycom2, and the specific terminals will not be updated in the list.

Device node	Serial port	Corresponding module	Applicable models
/dev/ttyAMA0	Modem	Modem	S800

/dev/ttyAMA1	Wireless	Wireless	S800/S900
/dev/ttyAMA2	PORT_COM1	COM1	S800/S900/S300/ D200
/dev/ttyAMA3	PORT_PINPAD	Pinpad	S800
/dev/ttyAMA3	PORT_COM2	COM2	P ^X 5/P ^X 7
/dev/ttyhost	PORT_USBHOS T	USB	S800/S900/S300/ D200/S920//P ^X 5/P ^X 7
/dev/ttydev	PORT_USBDEV	USB	S800/S900/S300/ D200/S920//P ^X 5/P ^X 7

Specific operations of serial port are as follows:

17.3.1 Open

Open communication port and the device names are ttyAMA0, ttyAMA1, ttyAMA2, ttyAMA3.

Sample code:

```
int fd;
fd = open("/dev/ttyAMA1", O_RDWR);
if(-1 == fd){
    perror("Open uart error");
}
```

17.3.2 Read

Read data from communication port.

Sample code:

```
charbuff [1024];
int Len = 1024;
int readByte = read (fd, buff, Len);
```

17.3.3 Write

Write data to communication port. (Send)

Sample code:

```
charbuffer [1024];
```

```
int Length = 1024;
int nByte;
nByte = write (fd, buffer, Length);
```

17.3.4 Close

Close communication port.

```
close(fd);
```

17.3.5 Query the Buffer Data of Communication Port

Sample code:

```
int remain;
int count;
/* Inquiry the number of bytes remained in send buffer */
ioctl(fd, TIOCOUTQ, &remain);
/* Inquiry the number of bytes remained in receive buffer */
ioctl(fd, TIOCINQ, &count);
```

17.3.6 Clear the Buffer Data of Communication Port

Sample code:

```
/* clear the buffer data*/
tcflush(fd, TCIOFLUSH);
```

17.3.7 Set the Configuration Parameters of Communication Port

Sample code:

```
/* Set baud rate, data bits, parity bits and stop bits of uart*/
int SetTermios (int fd, int nSpeed, int nBits, char cEvent, int nStop)
{
    struct termios newtio, oldtio;

    /* Get configuration messages of UART */
    if (tcgetattr (fd, &oldtio) != 0)
    {
        printf("Get serial error\n");
        return -1;
    }
```

```
/* Initialize configuration messages*/
bzero (&newtio, sizeof (newtio));
newtio.c_cflag |= CLOCAL | CREAD;
newtio.c_cflag &= ~CSIZE;

/* Close soft flow control*/
newtio.c_iflag &= ~(ICRNL | IXON | IXOFF);

/* Close hard flowcontrol*/
newtio.c_cflag &= ~CRTSCTS;

/* Set data bits */
switch (nBits)
{
case 7:
    newtio.c_cflag |= CS7;
    break;
case 8:
    newtio.c_cflag |= CS8;
    break;
}

/* Set parity bit*/
switch (cEvent)
{
case 'o':
    newtio.c_cflag |= PARENB;
    newtio.c_cflag |= PARODD;
    newtio.c_iflag |= (INPCK | ISTRIP);
    break;
case 'e':
    newtio.c_iflag |= (INPCK | ISTRIP);
    newtio.c_cflag |= PARENB;
    newtio.c_cflag &= ~PARODD;
    break;
case 'n':
    newtio.c_cflag &= ~PARENB;
    break;
}
```

```
/* Set baud rate*/
switch (nSpeed)
{
    case 1200:
        cfsetispeed (&newtio, B1200);
        cfsetospeed (&newtio, B1200);
    case 2400:
        cfsetispeed (&newtio, B2400);
        cfsetospeed (&newtio, B2400);
        break;
    case 4800:
        cfsetispeed (&newtio, B4800);
        cfsetospeed (&newtio, B4800);
        break;
    case 9600:
        cfsetispeed (&newtio, B9600);
        cfsetospeed (&newtio, B9600);
        break;
    case 19200:
        cfsetispeed (&newtio, B19200);
        cfsetospeed (&newtio, B19200);
        break;
    case 38400:
        cfsetispeed (&newtio, B38400);
        cfsetospeed (&newtio, B38400);
        break;
    case 57600:
        cfsetispeed (&newtio, B57600);
        cfsetospeed (&newtio, B57600);
        break;
    case 115200:
        cfsetispeed (&newtio, B115200);
        cfsetospeed (&newtio, B115200);
        break;
    default:
        printf ("Not support the speed %d\n", nSpeed);
        cfsetispeed (&newtio, B9600);
        cfsetospeed (&newtio, B9600);
        return -1;
}
```

```
/* set stop bits */
if (nStop == 1)
    newtio.c_cflag &= ~CSTOPB;
else if (nStop == 2)
    newtio.c_cflag |= CSTOPB;
/* Set waiting time and the minimum number of characters */
newtio.c_cc[VTIME] = 0;
newtio.c_cc[VMIN] = 0;
/* Clear send buffer*/
tcflush (fd, TCIFLUSH);
/* Set new configuration message */
if ((tcsetattr (fd, TCSANOW, &newtio)) != 0)
{
    printf("Set serial error\n");
    return -1;
}
return 0;
}
```

18MODEM

Prolin system implements Modem communication through built-in serial ports to send AT commands to Modem. Besides, there are a series of encapsulated interfaces for developers to realize Modem communication.

18.1 Return Code List

Table 18.1 Modem return code list

Macro	Value	Description
MODEM_CONNECTING	10	<i>Dialing.</i>
MODEM_CONNECTED	0	<i>Connected.</i>
MODEM_HAVE_DIALED	6	<i>Sending numbers (used only when switching from automatically sending mode to manually answering mode).</i>
MODEM_RECV_POOL_HAVE_DATA	8	<i>Receive buffer is not empty (have received remote data).</i>
MODEM_RECVDATA_SEND_IS_FULL	9	<i>Receive buffer is not empty, and send buffer is full.</i>

MODEM_SEND_POOL_FULL	1	Send buffer is full. (In OsModemCheck(), the full status means send buffer is being used, at this time, the OsModemSend() cannot be used.)
MODEM_IDLE	11	Idle.
ERR_MDM_TXOVER	-3100	Send buffer is full. (It is a return error. If synchronous, it means the system is using the send buffer; If asynchronous, it means the send buffer overflows.)
ERR_MDM_BYPASS_BUSY	-3101	Parallel line is busy. Not applicable for Prolin S800 which has no bypass telephone port.
ERR_MDM_LINE_BUSY	-3102	Telephone line is not properly connected, or parallel line is occupied.
ERR_MDM_NO_CARRIER	-3103	Carrier wave of telephone is lost. (Fail to build synchronization chain.)
ERR_MDM_NO_ANSWER	-3104	Not answered.
ERR_MDM_CALLEE_BUSY	-3105	Line is busy.
ERR_MDM_NO_LINE	-3106	Telephone line is not connected (Line voltage is 0).
ERR_MDM_CMD_BUF_FULL	-3108	The excommand() buffer is full.
ERR_MDM_CMD_TOO_LONG	-3109	Command of excommand() is too long, exceeding 100.
ERR_MDM_CMD_NOT_SUPPORT	-3110	excommand() does not support the command.
OTHERS	-3XXX (-3111 ~)	Abnormal error code is used for abnormal error. It is not frequently used and it is just to keep the

	-3199)	<i>completeness and maintainability of programming. Its meaning is not important.</i>
	-3115	<i>Calling synchronization handshake receiving process 1 error.</i>
	-3116	<i>Calling synchronization handshake receiving data package error.</i>
	-3117	<i>Calling synchronization handshake receiving package type error.</i>
	-3118	<i>Calling synchronization handshake receiving process 2 errors.</i>
	-3119	<i>Calling synchronous communication receiving process 1 error.</i>
	-3120	<i>Calling synchronous communication chip is on-hook.</i>
	-3121	<i>Calling synchronous communication receiving packet series number error.</i>
	-3122	<i>Calling synchronous communication receiving process 2 errors.</i>
	-3123	<i>Calling synchronous communication send overload.</i>
	-3124	<i>Calling synchronous communication send underload.</i>
	-3130	<i>Calling synchronous communication line rate is illegal.</i>
	-3131	<i>Calling synchronous communication sending state packet 1 error.</i>

	-3132	<i>Calling synchronous communication sending data packets retry more than specified times.</i>
	-3133	<i>Calling synchronous communication sending data packets timeout.</i>
	-3134	<i>Calling synchronous communication receiving acknowledgement packets retry more than specified times.</i>
	-3135	<i>Calling synchronous communication sending state packet 2 errors.</i>
	-3136	<i>Calling synchronous communication sending state packet 3 errors.</i>
	-3137	<i>Calling synchronous communication sending state packet 4 errors.</i>
	-3138	<i>Calling synchronous communication receiving data packets retry more than specified times.</i>
	-3139	<i>Calling synchronous communication sending state packet 5 errors.</i>
	-3140	<i>Calling synchronous communication sending state packet 6 errors.</i>
	-3144	<p><i>Bypass telephone and parallel telephone are both idle (used only when switching from automatically sending number to manually answering).</i></p> <p><i>In automatically sending number mode, the phone is not picked up timely.</i></p>

	-3145	<i>Called synchronization handshake fails to send handshake packet.</i>
	-3146	<i>Called synchronization handshake fails to receive handshake packet.</i>
	-3147	<i>Called synchronization handshake more than specified times.</i>
	-3148	<i>Called synchronous communication sending state packet 1 error.</i>
	-3149	<i>Called synchronous communication receiving process 1 error.</i>
	-3150	<i>Called synchronous communication chip is on-hook.</i>
	-3151	<i>Called synchronous communication receiving process 2 errors.</i>
	-3152	<i>Called synchronous communication receiving retry more than specified times.</i>
	-3153	<i>Called synchronous communication sending state packet 2 errors.</i>
	-3154	<i>Called synchronous communication sending data packet error</i>
	-3155	<i>Called synchronous communication receiving process 3 errors.</i>
	-3156	<i>Called synchronous communication receiving packet retry more than specified times.</i>

	-3157	<i>Called synchronous communication sending state packet 3 errors.</i>
	-3160	<i>Called connection receiving ring information error.</i>
	-3161	<i>Called connection fails to detect line voltage.</i>
	-3162	<i>Called connection detecting line voltage data format error.</i>
	-3163	<i>Called connection voltage is less than threshold value.</i>
	-3164	<i>Called connection timeout.</i>
	-3165	<i>Called asynchronous line rate format error.</i>
	-3166	<i>Called asynchronous line rate is illegal.</i>
	-3167	<i>Called connection information format error.</i>
	-3170	<i>Called connection fails to set command string 1.</i>
	-3171	<i>Called connection fails to set command string 2.</i>
	-3172	<i>Called connection fails to set extended command string.</i>
	-3175	<i>Calling connection fails to set command string 1.</i>
	-3176	<i>Calling connection fails to set command string 2.</i>
	-3177	<i>Calling connection fails to set command string 3.</i>
	-3178	<i>Calling connection fails to set command string 4.</i>
	-3179	<i>Calling connection fails to set command string 5.</i>
	-3180	<i>Calling connection fails to set the threshold of</i>

	<i>the answering tone.</i>
-3181	<i>Calling connection asynchronous line rate is illegal.</i>
-3182	<i>Calling connection fails to get the voltage of telephone line.</i>
-3183	<i>Calling connection fails to set extended command string.</i>
-3184	<i>Calling connection fails to set the transmission level.</i>
-3185	<i>Calling connection has no dial tone.</i>
-3186	<i>Calling connection chip indicator error.</i>
-3187	<i>Calling connection digital lines is detected.</i>
-3188	<i>Calling connection has no dial tone and the voltage is too low.</i>
-3189	<i>Calling connection has other abnormal errors.</i>
-3192	<i>Non-pre-dial-up dial up timeout (300s).</i>
-3193	<i>When FSK sends data, DCD signal timeout.</i>
-3194	<i>When FSK sends data, CTS signal timeout.</i>
-3195	<i>FSK sending data timeout.</i>
-3196	<i>Called synchronous communication sending data packet format error.</i>
-3197	<i>Asynchronous communication does not support the ConnectFormat parameter.</i>
-3198	<i>Daemon fails to create thread.</i>

	-3199	<i>Communication with Daemon communication fails or error.</i>
	-3200	<i>Modem is using the binding serial port.</i>
	-3201	<i>Fail to create Socket.</i>
	-3202	<i>Fail to connect Socket.</i>
	-3203	<i>Fail to send Socket.</i>
	-3204	<i>Fail to create semaphore.</i>
	-3205	<i>Fail to set semaphore value.</i>
	-3206	<i>Semaphore is pre-occupied.</i>
	-3207	<i>Fail to release Semaphore.</i>
	-3208	<i>Fail to initialize Semaphore.</i>
	-3209	<i>Fail to gettimeofday.</i>
	-3210	<i>More than 2 links are using modem daemon.</i>
ERR_MDM_CANCEL_CONNECT	-3211	<i>Received cancel button in dial-up process.</i>
	-3212	<i>The request of receiving data is rejected. (Receive buffer is empty.)</i>
	-3213	<i>Calling connection fails to set command string 7.</i>
	-3214	<i>Calling connection fails to set command string 8.</i>
	-3215	<i>FSK sending timeout, but still has data in send buffer.</i>
	-3216	<i>Invalid data length (len=0 or len>2048), won't send data.</i>
ERR_MDM_INIT	-3217	<i>Fail to initialize Modem.</i>
	-3218	<i>If no or error implementation of OsModemConnect(),</i>

	<i>then implement OsPppomLogin() orOsPppomCheck().</i>
-3219	<i>Modem or ModemPPP is being used, Modem cannot be powered off.</i>
-3220	<i>Modem is not powered on.</i>

18.2 Data Structure

ST_MODEM_SETUP structure of MODEM:

ST_MODEM_SETUP:

```
typedef struct {

    int CallMode;

    int CommMode;

    int CodeType;

    int CodeDuration;

    int CodeSpacing;

    int DetectLineVoltage;

    int DetectDialTone;

    int DialToneTimeout;

    int CommaPauseTime;

    char ConnectRate[20];

    char ConnectFormat[20];

    int ConnectTimeout;
```

```

int DialTimes;

int IdleTimeout;

int Pppom;

int Reserved[9];

}ST_MODEM_SETUP;

```

Table 18.2 Variable definitions of ST_MODEM_SETUP

CallMode	<i>MODEM_PRE_DIAL</i>	<i>Pre-dial.</i>
	<i>MODEM_DIAL</i>	<i>Dial.</i>
	<i>MODEM_WAIT_CALL</i>	<i>Called/Answer the call.</i>
CommMode	<i>MODEM_COMM_SYNC</i>	<i>Synchronization.</i>
	<i>MODEM_COMM_ASYN</i>	<i>Asynchronization.</i>
CodeType	<i>MODEM_CODE_DTMF</i>	<i>DTMF(Dual Tone Multiple Frequency) dialing.</i>
	<i>MODEM_CODE_PULSE1</i>	<i>Pulse dialing 1(Pulse rate 10/s; break-make ratio 1.6:1; signal interval >=500ms).</i>
	<i>MODEM_CODE_PULSE2</i>	<i>Pulse dialing 2(Pulse rate 10/s; break-make ratio 2:1; signal interval >=600ms).</i>
	<i>Other values</i>	<i>Reserved.</i>
CodeDuration	<i>The duration time of dual-tone dial-up a single number[Unit:10ms].</i> <i>The valid value ranges from 5 to 25.</i>	
CodeSpacing	<i>The interval time between two numbers of dual-tone dial-up.[Unit:10ms]</i> <i>Chip 93011 cannot set this value and S800 does not support it.</i> <i>The valid value ranges from 5 to 25.</i>	
DetectLineVoltage	<i>TRUE</i>	<i>Detect whether the parallel telephone is occupied (used in Caller dialing or</i>

		<i>none-manually-answering mode).</i>
	<i>FALSE</i>	<i>Do not detect whether the parallel telephone is occupied (used in Caller dialing or none-manually-answering mode).</i>
DetectDialTone	<i>TRUE</i>	<i>Detect dial tone. Refer to the instruction of DialTone Timeout.</i>
	<i>FALSE</i>	<i>Do not detect dial tone.</i>
DialToneTimeout	<p>1. If detecting dial tone: <i>DialToneTimeout () refers to the longest waiting time for the dial tone after off-hook. During this process, system will exit waiting as long as the dial tone is detected.[Unit: 100ms]</i></p> <p>2. If not detecting dial tone: <i>DialToneTimeout () refers to the waiting time from off-hook to dialing.[Unit: 100ms]</i></p> <p><i>The valid timeout ranges from 20 to 50. Minimum value and the default value are both set to 20.</i></p> <p><i>In both cases above, the time is started about 450~500ms after off-hook.</i></p>	
CommaPauseTime	<p><i>,”waiting time when dialing an outside line.[Unit: 100ms]</i></p> <p><i>This value should be set according to the actual application environment. Some interfaces should be reserved for manual setting.</i></p> <p><i>The valid range is from 0 to 255. The value range doesn't apply to S800 whose valid range is from 1 to 26s. (It is inconsistent with the Datasheet because of the modem patch).</i></p>	
ConnectRate[20]	<p><i>The rate of connection and communication(Expressed as a string):</i></p> <p><i>“1200”/*1200 bps fast connect */</i></p> <p><i>“1200,V22”/*1200 bps normal connect */</i></p> <p><i>“1200,V23C”/*1200 bps for V.23C(FSK) */</i></p> <p><i>“1200,B202”/*1200 bps for Bell 202(FSK) */</i></p> <p><i>“2400,FC”/*2400 bps fast connect */</i></p> <p><i>“2400”/*2400 bps normal connect*/</i></p> <p><i>“4800”/*4800 bps */</i></p> <p><i>“7200”/*7200 bps*/</i></p> <p><i>“9600”/*9600 bps */</i></p>	

```

“1200”/*1200 bps*/
“1440”/*1440 bps */
“1920”/*1920 bps */
“2400”/*2400 bps */
“2640”/*2640 bps */
“2880”/*2880 bps */
“3120”/*3120 bps */
“3360”/*3360 bps */
“4800”/*4800 bps */
“5600”/*5600 bps */

```

For null string "\0", the system will select "1200" by default in synchronous communication and select the maximum rate that the chip supports by default in asynchronous communication.

S800 supports baud rate.

Asynchronization:

```

“1200”/*1200 bps */
“1200,V22”/*1200 bps normal connect */
“1200,V23C”/*1200 bps for V.23C(FSK) */
“1200,B202”/*1200 bps for Bell 202(FSK) */
“2400,FC”/*2400 bps fast connect */
“2400”/*2400 bps*/
“4800”/*4800 bps */
“7200”/*7200 bps */
“9600”/*9600 bps */
“12000”/*12000 bps */
“14400”/*14400 bps */
“19200”/*19200 bps */
“24000”/*24000 bps */
“26400”/*26400 bps */
“28800”/*28800 bps */
“31200”/*31200 bps */
“33600”/*33600 bps */
“48000”/*48000 bps */
“56000”/*56000 bps */

```

Synchronization:

```

“1200”/*1200 bps */
“1200,V22”/*1200 bps normal connect */
“2400,FC”/*2400 bps fast connect */
“2400”/*2400 bps*/

```

	“9600”/*9600 bps*/
ConnectFormat[20]	<p><i>Format of connection and communication(Expressed as a string):</i></p> <p>“8, n, 1” “8, e, 1” “8, o, 1” “7, e, 1” “7, o, 1”</p> <p><i>For null string “\ 0”, the system will select “8, n, 1” by default.</i></p> <p><i>For synchronous communication, the system will select “8, n, 1” automatically.</i></p>
ConnectTimeout	<p><i>Connection timeout. [unit: second]</i></p> <p><i>The valid timeout ranges from 0 to 60s.</i></p>
DialTimes	<p><i>The total number of dial-up cycle.</i></p> <p><i>The valid range is 1 to 255. 0 will be converted to 1 automatically.</i></p> <p><i>Dialing all the dialer numbers is one dial-up cycle.</i></p>
IdleTimeout	<p><i>If there is no data exchange in application-layer within specified time, MODEM will hang up.[Unit: 10s]</i></p> <p><i>0 means no timeout. The maximum timeout is 900s. This value is invalid to ModemPPP.</i></p>
Pppom	<p>TRUE Modem PPP communication</p> <p>FALSE Common Modem communication</p>
Reserved[9]	Reserved.

18.3 OsModemOpen

Prototype	int OsModemOpen(void);	
Function	Open Modem device.	
Parameters	None.	
Return	RET_OK	Succeeded.
	ERR_DEV_NOT_EXIST	Device does not exist.
	ERR_DEV_BUSY	Device is busy.
	ERR_NO_PORT	No communication port.
Instruction	1. This function must be called successfully to open Modem device; otherwise, other related functions will not work.	

2. OsModemSwitchPower() must be called to power on the device before calling OsModemOpen().

18.4 OsModemClose

Prototype	void OsModemClose(void);	
Function	Close Modem device.	
Parameters	None.	
Return	None.	
Instruction	This function can only close the modem that is opened by the same application. If the modem device is not open, this function will not work.	

18.5 OsModemSwitchPower

Prototype	int OsModemSwitchPower(int OnOff);	
Function	Manage Modem power.	
Parameters	int OnOff	OnOff=1, power on, OnOff=0, power off.
Return	RET_OK	Succeeded. -3219 Modem or ModemPPP is being used, and Modem cannot be powered off. -3220 Modem isn't powered on.
Instruction	1. Modem must be powered on before calling Modem or ModemPPP. 2. This function is independent of the interfaces in Modem module. 3. OsModemClose() is not automatically performed when Modem module is powered off.	

18.6 OsModemConnect

Prototype	int OsModemConnect(const ST_MODEM_SETUP *Setup, const unsigned char *TelNo);
------------------	---

Function	Establish Modem communication link suitable for both dialing and called mode.	
Parameters	Setup[Input]	A pointer to Modem parameter structure ST MODEM SETUP . While mdm_setup==NULL, default dialing parameter will be used. The default dialing mode is synchronous, 1200, DTMF, connection timeout for 10 seconds, idle hang up for 60 seconds.
	TelNo[Input]	Telephone number.
Return	RET_OK	Succeeded.
	ERR_MDM_BYPASS_BUSY	Parallel line is busy.
	ERR_MDM_LINE_BUSY	Telephone line is not properly connected, or parallel line is occupied.
	ERR_MDM_NO_ANSWER	Not answered.
	ERR_MDM_CALLEE_BUSY	Line is busy.
	ERR_MDM_NO_LINE	Telephone line is not connected (Line voltage is 0).
	ERR_MDM_NO_CARRIER	Carrier wave of telephone is lost.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_MDM_CANCEL_KEY_DOWN	Dial is canceled during the dialing process. When OsModemConnect() is blocking (that is, CallMode==MODEM_DIAL), dialing can be canceled by creating a thread to call OsModemConnect(NULL,NULL).
	ERR_DEV_NOT_OPEN	Device is not open.
Instruction	<ol style="list-style-type: none"> 1. The function can also be used to set Modem mode to be called. 2. Telephone icon is controlled by modem driver. It shows hang-up icon when connecting; it shows pickup icon when communication is established, or during switching time of pre-dial after hang-up. 3. The OsModemCheck()is used to query the result of 	

	<p>pre-dial.</p> <p>4. For ModemPPP, OsModemConnect() should be called first and set ST_MODEM_SETUP.Pppom to 1 (OsModemOpen() doesn't need to be executed), then call OsPppomLogin().</p> <p>5. After Modem dialed successfully, calling OsSysSleep() or OsSysSleepEx(2) will return ERR_DEV_BUSY, and the system cannot sleep. After Modem is hung up, the system can sleep by calling sleep function.</p>
--	---

NOTE

Code meanings of telephone numbers:

0-9, *, #, A~D — Telephone numbers

, — Dialing delayed.

; — Transmitting next telephone number.

. — End mark of numbers, which is used to keep connecting with application after sending numbers.

.. — Extended end mark of numbers, which is used when switching to manual answering after sending numbers.

18.7 OsModemCheck

Prototype	int OsModemCheck(void);	
Function	Check the status of Modem and telephone line.	
Parameters	None.	
Return	MODEM_CONNECTING	It is dialing.
	MODEM_CONNECTED	Connected.
	MODEM_IDLE	Idle.
	MODEM_RECV_POOL_HAVING_DATA	Receive buffer is not empty (Have received remote data).
	MODEM_SEND_POOL_FULL	Send buffer is full.
	ERR_MDM_BYPASS_BUSY	Parallel line is busy.
	ERR_MDM_LINE_BUSY	Telephone line is not properly connected, or parallel line is occupied.
	ERR_MDM_NO_ANSWER	Not answered.
	ERR_MDM_CALLEE_BUSY	Line is busy.

	Y	
	ERR_MDM_NO_LINE	Telephone line is not connected (Line voltage is 0).
	ERR_MDM_NO_CARRIER	Carrier wave of telephone is lost.
	ERR_DEV_NOT_OPEN	Device is not open.
Instruction	<ol style="list-style-type: none"> 1. This function can be used to check whether the pre-dial is successfully connected. 2. After calling OsModemOpen(), OsModemHangup() or OsModemClose(), the status of the last Modem dial will become MODEM_IDLE. 	

18.8 OsModemExCmd

Prototype	<code>int OsModemExCmd(const char *Cmd, char *Rsp, int *RespLen, int TimeoutMs);</code>	
Function	Set additional AT control command for OsModemConnect() to control the connection behavior of Modem dialing.	
Parameters	Cmd[Input]	Input AT control command.
	Rsp[Output]	Response data.
	RespLen[Output]	Length of response data.
	TimeoutMs	Waiting time for response.[unit:ms]
Return	RET_OK	Succeeded.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_MDM_CMD_BUF_FULL	Command buffer overflow.
	ERR_MDM_CMD_TOO_LONG	Command is too long.
	ERR_MDM_CMD_NOT_SUPPORTED	Does not support the command. (command that doesn't begin with AT,S3,S7,WT=)
	ERR_DEV_NOT_OPEN	Device is not open.
Instruction	<ol style="list-style-type: none"> 1. The function needs to be called before OsModemConnect(); it is only valid during the current calling process of OsModemConnect(). 2. While executing this function, the current dialing or 	

	<p>communication process will automatically hang up.</p> <p>3. The function can be called 100 times continuously at most and the exceeding callings will be discarded and report error.</p>
--	---

CAUTION

1. The maximum string length is 100 bytes for each call. If more than 100 bytes, the entire control command will be discarded and report error.
2. The input control command must be the AT control command supported by this Modem chip. Otherwise, it will lead to OsModemConnect() failure.

18.9 OsModemHangup

Prototype	void OsModemHangup(void);
Function	Hang up Modem or terminate Modem dialing.
Parameters	None.
Return	None.
Instruction	

NOTE

If dialing the number again right after hanging up, the Modem will wait and start redialing after 3 seconds so as to allow PABX to finish hanging up action and transmitting dialing tone again.

18.10 OsModemSend

Prototype	int OsModemSend(const void *SendBuf, int SendLen);	
Function	Send data packets through Modem.	
Parameters	SendBuf[Input]	Send buffer.
	SendLen	Length of packets which will be sent.[unit: bytes]
Return	RET_OK	Succeeded.
	ERR_NOT_CONNECT	Not connected.

	ERR_MDM_TXOVER	Send buffer is full.
	ERR_MDM_NO_CARRIER	No carrier waves.(Disconnected)
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_DEV_NOT_OPEN	Device is not open.
Instruction	The maximum length of sent data is 2048 bytes each time. As there are 5 additional control characters when called in synchronous communication, the maximum length of received and sent data are both 2053 bytes.	

18.11 OsModemRecv

Prototype	int OsModemRecv(void *RecvBuf, int BufSize, int Timeout);	
Function	Receive packets returned from MODEM.	
Parameters	RecvBuf[Output]	Buffer of received packets. [Buffer size can be defined according to actual practice.]
	BufSize	Size of RecvBuf. The valid buffer size is not more than 2048 bytes.
	Timeout	Timeout.[ms]
Return	= 0	The actual number of received data.
	ERR_NOT_CONNECT	Not connected.
	ERR_MDM_NO_CARRIER	No carrier waves.(Disconnected)
	ERR_INVALID_PARAM	Invalid parameter
	ERR_DEV_NOT_OPEN	Device is not open.
Instruction	<ol style="list-style-type: none"> 1. The maximum length of received packet is 2048 bytes each time. As there are 5 additional control characters when called in synchronous communication, the maximum length of received and sent data are both 2053 bytes. 2. If the size of actually received data is not larger than the size of specified receive buffer within specified time, it will return immediately. 3. If there is line error when receiving data, it will immediately 	

	<p>return corresponding error code.</p> <ol style="list-style-type: none"> 4. For SDLC synchronous communication, it will immediately return after receiving any packet. (Even if the received packet length is less than the BufSize.) 5. For asynchronous communication, it will return after receiving BufSize bytes of data; otherwise, it will wait until timeout. 6. For synchronous receiving, it will receive a complete frame each time, and its length is not limited by BufSize. 7. For FSK, timeout does not work.
--	--

18.12 OsPppomLogin

Prototype	<pre>int OsPppomLogin(const char *Name const char *Password, long Auth, int TimeOutMs);</pre>					
Function	Establish Modem PPP network link.					
Parameters	<p>Name[Input]</p> <p>User name. The valid string length is not more than 50 bytes. For the 96169 background of China Telecom, any user name with at least one character can be entered. It can't be NULL.</p>					
	<p>Password[Input]</p> <p>Password. The valid string length is not more than 50 bytes. For the 96169 background of China Telecom, any user name with at least one character can be entered. It can't be NULL.</p>					
	<p>Auth</p> <p>Authentication algorithms. The supported authentication algorithms are:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;">PPP_ALG_PA P</td> <td style="padding: 5px;">0x00 0000 01</td> <td style="padding: 5px;">PAP Authentication Algorithm</td> </tr> <tr> <td style="padding: 5px;">PPP_ALG_CH AP</td> <td style="padding: 5px;">0x00 0000 02</td> <td style="padding: 5px;">CHAP Authentication Algorithm</td> </tr> </table>	PPP_ALG_PA P	0x00 0000 01	PAP Authentication Algorithm	PPP_ALG_CH AP	0x00 0000 02
PPP_ALG_PA P	0x00 0000 01	PAP Authentication Algorithm				
PPP_ALG_CH AP	0x00 0000 02	CHAP Authentication Algorithm				

		<table border="1"> <tr> <td>PPP_ALG_MS CHAPV1</td><td>0x00 0000 04</td><td>MSCHAPV1 Authentication Algorithm</td></tr> <tr> <td>PPP_ALG_MS CHAPV2</td><td>0x00 0000 08</td><td>MSCHAPV2 Authentication Algorithm</td></tr> <tr> <td>PPP_ALG_ALL</td><td>0xff</td><td>All Authentication Algorithms</td></tr> </table>	PPP_ALG_MS CHAPV1	0x00 0000 04	MSCHAPV1 Authentication Algorithm	PPP_ALG_MS CHAPV2	0x00 0000 08	MSCHAPV2 Authentication Algorithm	PPP_ALG_ALL	0xff	All Authentication Algorithms
PPP_ALG_MS CHAPV1	0x00 0000 04	MSCHAPV1 Authentication Algorithm									
PPP_ALG_MS CHAPV2	0x00 0000 08	MSCHAPV2 Authentication Algorithm									
PPP_ALG_ALL	0xff	All Authentication Algorithms									
		<p>Either one type of authentication or several authentications with (+) or () should be used, for example, PPP_ALG_PAP PPP_ALG_CHAP.</p> <p>If the algorithm is unknown, fill in PPP_ALG_ALL.</p>									
	TimeOutMs	<p>Timeout.[unit:ms]</p> <p>The valid timeout ranges from 0 to 3600000.</p> <p>If timeout<0, it will be automatically set to 0.</p> <p>If timeout >3600000, it will be automatically set to 3600000.</p>									
Return	PPP_LOGINING	In the process of logging in.									
	RET_OK	The link established successfully.									
	ERR_INVALID_PARAM	Invalid parameter.									
	ERR_NET_PASSWD	Wrong password.									
	ERR_NET_SERVER_BUS Y	Server is busy, communication failed.									
Instruction		<ol style="list-style-type: none"> 1. Before calling this function, OsModemConnect() must be called first and set the ST_MODEM_SETUP.Pppom to 1 (OsModemOpen() doesn't need to be called). 2. TimeOutMs=0 means no wait time and will return immediately. 3. Call OsPppomCheck() to check the link status. 4. The login time varies according to the settings of ST_MODEM_SETUP. The maximum asynchronous baud rate supported by S800 modem chip is 33600. Dialing-up when baud rate is not more than 33600 will enjoy a low re-training rate and high success rate. 5. For 96169 background (Guidway A8010), if re-training occurs and the duration time is more than 20 seconds after sending number, then the background communication will no longer conform to ppp protocol and fail to establish link. 6. After successfully setup the link, it can communicate through 									

the IP network communication function.

7. If users want to hang up by pressing the cancel button in the dialing process, operate as follows:

Start a thread and take a key. If it is the cancel key, perform OsPppomLogin ("a", "a", 1, -1). The first 3 parameters should be filled in accordance with the requirements, and the fourth parameter must be set to -1, then ModemPPP will hang-up and automatically logout.

18.13 OsPppomCheck

Prototype	int OsPppomCheck(void);	
Function	Check the link status of Modem network.	
Parameters	None.	
Return	PPP_LOGOUTING	Disconnected.
	PPP_LOGINING	In the process of logging in.
	RET_OK	Link is established successfully.
	ERR_NET_PASWD	Wrong password.
	ERR_NET_LOGOUT	OsPppomLogout() is called to disconnect the link.
	ERR_NET_IF	Link is disconnected.
Instruction		

18.14 OsPppomLogout

Prototype	int OsPppomLogout(void);
Function	Exit network and disconnect Modem PPP link.
Parameters	None.
Return	RET_OK Succeeded.
Instruction	

19Network Communication

Prolin uses unified network protocol stack to manage different physical links. Prolin provides a standard socket programming for network communication.

19.1 TCP Programming

TPC programming sample code:

```
/* Server code in C language*/  
  
#include <sys/types.h>  
#include <sys/socket.h>  
#include <netinet/in.h>  
#include <arpa/inet.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <unistd.h>  
  
int main(void)
```

```

{
    struct sockaddr_in stSockAddr;
    int SocketFD = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);

    if(-1 == SocketFD)
    {
        perror("cannot create socket");
        exit(EXIT_FAILURE);
    }
    memset(&stSockAddr, 0, sizeof(stSockAddr));
    stSockAddr.sin_family = AF_INET;
    stSockAddr.sin_port = htons(1100);
    stSockAddr.sin_addr.s_addr = INADDR_ANY;
    if(-1 == bind(SocketFD,(struct sockaddr *)&stSockAddr,
    sizeof(stSockAddr)))
    {
        perror("error bind failed");
        close(SocketFD);
        exit(EXIT_FAILURE);
    }
    if(-1 == listen(SocketFD, 10))
    {
        perror("error listen failed");
        close(SocketFD);
        exit(EXIT_FAILURE);
    }

    for(;;)
    {
        int ConnectFD = accept(SocketFD, NULL, NULL);
        if(0 > ConnectFD)
        {
            perror("error accept failed");
            close(SocketFD);
            exit(EXIT_FAILURE);
        }
        /* perform read-write operations ...  read(ConnectFD,buff,size) */
        if (-1 == shutdown(ConnectFD, SHUT_RDWR))
        {
            perror("cannot shutdown socket");
        }
    }
}

```

```

        close(ConnectFD);
        exit(EXIT_FAILURE);
    }
    close(ConnectFD);
}
return EXIT_SUCCESS;
}

/* Client code in C language*/
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
int main(void)
{
    struct sockaddr_in stSockAddr;
    int Res;
    int KeepAlive = 1;
    int SocketFD = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (-1 == SocketFD)
    {
        perror("cannot create socket");
        exit(EXIT_FAILURE);
    }
    Res = setsockopt(SocketFD, SOL_SOCKET, SO_KEEPALIVE, (void *)
*)&KeepAlive, sizeof(KeepAlive));
    if (-1 == Res)
    {
        perror("cannot set keepalive");
        exit(EXIT_FAILURE);
    }
    memset(&stSockAddr, 0, sizeof(stSockAddr));
    stSockAddr.sin_family = AF_INET;
    stSockAddr.sin_port = htons(1100);
    Res = inet_pton(AF_INET, "192.168.1.3", &stSockAddr.sin_addr);
    if (0 > Res)
    {

```

```

    perror("error: first parameter is not a valid address family");
    close(SocketFD);
    exit(EXIT_FAILURE);
}
else if (0 == Res)
{
    perror("char string (second parameter does not contain valid
ipaddress)");
    close(SocketFD);
    exit(EXIT_FAILURE);
}
if (-1 == connect(SocketFD, (struct sockaddr *)&stSockAddr,
sizeof(stSockAddr)))
{
    perror("connect failed");
    close(SocketFD);
    exit(EXIT_FAILURE);
}
/* perform read write operations ... */
shutdown(SocketFD, SHUT_RDWR);
close(SocketFD);
return EXIT_SUCCESS;
}

```

NOTE

Prolin TCP communication complies with Linux standard. KeepAlive function is disabled by default, which can be enabled by calling setsockopt() to set SO_KEEPALIVE parameter. If Keepalive is enabled, a detection packet will be sent every 30 seconds when TCP is idle.

19.2 UDP Programming

UDP programming sample code:

```

#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>

```

```

#include <netinet/in.h>
#include <unistd.h> /* for close()socket */
#include <stdlib.h>

int main(void)
{
    int sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP);
    struct sockaddr_in sa;
    char buffer[1024];
    ssize_t recsize;
    socklen_t fromlen;
    memset(&sa, 0, sizeof sa);
    sa.sin_family = AF_INET;
    sa.sin_addr.s_addr = INADDR_ANY;
    sa.sin_port = htons(7654);
    fromlen = sizeof(sa);
    if (-1 == bind(sock,(struct sockaddr *)&sa, sizeof(sa)))
    {
        perror("error bind failed");
        close(sock);
        exit(EXIT_FAILURE);
    }
    for (;;)
    {
        printf ("recv test....\n");
        recsize = recvfrom(sock, (void *)buffer, sizeof(buffer), 0, (struct sockaddr *)
*)&sa, &fromlen);
        if (recsize < 0) {
            fprintf(stderr, "%s\n", strerror(errno));
            exit(EXIT_FAILURE);
        }
        printf("recsize: %z\n ", recsize);
        sleep(1);
        printf("datagram: %. *s\n", (int)recsize, buffer);
    }
}

#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <string.h>

```

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <unistd.h>
#include <arpa/inet.h>

int main(int argc, char *argv[])
{
    int sock;
    struct sockaddr_in sa;
    int bytes_sent;
    char buffer[200];
    strcpy(buffer, "hello world!");
    sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP);
    if (-1 == sock) /* if socket failed to initialize, exit */
    {
        printf("Error Creating Socket");
        exit(EXIT_FAILURE);
    }
    memset(&sa, 0, sizeof sa);
    sa.sin_family = AF_INET;
    sa.sin_addr.s_addr = inet_addr("127.0.0.1");
    sa.sin_port = htons(7654);
    bytes_sent = sendto(sock, buffer, strlen(buffer), 0,(struct sockaddr*)&sa,
sizeof sa);
    if (bytes_sent < 0) {
        printf("Error sending packet: %s\n", strerror(errno));
        exit(EXIT_FAILURE);
    }
    close(sock); /* close the socket */
    return 0;
}
```

20Network Configuration

Prolin provides network configuration interfaces, including ARP setting, Ping service, DNS configuration, network card configuration, DHCP service, routing setting and PPPoE service etc.

20.1 Return Code List

Table 20.1 Network configuration return code list

Macro	Value	Description
<i>ERR_NET_SERVER_BAD</i>	-3301	<i>IP server is abnormal.</i>
<i>ERR_NET_SERVER_BUSY</i>	-3302	<i>IP server is busy; it can serve for at most 5 applications at the same time.</i>
<i>ERR_NET_NO_ROUTE</i>	-3305	<i>Router is not configured.</i>
<i>ERR_NET_FULL</i>	-3306	<i>Connection is full; application can establish at most 20 connections at the same time.</i>
<i>ERR_NET_IF</i>	-3307	<i>Network interface link cannot be used. (The link is not established or has no corresponding</i>

		<i>device.)</i>
ERR_NET_SESS_BROKEN	-3308	<i>TCP / UDP session is disconnected.</i>
ERR_NET_PASSWD	-3309	<i>Wrong password.</i>
ERR_NET_LOGOUT	-3310	<i>Application logout actively.</i>
ERR_NET_INIT	-3311	<i>Initialization failed.</i>
ERR_NET_DHCP_DISCOVER	-3312	<i>DHCP Server is not found.</i>
ERR_NET_DHCP_OFFER	-3313	<i>DHCP cannot get the IP address.</i>
ERR_NET_DHCP_START	-3314	<i>DHCP is not started.</i>
ERR_NET_DNS	-3315	<i>DNS cannot analyze the corresponding domain.</i>
ERR_NET_SERV_USING	-3316	<i>The port specified by the server is in use.</i>
ERR_NET_NODNServer	-3317	<i>Domain name server is not configured.</i>
ERR_NET_LINKDOWN	-3318	<i>Link is disconnected by the server.</i>
ERR_NET_CONN	-3319	<i>Cannot connect to the specified server.</i>
ERR_NET_TIMEOUT	-3320	<i>Connection timeout.</i>
ERR_NET_PPP	-3321	<i>PPP connection error.</i>
ERR_NET_SERV	-3322	<i>PPPoE server is not found.</i>
ERR_NET AGAIN	-3323	<i>Request resource is not found, please try again.</i>
ERR_NET_AUTH	-3324	<i>Cannot connect to RADIUS server.</i>
ERR_PPP_SERV_NOTREADY	-3325	<i>Windows server is not ready when connected to ppp_direc.</i>
ERR_NET_ICMPV6_UNREACH	-3327	<i>Target is unreachable.</i>
ERR_NET_ICMPV6_PKT_TOOBIG	-3328	<i>SIZE is greater than the minimum MTU.</i>
ERR_NET_ICMPV6_TIME_EXCEEDED	-3329	<i>TTL exceeds the maximum value (default value: 64).</i>

ERR_NET_ICMPV6_PARAMPROB	-3330	Protocol parameter error, cannot process middle route or target node.
ERR_NET_DHCPV6_DNS	-3331	Unrecognizable keywords or illegal IP address in Resolv.conf.
ERR_NET_DHCP6_SOLICITE	-3335	DHCPv6 server has no response.
ERR_NET_DHCP6_START	-3336	DHCPv6 is not started.
ERR_NET_DHCP6_REQUEST	-3337	Send request command; no correct response.
ERR_NET_DHCP6_INFORMATION	-3338	Send information request, no correct response.
ERR_NET_DHCP6_VALID_LIFE	-3339	Expired address.
ERR_NET_DHCP6_NOADDRSAVAIL	-3340	No available address.
ERR_NET_DHCP6_DUPLICATE_ADDR	-3341	Address conflict.
ERR_ROUTE_EXIST	-3360	Route already exists.
ERR_ROUTE_NONEXIST	-3361	The specified route doesn't exist.
NET_DOING	1	Related operations are in process (such as PPP login, DHCP).

20.2 Data Definition

20.2.1 Physical Channel List

Table 20.2 Physical channel list

Macro	Description
NET_LINK_ETH	Ethernet.
NET_LINK_WL	Wireless network, including GPRS, CDMA, TDSCDMA

NET_LINK_WIFI	<i>WiFi</i>
NET_LINK_PPPOE	<i>ADSL link</i>
NET_LINK_MODEM	<i>Modem PPP link</i>
NET_LINK_PPPDIRECT	<i>ppp_direct link</i>
NET_LINK_BT	<i>Bluetooth link</i>
NET_LINK_USB	<i>USB link</i>

NOTE

1. All of the macro values listed above are positive integers.
2. For more details, refer to “osal.h”.

20.2.2 IPv6 Status Value List

Table 20.3 IPv6 status value list

Macro	Value	Description
IP6_STATELESS_ADDR_AUTOCONF	0	Stateless address auto-configuration.
IP6_DHCP6_STATEFULL_ADDR_CONFIG	1	DHCPv6 statefull address configuration.
IP6_DHCP6_STATELESS_ADDR_CONFIG	2	DHCPv6 stateless address configuration.

20.2.3 Data Structure

Structure: **struct IPv6Info**

IPv6Info is used to get an existed IPV6 address, and link address, site address and global address are different.

Struct **IPv6Info**

```
Struct IPv6Info{
    char LinkAddr[64]; /*Link address of IPv6,at most 1 link address*/
    int NumSiteAddr; /*Number of site address of IPv6*/
    char SiteAddr[4][64]; /*Site address of IPv6,at most 4 site
addresses*/
    int NumGlobalAddr; /*Number of global address of IPv6*/
    char GlobalAddr[8][64]; /*Global address of IPv6,at most 8
addresses*/
}
```

```

addresses*/
    int NumDns;           /*Number of DNS*/
    char Dns[2][64];      /*DNS address*/
}

```

Structure: struct IPv6RouteTable

Get all IPv6 route information in OS:

IPv6RouteTable

```

struct IPv6RouteTable{
    Char RouteNum;          /* The number of items in the routing table
*/
    Struct {
        char DestAddr[64];      /*Prefix of destination address*/
        char DestAddrPrefixLen;  /*Not supported*/
        char NextHop[64];       /*Next hop address*/
    }RouteTable[16];
}

```

Structure: struct DnsAddrInfo

Save the result of domain name resolution:

DnsAddrInfo

```

struct DnsAddrInfo{
    int NumIPv4Addr;        /*numbers of IPv4 addr*/
    char IPv4Addr[4][16];   /*IPv4 addr*/
    int NumIPv6Addr;        /*numbers of IPv6 addr*/
    char IPv6Addr[4][64];   /*IPv6 addr*/
};

```

Structure: IPv4RouteTable

Get operating system IPv4 route table:

IPv4RouteTable

```

struct IPv4RouteTable{
    int RouteNum; /* number of Route*/
    struct{
        char DestAddr[16];/* dest addr*/

```

```

char GenMask[16]; /* gen mask */
char GateWay[16]; /* Gateway*/
}RouteTable4[16];
}

```

20.3 IPv4 Network Configuration Interface

The following interfaces only apply to IPv4 network environment.

20.3.1 OsNetAddArp

Prototype	int OsNetAddArp(int NetLink, const char *Addr, const char MAC[6]);	
Function	Add IP address to static mapping table of MAC address.	
Parameters	NetLink	Physical channel: <ul style="list-style-type: none">▪ NET_LINK_ETH: Ethernet;▪ NET_LINK_WIFI: WiFi;▪ NET_LINK_USB: USB
	Addr[Input]	IP address. The format is “XXX.XXX.XXX.XXX”. The value range of XXX is from 0 to 255. e.g.: “192.168.0.1”. It can't be NULL.
	MAC[Input]	The MAC address corresponding to the IP address; The size of MAC is 6 bytes. It can't be NULL.
Return	RET_OK	Succeeded.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_NET_IF	The network link cannot be used (not established or disconnected).
Instruction		

NOTE


1. Static ARP table is used to resist the attack from ARP Cheat.
2. Static ARP table is obtained dynamically, so application doesn't need to configure it.

20.3.2 OsNetPing

Prototype	int OsNetPing(const char *Addr, int TimeOutMs);	
Function	Send ICMP ECHO_REQUEST request to specified network host and check whether the host can be reachable.	
Parameters	Addr[Input]	The IP address of target device; Format is “XXX.XXX.XXX.XXX”; The value range of XXX is from 0 to 255, e.g.: “192.168.0.3”.
	TimeOutMs	Timeout.[unit:ms] The valid timeout ranges from 3000 to 3600000.
Return	RET_OK	Succeeded.
	ERR_NET_IF	The network link cannot be used (not established or disconnected).
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_TIME_OUT	Timeout.
Instruction	<ol style="list-style-type: none"> 1. The communication is performed with default route. 2. Call OsNetGetRoute() to query the current default route. 	

20.3.3 OsNetPingEx

Prototype	int OsNetPingEx(const char *Addr, int TimeOutMs, int Size);	
Function	Ping a specific IP address to check the status of the network connection.	
Parameters	Addr[Input]	IP address of the target device. The format is “XXX.XXX.XXX.XXX”. The value range of XXX is 0~255, e.g.: “192.168.0.3”.
	TimeOutMs [Input]	Timeout[unit:ms] The valid timeout ranges from 3000 to 3600000.
	Size [Input]	The size of the data The valid data size is not more than 1024 bytes.

Return	>0 Succeeded, the time it takes to ping. ERR_NET_IF The network link cannot be used (not established or disconnected). ERR_INVALID_PARAM Invalid parameter. ERR_TIME_OUT Timeout.
Instruction	1. The communication is performed with default route. 2. Call OsNetGetRoute() to query the current route.

20.3.4 OsNetDns

Prototype	<code>int OsNetDns(const char *Name, char *Addr, int TimeOutMs);</code>	
Function	Analyze the IP address corresponding to the domain name and store the results in structure <i>struct DnsAddrInfo</i> .	
Parameters	Name[Input] The valid Name is not more than 255 characters and it can't be NULL.	Domain name, for example: [www.sina.com.cn]. The valid Name is not more than 255 characters and it can't be NULL.
	Addr[Output]	Addr stores the IP address mapped by the domain name. The format is “XXX.XXX.XXX.XXX”. The value range of XXX is from 0 to 255, e.g.: “192.168.0.3”. The valid length of Addr should be at least 16 bytes and it can't be NULL.
	TimeOutMs	Timeout[unit:ms]. The valid timeout ranges from 1000 to 3600000.
Return	RET_OK ERR_NET_IF ERR_INVALID_PARAM ERR_TIME_OUT ERR_NET_DNS	Succeeded. Network link cannot be used (not established or disconnected). Invalid parameter Timeout. Domain name resolution failed, and corresponding domain name may not exist.
Instruction	1. The communication is performed with default route.	

- | | |
|--|---|
| | 2. Call OsNetGetRoute() to query the current route. |
|--|---|

20.3.5 OsNetDnsEx

Prototype	int OsNetDnsEx(const char *Name, struct DnsAddrInfo *Addr, int TimeOutMs);															
Function	IP address of the corresponding domain name resolution, which will be saved in structure DnsAddrInfo .															
Parameters	Name[Input]	Domain name, for example: [www.sina.com.cn]. The valid Name is not more than 255 characters and it can't be NULL.														
	Addr[Output]	Store the IP address mapped by the domain name.														
	TimeOutMs [Input]	Timeout[unit: ms]the valid value ranges from 1000 to 3,600,000.														
Return	<table> <tr> <td>RET_OK</td> <td>Success</td> </tr> <tr> <td>ERR_NET_IF</td> <td>Network link cannot be used (not established or disconnected).</td> </tr> <tr> <td>ERR_INVALID_PARAM</td> <td>Invalid parameter</td> </tr> <tr> <td>ERR_TIME_OUT</td> <td>Timeout</td> </tr> <tr> <td>ERR_NET_DNS</td> <td>Domain name resolution failed, and corresponding domain name may not exist.</td> </tr> <tr> <td>ERR_NET_SERVER_BA D</td> <td>IP server exception.</td> </tr> <tr> <td>ERR_NET_NODNServer</td> <td>The domain name server is not configured.</td> </tr> </table>		RET_OK	Success	ERR_NET_IF	Network link cannot be used (not established or disconnected).	ERR_INVALID_PARAM	Invalid parameter	ERR_TIME_OUT	Timeout	ERR_NET_DNS	Domain name resolution failed, and corresponding domain name may not exist.	ERR_NET_SERVER_BA D	IP server exception.	ERR_NET_NODNServer	The domain name server is not configured.
RET_OK	Success															
ERR_NET_IF	Network link cannot be used (not established or disconnected).															
ERR_INVALID_PARAM	Invalid parameter															
ERR_TIME_OUT	Timeout															
ERR_NET_DNS	Domain name resolution failed, and corresponding domain name may not exist.															
ERR_NET_SERVER_BA D	IP server exception.															
ERR_NET_NODNServer	The domain name server is not configured.															
Instruction	<ol style="list-style-type: none"> 1. The default router has been adopted to do the communication, OsNetGetRoute() can be called to check the current default router; 2. OsNetDnsEx() supports IPv4 and IPv6 domain name resolution; 3. OsNetDnsEx() can return a maximum number of 4 IPv4 addresses and 4 IPv6 addresses for one domain name; 4. It is recommended to call OsNetDnsEx() to do the domain name resolution instead of OsNetDns(). 															

20.3.6 OsNetSetConfig

Prototype	<pre>int OsNetSetConfig(int NetLink, const char *Addr, const char *Mask, const char *Gateway, const char *DNSServer);</pre>	
Function	Configure the local network.	
Parameters	NetLink	<p>Physical channel:</p> <ul style="list-style-type: none"> ▪ NET_LINK_ETH: Ethernet; ▪ NET_LINK_WIFI: WiFi; ▪ NET_LINK_BT: Bluetooth; ▪ NET_LINK_USB: USB.
	Addr[Input]	<p>POS terminal address. The format is “XXX.XXX.XXX.XXX”. The value range of XXX is from 0 to 255, e.g.: “192.168.0.3”. If the address is “” or NULL, the previous configuration will remain unchanged.</p>
	Mask[Input]	<p>POS terminal mask code. Format is “XXX.XXX.XXX.XXX”; e.g.: “255.255.255.0”; If the mask code is “” or NULL, the previous configuration will remain unchanged.</p>
	Gateway[Input]	<p>Address of the default gateway. Format is “XXX.XXX.XXX.XXX”. XXX ranges from 0 to 255, e.g.: “192.168.0.1”. If the address is “” or NULL, the previous configuration will remain unchanged.</p>
	DNSServer[Input]	<p>DNS server address Format is “XXX.XXX.XXX.XXX”. XXX ranges from 0 to 255. e.g.: “192.168.0.1”; If the address is “” or NULL, the previous configuration will remain unchanged.</p>

Return	RET_OK Succeeded ERR_NET_IF Network link cannot be used (not established or disconnected). ERR_INVALID_PARAMETER Invalid parameter.
Instruction	<ol style="list-style-type: none"> 1. The DHCP function will be stopped after calling this function successfully. 2. The function only checks the validity of Addr, Mask and Gateway's formats without checking the matching relation between them. 3. Wireless link, PPPoE, and ModemPPP is dynamically allocated and cannot be configured by this interface. 4. After configuring network information successfully, this link will be set as the default route.

20.3.7 OsNetGetConfig

Prototype	int OsNetGetConfig(int NetLink, char *Addr, char *Mask, char *Gateway, char *DNSServer);	
Function	Get the network information.	
Parameters	NetLink	Physical channel: <ul style="list-style-type: none"> • NET_LINK_ETH: Ethernet; • NET_LINK_WIFI: WiFi. • NET_LINK_BT: Bluetooth; • NET_LINK_USB: USB.
	Addr[Output]	POS terminal address. The size of Addr should be at least 16 bytes. The format is "XXX.XXX.XXX.XXX", XXX ranges from 0 to 255, e.g.: "192.168.0.3" and it can be NULL.
	Mask[Output]	POS terminal mask code. The size should beat least 16 bytes. Format is "XXX.XXX.XXX.XXX", e.g.: "255.255.255.0" and it can be NULL.
	Gateway[Output]	Address of the default gateway. The size of Gateway should be at least 16 bytes. Format is "XXX.XXX.XXX.XXX". XXX ranges from 0 to 255, e.g.: "192.168.0.1" and it

		can be NULL.
	DNSServer[In put]	DNS server address. The size of DNSServer should be at least 16 bytes. Format is “XXX.XXX.XXX.XXX”, XXX ranges from 0 to 255. e.g.: “192.168.0.1” and it can be NULL.
Return	RET_OK	Succeeded.
	ERR_NET_IF	Network link cannot be used (not established or disconnected).
Instruction		

NOTE

If Addr, Mask, Gateway and DNSServer returns the string "", it means the network has not been configured.

20.3.8 OsNetStartDhcp

Prototype	int OsNetStartDhcp(int NetLink);	
Function	Start DHCP to obtain dynamic address.	
Parameters	NetLink	Physical channel: • NET_LINK_ETH: Ethernet; • NET_LINK_WIFI: WiFi; • NET_LINK_BT: Bluetooth; • NET_LINK_USB: USB
Return	RET_OK	Succeeded.
	ERR_NET_IF	Ethernet or WiFi module is not configured.
Instruction	<ol style="list-style-type: none"> This interface is just used to start the DHCP and will not wait for the address allocation process. OsNetCheckDhcp() can be called to check the status of address allocation. After obtaining the address successfully, this link will be set as the default route. 	



CAUTION
Please close all the connections before starting DHCP; otherwise, these connections may fail to communicate.

20.3.9 OsNetCheckDhcp

Prototype	int OsNetCheckDhcp(int NetLink);	
Function	Check DHCP status.	
Parameters	NetLink	Physical channel: <ul style="list-style-type: none"> ▪ NET_LINK_ETH: Ethernet; ▪ NET_LINK_WIFI: WiFi; ▪ NET_LINK_BT: Bluetooth; ▪ NET_LINK_USB: USB
Return	NET_DOING RET_OK ERR_NET_DHCP_DISC OVER ERR_NET_DHCP_OFFE R ERR_NET_DHCP_STAR T	NET_DOING DHCP is getting the address. RET_OK Dynamic allocation succeeded. ERR_NET_DHCP_DISC OVER DHCP Server is not found. ERR_NET_DHCP_OFFE R DHCP cannot get IP address. ERR_NET_DHCP_STAR T DHCP is not started.
Instruction		

20.3.10 OsNetStopDhcp

Prototype	void OsNetStopDhcp(int NetLink);	
Function	Stop DHCP.	
Parameters	NetLink	Physical channel: <ul style="list-style-type: none"> ▪ NET_LINK_ETH: Ethernet; ▪ NET_LINK_WIFI: WiFi; ▪ NET_LINK_BT: Bluetooth; ▪ NET_LINK_USB: USB
Return	None.	
Instruction	<ol style="list-style-type: none"> 1. If the DHCP is stopped, OsNetSetConfig() shall be called to re-configure the network; 2. After a successful DHCP, the system will update the configuration information at regular interval; if the update fails, the network will be unavailable. 	

20.3.11 OsPppoeLogin

Prototype	<code>int OsPppoeLogin(const char *Name, const char *Password, int TimeOutMs);</code>	
Function	PPPoE link login.	
Parameters	Name[Input]	User name. The valid length ranges from 0 to 50 bytes. It can't be NULL. If there is no user name, a null string "" can be used instead.
	Password[Input]	Password. The valid length ranges from 0 to 50 bytes. It can't be NULL. If there is no password, a null string "" can be used instead.
	TimeOutMs	Timeout.[unit:ms] The valid timeout ranges from 0 to 3600000.
Return	NET_DOING	Logging in/out.
	RET_OK	Succeeded.
	<0	Failed.
Instruction	After established successfully, this link will be set as the default route.	

20.3.12 OsPppoeCheck

Prototype	<code>int OsPppoeCheck(void);</code>	
Function	Check the status of PPPoE link.	
Parameters	None.	
	NET_DOING	Logging in/out.
	RET_OK	The link is established successfully.
	<0	The link is disconnected.
Instruction		

20.3.13 OsPppoeLogout

Prototype	<code>void OsPppoeLogout(void);</code>	
Function	Disconnect the PPPoE link.	
Parameters	None.	

Return	None.
Instruction	

20.3.14 OsNetSetRoute

Prototype	int OsNetSetRoute(int NetLink);	
Function	Set system default route.	
Parameters	NetLink	Physical channel, please refer to Physical channel list
Return	RET_OK ERR_INVALID_PARAM ERR_NET_IF	Succeeded, the new route came into effect. Invalid parameter. This link has not been set up.
Instruction	<ol style="list-style-type: none"> It is not necessary to call this function to set the default route if there is only one link. If there are several links, this function can be called to switch the default route. It is allowed to switch default only when the network link has been set up successfully, otherwise ERR_NET_IF will be returned. Switching the default route during communication process will cause communication failure. 	

20.3.15 OsNetGetRoute

Prototype	int OsNetGetRoute(void);	
Function	Get the default route.	
Parameters	None	
Return	>0 ERR_NET_NO_ROUTE	Physical channel, please refer to Physical channel list . No default route.
Instruction	<p>In following situations, ERR_NET_NO_ROUTE will be returned.</p> <ol style="list-style-type: none"> No link has been set up. The last used link is logged out, as any link that has been set up successfully will be configured as the default route. If the last used link is logged out, the route will be deleted. At this time, OsNetSetRoute() shall be called to set route again. 	

20.3.16 OsNetSetRouteTable

Prototype	int OsNetSetRouteTable(int NetLink, const char *DestAddr, const char *GenMask, const char *GateWay);										
Function	Add IPv4 static address to the route table.										
Parameters	<table> <tr> <td>NetLink</td><td>Physical channel, please refer to Physical channel list.</td></tr> <tr> <td>DestAddr</td><td>Destination network address</td></tr> <tr> <td>GenMask</td><td>Mask address of destination network</td></tr> <tr> <td>GateWay</td><td>Gateway address</td></tr> </table>	NetLink	Physical channel, please refer to Physical channel list .	DestAddr	Destination network address	GenMask	Mask address of destination network	GateWay	Gateway address		
NetLink	Physical channel, please refer to Physical channel list .										
DestAddr	Destination network address										
GenMask	Mask address of destination network										
GateWay	Gateway address										
Return	<table> <tr> <td>RET_OK</td><td>Success</td></tr> <tr> <td>ERR_INVALID_PARAMETER</td><td>Invalid parameter</td></tr> <tr> <td>ERR_NET_SERVER_BAD</td><td>IP server exception</td></tr> <tr> <td>ERR_NET_SERVER_BUSY</td><td>IP server busy</td></tr> <tr> <td>ERR_ROUTE_EXIST</td><td>Route already exists</td></tr> </table>	RET_OK	Success	ERR_INVALID_PARAMETER	Invalid parameter	ERR_NET_SERVER_BAD	IP server exception	ERR_NET_SERVER_BUSY	IP server busy	ERR_ROUTE_EXIST	Route already exists
RET_OK	Success										
ERR_INVALID_PARAMETER	Invalid parameter										
ERR_NET_SERVER_BAD	IP server exception										
ERR_NET_SERVER_BUSY	IP server busy										
ERR_ROUTE_EXIST	Route already exists										
Instruction	<ol style="list-style-type: none"> Calling this function will add IPv4 static address to the route table. Normally, the network communication data will go through the default route, there is no need to add the route information manually. For Prolin system, only one default route is allowed. When there are multiple network cards, application may need to call OsNetSetRouteTable() add the static route to the specified link to make sure each link can communicate with the external network at the same time. When application calls this function to add static route, the system will only check the validity of each parameter, to ensure the communication will not have exception, application needs to check that the inputting parameters actually apply to the current network. 										

20.3.17 OsNetDelRouteTable

Prototype	int OsNetDelRouteTable(int NetLink, const char *DestAddr, const char *GenMask);
-----------	--

Function	Delete the specified IPv4 address from the router.	
Parameters	NetLink	Physical channel, please refer to Physical channel list .
	DestAddr	Destination network address
	GenMask	Mask of destination address
Return	RET_OK	Success
	ERR_INVALID_PARAM	Invalid parameter
	ERR_NET_SERVER_BAD	IP server exception
	ERR_NET_SERVER_BUSY	IP server busy
	ERR_ROUTE_NONEEXIST	Route doesn't exist
Instruction	When application calls this function to delete a certain route, system will only check the validity of each parameter, application also needs to ensure that the current communication will not be affected after the specified route has been deleted.	

20.3.18 OsNetGetRouteTable

Prototype	<code>int OsNetGetRouteTable(int NetLink, struct IPv4RouteTable *RouteTable);</code>	
Function	Acquire the specified IPv4 route.	
Parameters	NetLink	Physical channel, please refer to Physical channel list .
	RouteTable	Structure that is used to receive the IPv4 route table information: IPv4RouteTable .
Return	RET_OK	Success
	ERR_INVALID_PARAM	Invalid parameter
	ERR_NET_SERVER_BAD	IP server exception.
	ERR_NET_SERVER_BUSY	IP server is busy
	ERR_SYS_BAD	System error
	ERR_MEMFAULT	Memory error

	ERR_NET_SESS_B ROKEN	Connection error
Instruction	Application can call OsNetGetRouteTable() to acquire the route information of IPv4 address.	

20.3.19 OsNetNat

Prototype	int OsNetNat(int DestLink);	
Function	Set up network data forwarding.	
Parameters	DestLink [Input]	A link used to forward packets out. The value of DestLink can refer to the definition of physical channel list; When DestLink is 0, the data will be forwarded from the link where the default route of the current system is located. To avoid forwarding failure due to routing table configuration problems, it is recommended to set the data to be forwarded from the link where the default route is located.
Return	RET_OK ERR_NET_IF ERR_NET_SERVER_BAD	Succeeded This link has not been set up. IP server exception
Instruction	<ol style="list-style-type: none"> Before configuring forwarding, please ensure that the network parameters set in the forwarding device are normal, and the forwarding device can communicate with the destination host which the packet sent by the forwarded device normally; When forwarding data in multi-link devices, please configure a reasonable routing table, otherwise the data may not be forwarded normally due to improper configuration of the routing table; After the application successfully calls this function, the system will forward the non-native packets in the destination address received through DestLink link; This function can be called multiple times in the forwarding device to set up different forwarding links, so that different packets in the destination host can be forwarded from different links together with reasonable routing table configuration. 	

20.4 IPv4/IPv6 Network Configuration Interface

The following interfaces apply to both IPv4 and IPv6 network environment.

20.4.1 OsNetSetDns

Prototype	<code>int OsNetSetDns(const char *Dns);</code>
Function	Set DNS address.
Parameters	Dns[Input] DNS address.
Return	RET_OK Succeeded. ERR_INVALID PARA M Invalid parameter.
Instruction	<ol style="list-style-type: none"> 1. The system can set 2 DNS addresses at most (doesn't distinguish IPv4 or IPv6). If two addresses are needed, users should call this function to set the slave DNS address first, and then call this function to set the master DNS address. 2. The settings will remain valid after a system reboot. 3. DNS is stored in <code>/etc/resolv.conf</code>, through which applications can check the current DNS. 4. By default, if DHCP is enabled, DNS server address set by this function may fail, and the DNS server address set by this function may be overridden by the address of DHCP DNS server. If DNS server address set by this function is needed when DHCP is enabled, users can call <code>OsRegSetValue()</code> to set "persist.sys.dns.static" to "1", and then call this function to set static DNS server address, and enable DHCP in the end.

20.5 IPv6 Network Configuration

The following interfaces only apply to IPv6 network environment.

20.5.1 OsNetPing6

Prototype	<code>int OsNetPing6(const char *DestAddr, const char *SrcAddr, int TimeOutMs, int size);</code>
Function	Send request message ICMPv6 ECHO_REQUEST to the specified network host.

Parameters	DestAddr <input/>	IPv6 address of the target network host. E.g.: "2015:10::5".
	SrcAddr <input/>	Source IPv6 address. E.g.: "2015:10::6". SrcAddr is the default address allocated by the system when it is NULL.
	TimeOutMs	Timeout[unit: ms] The valid timeout ranges from 3,000 to 3,600,000.
	size	The load size of the request message. The valid size is not more than 1,500 bytes.
Return	>=0 Succeeded. Time spent by ping(unit: ms).	
	ERR_NET_IF	The link is not established or is disconnected.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_TIME_OUT	Timeout.
	ERR_NET_ICMPV6_PACKET_TOOBIG	SIZE is greater than the minimum MTU.
	ERR_NET_ICMPV6_UNREACH	Target is unreachable.
	ERR_NET_ICMPV6_TTL_EXCEED	TTL exceeds the maximum value (TTL default value is 64).
	ERR_NET_ICMPV6_PROTOCOLPROB	Protocol parameter error. Cannot process the middle route or destination node.
Instruction	For example: <i>int ret;</i> <i>ret=OsNetPing6("2015:10::5","2015:10::6",9000,56);</i>	

20.5.2 OsNetSetIPv6Addr

Prototype	int OsNetSetIPv6Addr(int NetLink, const char *Addr, int PrefixLength);	
Function	Set static IPV6 address.	
Parameters	NetLink	Physical channel: • NET_LINK_ETH: Ethernet.
	Addr	IPv6 address.

	[Input]	E.g.: “2015:30::10”.
	PrefixLength [Input]	Length of the IPv6 address prefix. The valid length ranges from 1 to 128 bits. The PrefixLength will be set to 64 by default if it is 0.
Return	RET_OK	Succeeded.
	ERR_NET_IF	The link is not established or is disconnected.
	ERR_NET_DHCP6_DUPPLICATE_ADDR	The setting address is in conflict with a certain node address in local area network, setting address failed.
	ERR_INVALID_PARAM	Invalid parameter.
Instruction	<ol style="list-style-type: none"> 1. The static IPV6 address doesn't need to be set manually; it can be obtained automatically from the router by terminal or obtained by DHCPv6 server after starting DHCPv6. 2. All static IPv6 addresses will be cleared when device is powered down. 3. Prolin supports link address, site address and global address. And each kind of address can only be set for one address. For example, if a global address has been set, the new setting will replace the original global address. 	

20.5.3 OsNetGetIPv6Addr

Prototype	<code>int OsNetGetIPv6Addr(int NetLink, struct IPv6Info *AddrInfo);</code>	
Function	Get IPv6 and DNS address of specified physical channel.	
Parameters	NetLink	Physical channel: • NET_LINK_ETH: Ethernet.
	AddrInfo [Output]	Output classified IPv6 and DNS address, IPv6Info structure.
Return	RET_OK	Succeeded.
	ERR_NET_IF	The link is not established or is disconnected.
	ERR_INVALID_PARAM	Invalid parameter.
Instruction	<p>For example:</p> <pre>struct IPv6Info AddrInfo; ret = OsNetGetIPv6Addr(NET_LINK_ETH,&AddrInfo);</pre>	

20.5.4 OsNetGetRouteAdvertise6

Prototype	int OsNetGetRouteAdvertise6(int NetLink, int TimeOutMs, int *RouteMode);	
Function	Get the working mode of router within specified time.	
	NetLink	Physical channel: NET_LINK_ETH: Ethernet.
	TimeOutMs [Input]	Waiting time for router to respond. (Unit: ms) The valid timeout ranges from 1,000 to 5,000. If the value <1,000, it will be set to 1,000; If the value> 5,000, it will be set to 5,000.
Parameters	RouteMode [Output]	Working modes of router: 1. IP6_STATELESS_ADDR_AUTOCONFIG: Stateless address auto-configuration. 2.IP6_DHCP6_STATEFULL_ADDR_CONFIG: DHCPv6 stateful address configuration. 3.IP6_DHCP6_STATELESS_ADDR_CONFIG: DHCPv6 stateless address configuration.
Return	RET_OK	Succeeded.
	ERR_TIME_OUT	Timeout; have no response.
Instruction	<p>1. Stateless address auto-configuration: IPV6 address is automatically generated according to the address prefix assigned by router RA; but DNS needs to be set manually.</p> <p>2. DHCPv6 stateful address configuration: get IPv6 address and DNS address.</p> <p>3. DHCPv6 stateless address configuration: get DNS information.</p> <p>For example:</p> <pre>int ret,route_mode; ret=OsNetGetRouteAdvertise6(NET_LINK_ETH,3000,&route_mode); if(ret == RET_OK) { switch (route_mode) {</pre>	

```

Case IP6_STATELESS_ADDR_AUTOCONFIG:
    break; /*do not need to call DHCPv6*/
Case IP6_DHCP6_STATEFULL_ADDR_CONFIG :
    ret=OsNetStartDhcp6(NET_LINK_ETH,IP6_DHC
        P6_STATEFULL_ADDR_CONFIG);
    if(ret != RET_OK)
    {.....};
    break;
Case IP6_DHCP6_STATELESS_ADDR_CONFIG:
    ret=OsNetStartDhcp6(NET_LINK_ETH,IP6_DHC
        P6_STATELESS_ADDR_CONFIG);
    if(ret != RET_OK)
    {.....};
    break;
}
}
.....

```

20.5.5 OsNetStartDhcp6

Prototype	int OsNetStartDhcp6(int NetLink, int state);	
Function	Start DHCPv6 function to get the dynamically assigned IPv6 address and DNS address.	
Parameters	NetLink	Physical channel: • NET_LINK_ETH: Ethernet.
	state [Input]	IP6_DHCP6_STATEFULL_ADDR_CONFIG : DHCPv6 stateful address configuration. IP6_DHCP6_STATELESS_ADDR_CONFIG : DHCPv6 stateless address configuration.
Return	RET_OK	Succeeded.
	NET_DOING	Repetitive execution.
	ERR_NET_IF	The link cannot be used (not established or disconnected).
	ERR_INVALID_PARAM	Invalid parameter (<i>AddrInfo</i> is NULL).

Instruction	<ol style="list-style-type: none"> 1. DHCPv6 stateful address configuration mode can get the IPv6 address and DNS information. 2. DHCPv6 stateless address mode can only get DNS information. 3. OsNetGetRouteAdvertise6() shall be called first to select the type of DHCPv6.
-------------	---

20.5.6 OsNetCheckDhcp6

Prototype	int OsNetCheckDhcp6(int NetLink);																		
Function	Get the current working status of DHCPv6.																		
Parameters	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%;">NetLink</td> <td>Physical channel: • NET_LINK_ETH: Ethernet.</td> </tr> </table>	NetLink	Physical channel: • NET_LINK_ETH: Ethernet.																
NetLink	Physical channel: • NET_LINK_ETH: Ethernet.																		
Return	<table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">RET_OK</td> <td>Succeeded.</td> </tr> <tr> <td>ERR_NET_IF</td> <td>The link cannot be used (not established properly or disconnected)</td> </tr> <tr> <td>NET_DOING</td> <td>In process of assigning IPv6 and DNS address.</td> </tr> <tr> <td>ERR_NET_DHCP_START</td> <td>OsNetStartDhcp6() is not executed.</td> </tr> <tr> <td>ERR_NET_DHCP6_SOLICITE</td> <td>DHCPv6 server has no response</td> </tr> <tr> <td>ERR_NET_DHCP6_REQUEST</td> <td>DHCPv6 server has not responded correctly to REQUEST command.</td> </tr> <tr> <td>ERR_NET_DHCP6_INFORMATION</td> <td>DHCPv6 server has not responded correctly to INFORMATION REQUEST command.</td> </tr> <tr> <td>ERR_NET_DHCP6_VALID_LIFE</td> <td>Only DHCPv6 stateful mode can be returned: the valid life time has been expired, and the IPv6 address, for which previously applied has been released.</td> </tr> <tr> <td>ERR_NET_DHCP6_NOADDRSAVAIL</td> <td>DHCPv6 server has no available address to assign.</td> </tr> </table>	RET_OK	Succeeded.	ERR_NET_IF	The link cannot be used (not established properly or disconnected)	NET_DOING	In process of assigning IPv6 and DNS address.	ERR_NET_DHCP_START	OsNetStartDhcp6() is not executed.	ERR_NET_DHCP6_SOLICITE	DHCPv6 server has no response	ERR_NET_DHCP6_REQUEST	DHCPv6 server has not responded correctly to REQUEST command.	ERR_NET_DHCP6_INFORMATION	DHCPv6 server has not responded correctly to INFORMATION REQUEST command.	ERR_NET_DHCP6_VALID_LIFE	Only DHCPv6 stateful mode can be returned: the valid life time has been expired, and the IPv6 address, for which previously applied has been released.	ERR_NET_DHCP6_NOADDRSAVAIL	DHCPv6 server has no available address to assign.
RET_OK	Succeeded.																		
ERR_NET_IF	The link cannot be used (not established properly or disconnected)																		
NET_DOING	In process of assigning IPv6 and DNS address.																		
ERR_NET_DHCP_START	OsNetStartDhcp6() is not executed.																		
ERR_NET_DHCP6_SOLICITE	DHCPv6 server has no response																		
ERR_NET_DHCP6_REQUEST	DHCPv6 server has not responded correctly to REQUEST command.																		
ERR_NET_DHCP6_INFORMATION	DHCPv6 server has not responded correctly to INFORMATION REQUEST command.																		
ERR_NET_DHCP6_VALID_LIFE	Only DHCPv6 stateful mode can be returned: the valid life time has been expired, and the IPv6 address, for which previously applied has been released.																		
ERR_NET_DHCP6_NOADDRSAVAIL	DHCPv6 server has no available address to assign.																		
Instruction	<ol style="list-style-type: none"> 1. DHCPv6 stateful mode can get the IPv6 and DNS address. 2. DHCPv6 stateless working mode can only get DNS address. 																		

20.5.7 OsNetStopDhcp6

Prototype	int OsNetStopDhcp6(int NetLink);
-----------	---

Function	Stop DHCPv6 server from running.	
Parameters	NetLink	Physical channel: • NET_LINK_ETH: Ethernet.
Return	RET_OK ERR_NET_IF	Succeeded. The link cannot be used.
Instruction	The existing IPV6 and DNS address are reserved. The DNS will not be renewed, and address will be invalid after restarting.	

20.5.8 OsNetSetRouteTable6

Prototype	<code>int OsNetSetRouteTable6(int NetLink, const char *DestAddr, int DestAddrPrefixLength, const char *NextHop);</code>	
Function	Set IPv6 static routing table	
Parameters	NetLink	Physical link: • NET_LINK_ETH Ethernet
	DestAddr[Input]	Destination address: E.g. : "2001:20::"; "::".
	DestAddrPrefixLength [Input]	The length of prefix destination address, ranging from 0 to 128.
	NextHop[Input]	Gateway address: E.g.: "fe80:9::23"; "::".
Return	RET_OK ERR_NET_IF ERR_INVALID_PARAM	Succeeded The link is not established or is disconnected. Invalid parameter
Instruction	<p>For example:</p> <pre>int ret; ret=OsNetSetRouteTable6(NET_LINK_ETH,"2016::",128, fe80:99::99"); 1. The setting addresses will be all cleared if the device powers down. 2. The terminal will automatically acquire the static routing table from the router without any manual setting.</pre>	

20.5.9 OsNetGetRouteTable6

Prototype	int OsNetGetRouteTable6(int NetLink, struct IPv6RouteTable *RoutingTable);	
Function	Get the information of specified routing table.	
Parameters	NetLink	Physical link: • NET_LINK_ETH Ethernet
	RoutingTable[Output]	IPv6RouteTable structure
Return	RET_OK	Succeeded
	ERR_NET_IF	The link is not established or is disconnected.
	ERR_INVALID_PARAM	Invalid parameter
Instruction	<p>Example:</p> <pre>struct IPv6RouteTable routing_table; ret= OsNetGetRouteTable6(NET_LINK_ETH,&routing_table); The acquired content doesn't include network interface "lo".</pre>	

21GPRS/CDMA

Prolin supports GPRS and CDMA network and provides a series of related APIs for developers.

21.1 Return Code List

Table 21.1 GPRS/CDMA return code list

Macro	Value	Description
PPP_LOGINING	1	<i>PPP is logging in.</i>
PPP_LOGOUTING	2	<i>PPP is logging out.</i>
WL_CSD_READY	3	<i>CSD dialup service is ready</i>
WL_GPRS_CSD_READY	4	<i>GPRS and CSD dialup service are ready.</i>
ERR_WL_POWER_ONING	-3501	<i>In the process of powering on the wireless module.</i>
ERR_WL_POWER_OFF	-3502	<i>Wireless module is powered off.</i>
ERR_WL_NOT_INIT	-3503	<i>Module fails to be initialized and cannot work normally.</i>

ERR_WL_NEEDPIN	-3504	<i>SIM card needs PIN.</i>
ERR_WL_RSPERR	-3505	<i>Module response error.</i>
ERR_WL_NORSP	-3506	<i>Module has no response.</i>
ERR_WL_NEEDPUK	-3507	<i>SIM card needs PUK.</i>
ERR_WL_WRONG_PIN	-3508	<i>PIN error of SIM card.</i>
ERR_WL_NOSIM	-3509	<i>No SIM card.</i>
ERR_WL_NOREG	-3510	<i>Cannot register network.</i>
ERR_WL_AUTO_RST	-3511	<i>Automatic reset.</i>
ERR_WL_BUF	-3512	<i>Module memory error.</i>
ERR_WL_GET_SIGNAL	-3513	<i>Module is getting signal, please wait 3s.</i>
ERR_WL_NOTYPE	-3514	<i>Unrecognisable module.</i>
ERR_WL_PPP_ONLINE	-3515	<i>PPP is on line; it cannot sleep.</i>
ERR_WL_ERR_BUSY	-3516	<i>Module is busy.</i>
ERR_WL_SLEEP_ONING	-3517	<i>Module is entering sleep mode.</i>
ERR_WL_SLEEP_FAIL	-3518	<i>Fail to sleep.</i>
ERR_WL_SIM_FAILURE	-3519	<i>Fail to operate SIM card.</i>

21.2 Wireless Module Interface

21.2.1 OsWILock

Prototype	int OsWILock(void);	
Function	Open wireless module, set up a link with Prolin wireless server and acquire the permission to wireless device/module.	
Parameters	None.	
Return	RET_OK	Succeeded.
	ERR_DEV_BUSY	Module/device is busy.
	ERR_DEV_NOT_EXIST	Module/device does not exist.
	ERR_BATTERY_VOLTAGE_TOO_LOW	Battery voltage is too low.
	ERR_BATTERY_ABSENT	Battery is absent.
Instruction	1. This function must be called to open the wireless module	

	<p>first before calling OsWIInit(), OsWIPowerSwitch(), OsWILogin() or OsWILogout().</p> <ol style="list-style-type: none"> 2. OsWIUnLock() shall be called to close wireless module after finishing the operation. 3. S920 and D200 mobile terminals need to be powered by battery; otherwise, ERR_BATTERY_ABSENT will be returned. 4. When the battery voltage is 0, module cannot work normally and ERR_BASENT_VOLTAGE_TOO_LOW will be returned.
--	---

21.2.2 OsWIUnLock

Prototype	void OsWIUnLock(void);	
Function	Release the usage rights of wireless device and disconnect from Prolin wireless service.	
Parameters	None.	
Return	None.	
Instruction		

21.2.3 OsWIInit

Prototype	int OsWIInit(const char *SimPin);	
Function	Initialize wireless device.	
Parameters	SimPin[Input]	A pointer to SIM card PIN. The valid string length is less than 50 bytes. It can be NULL, which means it doesn't need any password.
Return	RET_OK ERR_DEV_NOT_OPEN ERR_DEV_NOT_EXIST ERR_NO_PORT ERR_WL_NEEDPIN ERR_WL_RSPERR ERR_WL_NORSP ERR_WL_NEEDPUK ERR_WL_WRONG_PIN ERR_WL_NOSIM	Succeeded. Module/device is not open. Module/device does not exist. Physical serial port is not enough. SIM card needs PIN. Module/device response error. No response. SIM card needs PUK. PIN error. No SIM card.

	ERR_WL_NOTYPE	Unrecognizable module.
	ERR_WL_NOREG	Fail to register GPRS network.
Instruction	<ol style="list-style-type: none"> 1. OsWILOCK() needs to be called successfully before calling this function. 2. The password is automatically checked by SIM card. 3. Prolin OS ensures wireless device/module is powered on. 4. When SIM card is absent, OsWInit() returns ERR_WL_NOSIM and the application can only use some functions that do not need SIM card. 5. After calling OsWISwitchPower(), wireless module/device needs around 15 seconds to initialize itself. Application should wait at least 15 seconds until the module becomes stable; otherwise, OsWInit() may fail. 	

21.2.4 OsWInitEx

Prototype	<code>intOsWInitEx(constchar*SimPin, intTimeOutMs, char*CmeString, int Size);</code>	
Function	Initialize wireless device.	
Parameters	SimPin[Input]	A pointer to SIM card PIN. The valid string length is less than 50 bytes. It can be NULL, which means it doesn't need any password.
	TimeOutMs[Input]	Timed out time,[unit:ms]; Valid range is from 25000 to 100000; When the time is less than 25000, it will be set to 25000 automatically; When the time is more than 100000, it will be set to 100000 automatically.
	CmeString[Output]	Reserved. Current value is NULL.
	Size[Input]	Reserved. Current value is 0.
Return	WL_CSD_READY	CSD dialupservice is ready.
	WL_GPRS_CSD_READY	GPRS and CSD dialup service are ready.
	ERR_DEV_NOT_OPEN	Device/ module is not open.
	ERR_DEV_NOT_EXIST	Wireless device/ module does not

	ST	exist.
	ERR_NO_PORT	There's no physics serial port for terminal.
	ERR_WL_NEEDPIN	SIM card needs PIN.
	ERR_WL_RSPERR	Device/ module response error.
	ERR_WL_NORSP	Module has no response.
	ERR_WL_NEEDPUK	SIM card needs PUK.
	ERR_WL_WRONG_PIN	PIN error.
	ERR_WL_NOSIM	There's no SIM card.
	ERR_WL_PPP_ONLINE	PPP online.
	ERR_WL_NOREG	Could not register to the network.
	ERR_INVALID_PARAMETER	Invalid parameter.
Instruction	<ol style="list-style-type: none"> 1. Before calling this function, OsWILOCK() needs to be called successfully; 2. SIM card will check PIN automatically; 3. ProlinOS will ensure wireless device/ module has powered up; 4. When there's no SIM card inside the terminal, application can only use the functions which do not need SIM card; 5. After calling OsWISwitchPower() to power up, application program needs to wait more than 15 seconds to execute OsWIInitEx(), otherwise, it may return failure which caused by unstable module; 6. When ERR_WL_NOREG is returned, application program should stop dialing and other operations. 7. When WL_CSD_READY is returned, it indicates that application program can process CSD service; 8. When WL_GPRS_CSD_READY is returned, it indicates that application program can process GPRS or CSD service; 9. If CSD service is being used now, and GPRS service will be used in the future, OsWIInitEx() needs to be called to reinitialize. 10. When PPP is online, calling OsWIInitEx() will return error. 	

21.2.5 OsWISwitchPower

Prototype	int OsWISwitchPower(int OnOff);	
Function	Power on /off wireless module.	
Parameters	OnOff[Input]	The state of wireless device:

		<ul style="list-style-type: none"> ▪ 1: on ▪ 0: off
Return	RET_OK ERR_DEV_NOT_EXIST ERR_DEV_NOT_OPEN	Succeeded. Wireless module does not exist. Fail to call OsWILOCK().
Instruction	1. Please reboot terminal to reset wireless module in special cases such as always failing to call OsWILOGIN(). 2. Unless it is a special occasion, it is not recommended to call this function. Because after calling this function to power off the wireless module, the wireless network needs to be registered again when powering on the wireless device, which will decrease the success rate of OsWILOGIN().	

- NOTE**
- 
1. The power-on duration depends on wireless module itself.
 2. Power off module will cut off module's power supply.
 3. If a terminal has wireless module/device, the wireless module/device will be powered on by default during the startup of Prolin OS.
 4. The time interval between power-off and power-on again should be at least 8 seconds. If application has not wait long enough and is being powered on right away, it will be blocked long enough first and then be powered on.
 5. Prolin will disconnect ppp link automatically as soon as wireless module/device is powered off.
 6. After wireless module/device is switched from power-off to power-on, Prolin wireless service will be blocked for 15 seconds before responding to operations such as OsWIInit(). The system will wait 15 seconds to work normally to initialize module, get signal and other operations. For example, when performing login operation, the system will wait 15 seconds before logging in, resulting in a long login time.

21.2.6 OsWISwitchSleep

Prototype	int OsWISwitchSleep(int OnOff);	
Function	Set wireless device to be slept or woken up.	
Parameters	OnOff[Input]	<ul style="list-style-type: none"> ▪ 1: Sleep. ▪ 0: Wake up. ▪ Others: Unknown error.

Return	RET_OK	Succeeded.
	ERR_DEV_NOT_EXIST	Module does not exist.
	ERR_DEV_NOT_OPEN	Fail to call OsWLLock().
	ERR_WL_PPP_ONLINE	PPP is online.
Instruction	It is reserved and yet not supported.	

If PPP is on line, wireless module/device will be unable to sleep.

For MG323, it will not sleep in the following cases:



- no SIM card,
- deactivated PIN,
- not registered to network.

21.2.7 OsWLGetSignal

Prototype	int OsWLGetSignal(void);	
Function	Get wireless signal strength.	
Parameters	None.	
Return	0~5	Signal strength. 0 means no signal; 5 means the strongest signal.
	ERR_DEV_NOT_EXIST	Module does not exist.
	ERR_WL_RSPERR	Module response error.
	ERR_WL_POWER_ONLINE	In the process of powering on the module.
Instruction	<ol style="list-style-type: none"> 1. OsWLlock() does not need to be called before calling this function; 2. When the wireless link is not established, the signal values will be directly gotten from module through AT commands. 3. After establishing a wireless link, the module is in the data exchange mode and can't obtain the real-time signal strength. Calling OsWLGetSignal() at this time will return ERR_WL_RSPERR. 	

4. This function shall not be called until OsWIInit() returns RET_OK.

21.2.8 OsWICheck

Prototype	int OsWICheck(void);	
Function	Query the status of wireless link.	
Parameters	None.	
Return	PPP_LOGOUTING	Module is disconnecting link.
	bPPP_LOGINING	Module is establishing link.
	RET_OK	Establish link successfully.
	ERR_DEV_NOT_EXIST	Module does not exist.
	ERR_WL_POWER_ONING	In the process of powering on the module.
	ERR_WL_POWER_OFF	Module is powered off.
	ERR_WL_NOT_INIT	Fail to initialize module.
	ERR_NET_PASSWD	Wrong password.
	ERR_NET_LOGOUT	OsWILogout() disconnects the link.
Instruction		

21.2.9 OsWILogin

Prototype	int OsWILogin(const char *APN, const char *Name, const char *Password, long Auth, int TimeOutMs, int KeepAlive, int ReserParam);	
Function	Login wireless network and set up a wireless link.	
Parameters	APN[Input]	For GPRS communication, APN is access point name. For CDMA, APN is the dialing number. The valid length is from 0 to 50 characters. When it is NULL, the protocol stack will directly login PPP after the application

		dialing up.
	Name[Input]	User name. The valid string length ranges from 0 to 50 characters. It can't be NULL. If there is no user name, a null string "" can be used instead.
	Password[Input]	Password. The valid length ranges from 0 to 50 characters. It can't be NULL. If there is no password, a null string "" can be used instead.
Auth		Authentication algorithms. The system supports the following algorithms:
	PPP_ALG_PAP	0x00000001 PAP authentication algorithm
	PPP_ALG_CHAP	0x00000002 CHAP authentication algorithm
	PPP_ALG_MSCHAPV1	0x00000004 MSCHAPV1 authentication algorithm
	PPP_ALG_MSCHAPV2	0x00000008 MSCHAPV2 authentication algorithm
	PPP_ALG_ALL	0xff All algorithms are supported
		At least one type of authentication or several authentications with (+) or () should be used, for example, PPP_ALG_PAP PPP_ALG_CHAP. If the algorithm is unknown, fill in parameter PPP_ALG_ALL.
	TimeOutMs	Timeout.[unit:ms] The valid timeout ranges from 0 to 3600000.

		If timeout <0, it will be automatically set to 0. If timeout >3600000, it will be automatically set to 3600000.
	KeepAlive	Interval for link check.[unit:ms] The valid value ranges from 0 to 3600000. When it is 0, KeepAlive function is disabled; When it is 0~10,000, it will be set to 10,000 automatically; When it is 10,000 ~360,000, the corresponding setting value will be used.
	ReserParam	Reserved parameter, used for extension.
Return	PPP_LOGINING	Module is logging in.
	PPP_LOGOUTING	Module is logging out.
	RET_OK	The link is established successfully.
	ERR_DEV_NOT_EXIST	Module does not exist.
	ERR_DEV_NOT_OPEN	Fail to perform OsWILOCK().
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_WL_POWER_ONING	In the process of powering on the module.
	ERR_WL_POWER_OFF	Module is powered off.
	ERR_WL_NOT_INIT	Fail to initialize Module.
	ERR_NET_PASSWD	Wrong password.
Instruction	ERR_NET_SERVER_BUSY	Server is busy, communication fails.
	ERR_NET_AUTH	Cannot connect to RADIUS server.
	1.	OsWILOCK() must be called successfully before calling this function.
	2.	OsWIInit() must be called successfully before calling this function.
	3.	When TimeOutMs=0, the function will return immediately.
	4.	The login duration depends on wireless module and network environment. In general, if the login duration is over 60 seconds, it means login failure or module exception.
	5.	It is recommended to restart the terminal after three consecutive failure of login.
	6.	After successfully establishing the link, the link will be set as the default route and IP network communication can be created.

7. When the return value is PPP_LOGOUTING, OsWICheck() can be called to check the logout status. The system has logged out successfully when OsWICheck() returns ERR_NET_LOGOUT.
8. When *KeepAlive* function is disabled, according to base station configuration, if no data communication has been detected for a long time, the base station may disconnect the link. When *KeepAlive* function is enabled, if the link receives no message within the *KeepAlive* time interval, then system will send ICMP message to the 8.8.8.8 automatically every time interval. .

21.2.10 OsWILoginEx

Prototype	<code>int OsWILoginEx(const char *DialNum, const char *APN, const char *Name, const char *Password, long Auth, int TimeOutMs, int KeepAlive, int ReserParam);</code>	
Function	Login wireless network and set up wireless link (dialing command can be modified).	
Parameters	DialNum[Input]	PPP dialing command. When DialNum is NULL, the default dialing command will be used. The valid string length is not more than 50 bytes.
	APN[Input]	For GPRS: APN is the access point name; For CDMA: APN is the dialing number. The valid length is not more than 50 characters. It can't be NULL.
	Name[Input]	User name. The valid length is no more than 50 characters. It can't be NULL. If there is no user name, a null string "" can be used instead.
	Password[Input]	Password. The valid length is not more than 50 characters. It can't be NULL. If there is no password, a null string "" can be used instead.

Auth	<p>Authentication algorithm. The system supports the following authentication algorithms:</p> <table border="1" data-bbox="652 339 1271 1237"> <tbody> <tr> <td>PPP_ALG_PA P</td><td>0x000000 01</td><td>PAP authentication algorithm</td></tr> <tr> <td>PPP_ALG_CHAP</td><td>0x000000 2</td><td>CHAP authentication algorithm</td></tr> <tr> <td>PPP_ALG_MS CHAPV1</td><td>0x000000 04</td><td>MSCHAPV1 authentication algorithm</td></tr> <tr> <td>PPP_ALG_MS CHAPV2</td><td>0x000000 08</td><td>MSCHAPV2 authentication algorithm</td></tr> <tr> <td>PPP_ALG_ALL</td><td>0xff</td><td>All algorithms are supported</td></tr> </tbody> </table> <p>At least one type of authentication or several authentications with (+) or () should be used, for example, PPP_ALG_PAP PPP_ALG_CHAP. If the algorithm is unknown, fill in parameter PPP_ALG_ALL.</p>	PPP_ALG_PA P	0x000000 01	PAP authentication algorithm	PPP_ALG_CHAP	0x000000 2	CHAP authentication algorithm	PPP_ALG_MS CHAPV1	0x000000 04	MSCHAPV1 authentication algorithm	PPP_ALG_MS CHAPV2	0x000000 08	MSCHAPV2 authentication algorithm	PPP_ALG_ALL	0xff	All algorithms are supported
PPP_ALG_PA P	0x000000 01	PAP authentication algorithm														
PPP_ALG_CHAP	0x000000 2	CHAP authentication algorithm														
PPP_ALG_MS CHAPV1	0x000000 04	MSCHAPV1 authentication algorithm														
PPP_ALG_MS CHAPV2	0x000000 08	MSCHAPV2 authentication algorithm														
PPP_ALG_ALL	0xff	All algorithms are supported														
TimeOutMs	<p>Timeout.[unit:ms] The valid timeout ranges from 0 to 3600000. If timeout <0, it will be automatically set to 0. If timeout >3600000, it will be automatically set to 3600000.</p>															
KeepAlive	<p>Time interval for link check.[unit:ms] The valid value ranges from 0 to 3600000. When it is 0, KeepAlive function is disabled; When it is 0~10,000, it will be set to 10,000 automatically; When it is 10,000 ~360,000, the corresponding setting value will be used.</p>															

	ReserParam	Reserved parameter, used for extension.
Return	PPP_LOGINING	Module is logging in.
	RET_OK	Link is set up successfully.
	ERR_DEV_NOT_EXIST	Wireless module does not exist.
	ERR_DEV_NOT_OPEN	Module/device is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_WL_POWER_ONIN G	In the process of powering on the module.
	ERR_WL_POWER_OFF	Module is powered off.
	ERR_WL_NOT_INIT	Fail to initialize module.
	ERR_NET_PASSWD	Wrong password.
	ERR_NET_SERVER_BUS Y	Server is busy, communication fails.
Instruction	<ol style="list-style-type: none"> 1. This function is similar to OsWILogin(). If <i>Dial/Num</i> is NULL, the two functions share totally the same functionality. 2. OsWILogout() must be called successfully before calling this function. 3. When TimeOutMs=0, the function will return immediately. 4. When the function returns PPP_LOGINING, it means module is logging in and the login status can be checked by OsWICheck(). 5. The login duration depends on wireless module and network environment. In general, if the login duration is over 60 seconds, it means login failure or module exception. 6. It is recommended to restart the terminal after three consecutive failure of login. 7. After the link is successfully established, it will be set as the default route and IP network communication can be created. 8. When KeepAlive function is disabled, according to base station configuration, if no data communication has been detected for a long time, the base station may disconnect the link. When KeepAlive function is enabled, if the link receives no message within the KeepAlive time interval, then system will send ICMP message to the 8.8.8.8 automatically every time interval. 	

21.2.11 OsWILogout

Prototype	int OsWILogout(void);
-----------	------------------------------

Function	Log out wireless network and disconnect wireless link.	
Parameters	None	
Return	PPP_LOGOUTING	Module is disconnecting link.
Instruction		ERR_DEV_NOT_OPEN Fails to call OsWILOCK().
		1. OsWILOCK() must be called successfully before calling this function. 2. The logout status can be checked by calling OsWICheck(). It has logged out successfully if OsWICheck() returns ERR_NET_LOGOUT.

21.3 Wireless Module Information Settings

21.3.1 OsWISelSim

Prototype	int OsWISelSim(int simno);	
Function	Select SIM card.	
Parameters	simno [Input]	<ul style="list-style-type: none"> · 0: Select the SIM card in slot 1. · 1: Select the SIM card in slot 2. · Others: Parameter error.
Return	RET_OK Succeeded. ERR_DEV_NOT_EXIST Module does not exist. ERR_DEV_NOT_OPEN Fail to call OsWILOCK(). ERR_WL_ERR_BUSY Module is busy. Other non-zero value Refer to Return Code List .	
Instruction	1. OsWILOCK() must be called successfully before calling this function. 2. OsWILinit() shall be called again to initialize module after successfully calling OsWISelSim(). In this process, the module will be powered off and then powered on, this function will block for about 15 seconds. 3. If the selected card slot has a bad card or has no card, the function will still return; when logging in, whether it is a bad card or no card can be detected.	

22WiFi

Prolin WiFi supports two modes: Station and IBSS work mode.

Station mode: the communication between the terminal and Access Point (AP), such as wireless router.

IBSS mode: not supported.

22.1 Return Code List

Table 22.1 Return code list

Macro	Value	Description
<i>ERR_MODE_NOT_SUPPORT</i>	-3350	<i>Mode setting error.</i>
<i>ERR_WIFI_POWER_OFF</i>	-3351	<i>Module is powered off.</i>
<i>ERR_NOT_FOUND_AP</i>	-3352	<i>AP is not found.</i>
<i>ERR_AUTH_SEC_MODE</i>	-3353	<i>Authentication mode or encryption mode error.</i>
<i>ERR_WIFI_BAD_SIGNAL</i>	-3354	<i>WiFi signal is weak.</i>
<i>RET_CONNECTING</i>	1	<i>Module is connecting.</i>
<i>ERR_EAP_ID</i>	-3359	<i>Certificate chain error or certificate verification failure</i>

22.2 Encryption Type List

Table 22.2 Encryption type list

Macro	Value	Description
PARE_CIPHERS_NONE	0x00000000	No encryption.
PARE_CIPHERS_WEP64	0x00000001	WEP 40-bit key.
PARE_CIPHERS_WEP128	0x00000002	WEP 104-bit key.
PARE_CIPHERS_WEPX	0x00000004	WEP encryption, unknown key bits.
PARE_CIPHERS_CCMP	0x00000010	AES encryption.
PARE_CIPHERS_TKIP	0x00000020	TKIP encryption.

22.3 Data Structure

Authentication Modes:

```
WIFI_AUTH_MODE

enum WIFI_AUTH_MODE{
    AUTH_NONE_OPEN=1,
    AUTH_NONE_WEP,
    AUTH_NONE_WEP_SHARED, /* The mode will be scanned as
                           AUTH_NONE_WEP */
    AUTH_IEEE8021X,
    AUTH_WPA_PSK,
    AUTH_WPA_EAP,
    AUTH_WPA_WPA2_PSK,
    AUTH_WPA_WPA2_EAP,
    AUTH_WPA2_PSK,
    AUTH_WPA2_EAP
};
```

Extension for WEP64 and WEP128:

WEP_SEC_KEY

```
typedef struct _WepSecKey{
    char Key[4][40]; /* WEP key data */
    int KeyLen;        /* Length of WEP key data */
    int Idx;          /* WEP key index [0,3] */
} WEP_SEC_KEY;
```

NOTE

WiFi connection doesn't support EAP encryption.

Extension for WPA/WPA2-PSK:**WPA_PSK_KEY**

```
typedef struct _WpaPskKey{
    char Key[64]; /* PSK-Key data */
    int KeyLen;   /* PSK-Key data length */
} WPA_PSK_KEY;
```

Extension for EAP:**WPA_EAP_KEY**

```
typedef struct _WpaEapKey{
    int EapType;      /* EAP type */
    char Pwd[132];   /* Password */
    char Id[132];    /* Identity */
    char CaCert[132]; /* Path and filename of CA certificate */
    char CliCert[132]; /* Path and filename of client certificate */
    char PriKey[132]; /* Private key file from file path to client */
    char PriKeyPwd[132]; /* Private key file of password */
} WPA_EAP_KEY;
```

Scan the AP information:

ST_WifiApInfo

```
typedef struct _WifiApInfo
{
    char Essid[33]; /* AP name */
    char Bssid[20]; /* MAC address */
    int Channel;      /* Information channel */
    int Mode;         /* Connection mode, 0:Station; 1:IBSS */
    int Rssi;         /* Signal value, the value range is [-99,0] */
    int AuthMode;     /* Authentication mode */
    int SecMode;      /* Encryption mode, NONE,WEP,TKIP,CCMP */
}ST_WifiApInfo;
```

Connect to AP settings:**ST_WifiApSet**

```
typedef struct _WifiApSet
{
    char Essid[33]; /* AP name, valid length is not more than 32 bytes,
                     ending with '\0'*/
    char Bssid[20]; /* MAC address, ending with '\0'; Bssid can be'\0' if there
                     is no AP with the same ESSID*/
    int Channel;     /* Information channel, which is valid only in IBSS
                     mode; 0: default setting */
    int Mode;         /* Connection mode, 0:Station; 1:IBSS */
    int AuthMode;     /* Authentication mode */
    int SecMode;      /*Encryption mode, NONE,WEP,TKIP,CCMP*/
    union KEY_UNION{ /* Key setting */
        WEP_SEC_KEY WepKey; /* WEP mode */
        WPA_PSK_KEY PskKey; /*wpa,wpa2-psk mode*/
        WPA_EAP_KEY EapKey; /* wpa,wpa2-eap mode*/
    } KeyUnion;
}ST_WifiApSet;
```

WPS mode enumeration:**WPS Mode**

```
enum WPS_MODE
{
    WPS_MODE_PBC = 1; /* Use keypad to connect, it is also called PBC
                       connection */
```

```

WPS_MODE_PIN_CLIENT; /*Use PIN code generated from terminal to
connect */
WPS_MODE_PIN_AP     /*Use PIN code from AP side to connect */
};

```

22.4 OsWifiOpen

Prototype	int OsWifiOpen(void);	
Function	Connect with WiFi server and obtain usage rights of WiFi module.	
Parameters	None	
Return	RET_OK ERR_DEV_NOT_EXIS T ERR_DEV_BUSY ERR_BATTERY_VOLT AGE _TOO_LOW ERR_BATTERY_ABSENT	Succeeded. Abnormal loading of module driver or module error. WiFi is busy. Battery voltage is too low. Battery does not exist.
Instruction	<ol style="list-style-type: none"> 1. D200 and S920 POS models must be powered by battery before accessing WiFi; otherwise, ERR_BATTERY_ABSENT will be returned. 2. When the battery voltage is 0, the module fails to work normally and returns ERR_BATTERY_VOLTAGE_TOO_LOW. 	

22.5 OsWifiClose

Prototype	voidOsWifiClose(void);	
Function	Release the usage right of WiFi module.	
Parameters	None	
Return	None	
Instruction	Call this function will not affect WiFi communication.	

22.6 OsWifiSwitchPower

Prototype	int OsWifiSwitchPower(int Type);	
Function	Power on/off the WiFi module.	
Parameters	Type	<ul style="list-style-type: none"> ▪ 0: power off module hardware. ▪ 1: power on module hardware.
Return	RET_OK	Succeeded.
	ERR_INVALID PARA M	Invalid parameter.
	ERR_DEV_NOT_EXI ST	Abnormal loading of module driver or module error.
	ERR_DEV_NOT_OP EN	Fail to access WiFi device.
Instruction	<ol style="list-style-type: none"> 1. WiFi module is automatically powered on with booting up of Prolin, thus this function doesn't need to be called. 2. It is a reserved interface and is not supported. 	

22.7 OsWifiScan

Prototype	int OsWifiScan(ST_WifiAplInfo **Aps);	
Function	Scan the existing network.	
Parameters	Aps[Output]	A pointer to ST_WifiAplInfo structure, storing the scanned network information.
Return	>=0	Number of AP that has been found.
	ERR_MEM_FAULT	Memory error
	ERR_INVALID PARA M	Invalid parameter
	ERR_WIFI_POWER_OFF	WiFi module is powered off.
	ERR_DEV_NOT_OP EN	Fail to access WiFi device.
Instruction	<pre style="font-family: monospace; padding: 0;"><i>/*For example:*/</i> <i>int i, num;</i> <i>ST_WifiAplInfo * Aps;</i> <i>num = OsWifiScan (&Aps);</i> <i>if(num <= 0)</i></pre>	

```

    return -1;
    for(i=0; i<num; i++)
printf("[%d] AP name: %s\n", i,Aps[i].Essid);

```

22.8 OsWifiConnect

Prototype	<code>int OsWifiConnect(const ST_WifiApSet *Ap, int TimeOutMs);</code>	
Function	Connect to a specified wireless network.	
Parameters	Ap [Input]	A pointer to ST_WifiApSet structure, storing the attributes of the specified wireless network.
	TimeOutMs[Input]	Timeout.[unit:ms] The valid timeout ranges from 0 to 3600000.
Return	RET_OK RET_CONNECTING ERR_NOT_CONNEC T ERR_WIFI_BAD_SIG NAL ERR_NOT_FOUND_ AP ERR_NET_PASSWD ERR_AUTH_SEC_M ODE ERR_WIFI_POWER_ OFF ERR_DEV_NOT_OP EN ERR_INVALID_PARA M	Succeeded. Module is connecting. Connection failed. WiFi signal is weak. AP can't be found. Wrong password. Authentication mode or encryption mode error WiFi module is powered off. Fail to access WiFi device. Invalid parameter.
Instruction	<ol style="list-style-type: none"> OsWifiCheck() can be used to check the connection status after RET_CONNECTING is returned. After a successful connection, OsNetStartDhcp() can be used to get the dynamic IP address; the OsNetSetConfig() can be used to set the static IP address. In IBSS mode, ERR_NOT_CONNECT will be returned 	

	<p>when the connection time exceeds 90 seconds; while in Station mode, a specific error code will be returned when the connection fails.</p> <ol style="list-style-type: none"> 4. In IBSS mode, if the connection fails, RET_CONNECTING will be returned and the terminal will create a new network according to the connection parameters. RET_OK will be returned if any terminal is connected to the network within 90 seconds; otherwise, ERR_NOT_CONNECT will be returned and the network will be terminated if no terminal is connected to the network. 5. Steps of checking whether the terminal has successfully connected to the specified network in IBSS mode: firstly ensure RET_OK is returned; then set IP; finally ping the opposite end IP. The connection is successful if ping succeeds. 6. User can set the member <i>Bssid</i> of ST_WifiApSet structure to connect to the specified AP, it is used particularly to distinguish the same ESSID network environments. If roaming is needed, <i>Bssid</i> must be set to "\0".
--	--

22.9 OsWifiDisconnect

Prototype	<code>int OsWifiDisconnect(void);</code>				
Function	Disconnect from the current network.				
Parameters	None.				
Return	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 45%;">RET_OK</td> <td>Succeeded.</td> </tr> <tr> <td>ERR_DEV_NOT_OP EN</td> <td>Fail to access WiFi device.</td> </tr> </table>	RET_OK	Succeeded.	ERR_DEV_NOT_OP EN	Fail to access WiFi device.
RET_OK	Succeeded.				
ERR_DEV_NOT_OP EN	Fail to access WiFi device.				
Instruction					

22.10 OsWifiCheck

Prototype	<code>int OsWifiCheck(char *Essid, char *Bssid, int *Rssi);</code>	
Function	Acquire the current network status.	
Parameters	Essid[Output]	The current connected ESSID The valid size is 33 bytes; it can't be NULL.

	Bssid[Output]	The current connected BSSID. The valid size is 20 bytes; it can be NULL.
	Rssi [Output]	Signal strength. The valid value ranges from -99 to 0. 0 is the strongest signal strength. It can't be NULL.
Return	RET_OK	Succeeded.
	RET_CONNECTING	Module is connecting.
	ERR_NOT_CONNE CT	Not connected to network.
	ERR_WIFI_BAD_SI GNAL	WiFi signal is weak.
	ERR_NOT_FOUND _AP	AP is not found.
	ERR_NET_PASSW D	Wrong password.
	ERR_AUTH_SEC_ MODE	Authentication mode or encryption mode error.
	ERR_INVALID_PAR AM	Invalid parameter.
Instruction	<ol style="list-style-type: none"> 1. The OsWifiCheck() can be implemented at any time, even before calling OsWifiOpen(). 2. ERR_NOT_CONNECT is returned in any of the following cases: <ol style="list-style-type: none"> a) OsWifiConnect() is not called by any of the application; b) OsWifiDisconnect() is called and RET_OK is returned; c) The connection time exceeds 90 seconds in IBSS mode. 3. In Station mode, if the connection fails, it will return error codes except ERR_NOT_CONNECT. 	

NOTE

The default status is “not connected” before a successful connection. That is, calling OsWIFICheck before OsWIFIConnect() will return ERR_NOT_CONNECT.

22.11 OsWifiCmd

Prototype	<code>int OsWifiCmd(const char *Argv[], int Argc, char *Result, int Len);</code>	
Function	Send directly commands to the WPA_Supplicant, and obtain the return results.	
Parameters	Argv [Input]	Command supported by WPA_Supplicant. It can't be NULL.
	Argc[Input]	The number of commands or parameters that are stored in <i>Argv</i> two-dimensional array.
	Result[Output]	Results returned from WPA_Supplicant. The valid length is more than 2048 bytes. It can't be NULL.
	Len [Input]	Length of <i>Result</i> array.
Return	<p>RET_OK Succeeded.</p> <p>ERR_INVALID_PARAMETER Invalid parameter.</p> <p>ERR_WIFI_POWER_OFF WiFi module is powered off.</p> <p>ERR_DEV_NOT_OPEN Fail to access WiFi device.</p>	
Instruction	<ol style="list-style-type: none"> 1. <i>Argv</i> can be all the commands and parameters supported by WPA_Supplicant, such as 'SCAN' command. 2. If there is only one command in <i>Argv</i>, set <i>Argc</i> to 1. 3. This function is called only when other WiFi APIs cannot meet requirements. 4. <i>Result</i> is the original return value of WPA_Supplicant. 	

22.12 OsWifiWpsConnect

Prototype	<code>int OsWifiWpsConnect(WPS_MODE Mode, char *WpsPin);</code>	
Function	Connect to AP through WPS mode.	
Parameters	Mode [Input]	WPS connection mode: WPS_MODE .

	WpsPin [Input/Output]	<ol style="list-style-type: none"> 1. If the connection mode is WPS_MODE_PBC, then <i>WpsPin</i> is NULL. 2. If the connection mode is WPS_MODE_PIN_CLIENT, then <i>WpsPin</i> will output the PIN code generated from terminal, the buffer that <i>WpsPin</i> points to is 8 bytes. 3. If the connection mode is WPS_MODE_PIN_AP, <i>WpsPin</i> is the PIN code of AP, and the buffer that <i>WpsPin</i> points to is 8 bytes.
Return	RET_CONNECTING ERR_DEV_NOT_OPEN ERR_INVALID_PARAM ERR_SYS_NOT_SUPPORT ERR_PIN_FAIL	In the process of connecting Open failed Invalid parameter WiFi module is not supported by system. PIN code error
Instruction	<ol style="list-style-type: none"> 1. WPS connection includes the following three different types of method: <ol style="list-style-type: none"> a) For PCB method (push the QSS(WPS) button of AP), <i>WpsPin</i> is NULL. b) The method of using PIN code generated from terminal to do the connection, <i>WpsPin</i> is an output parameter, and the value is a PIN code generated from terminal automatically. c) The method of using PIN code from AP to do the connection, <i>WpsPin</i> is an input parameter; the value is AP built-in PIN code. 2. When the return value is RET_CONNECTING, OsWifiCheck() can be called to check the connection state, the connection is normally finished within 2 minutes. 3. OsWifiDisconnect() can be called to disconnect the link during the connection process. 	

23GPS

23.1 Return Code List

Macro	Value	Description
GPS_NAVIGATING	1	<i>GPS positioning..</i>
ERR_GPS_POWER_ONING	-3901	<i>GPS module is being powered on.</i>

23.2 Data Definition

Structure of location information

GPS_LOCATION

```
typedef struct {
    double latitude;           /*Latitude, unit: degree.*/
    double longitude;          /*Longitude, unit: degree.*/
    double altitude;           /*Altitude, unit: meter.*/
} GPS_LOCATION;
```

23.3 OsGpsOpen

Prototype	<code>int OsGpsOpen(int Mode, const char *Attr);</code>	
Function	Open and initialize GPS module.	
Parameters	Mode	Positioning mode: GPS_MODE: normal GPS positioning mode. AGPS_MODE: AGPS positioning mode.
	Attr[Input]	Positioning parameter. When Mode is GPS_MODE, Attr will be ignored. When Mode is AGPS_MODE, it needs to pass in AGPS server address.
Return	RET_OK ERR_DEV_NOT_EXIST ERR_INVALID_PARAMETER ERR_SYS_NOT_SUPPORTED ERR_DEV_NOT_OPEN ERR_GPS_POWER_ONING	Succeeded. Device does not exist. Invalid parameter. System does not support. GPS module is not open. GPS module is being powered on.
Instruction	Call this function will initialize GPS module and start GPS background service. The first calling of this function may need up to 6s due to the handshake between backstage service and hardware.	

23.4 OsGpsRead

Prototype	<code>int OsGpsRead(GPS_LOCATION *Location);</code>	
Function	Read GPS location information.	
Parameters	Location	GPS location structure, GPS_LOCATION .
Return	GPS_NAVIGATING	GPS is navigating.

	RET_OK	Succeeded.
	ERR_INVALID_PAR AM	Invalid parameter.
	ERR_DEV_NOT_EXI ST	Device does not exist.
	ERR_DEV_NOT_OP EN	GPS module is not open.
Instruction	<ol style="list-style-type: none"> Only when RET_OK is returned, can GPS location information be read correctly. Influenced by GPS signal, the positioning may take several minutes, and the correct location information cannot be read until the positioning is completed. In an extreme case (places without GPS signal such as indoor, tunnel etc.), when GPS signal cannot be searched, GPS_NAVIGATING will be returned all the time until the positioning is completed. For the terminals with ro.fac.gps equal to 3, because their GPS modules are integrated into the wireless module, therefore, during the process of using GPS, if OsWISwitchPower() is called to power off the wireless module, ERR_DEV_NOT_OPEN will be returned for OsGpsRead(). At this point, the application needs to call OsGpsClose() to release the resources. After powering on the wireless module, call OsGpsOpen() again to open GPS device. 	

23.5 OsGpsClose

Prototype	void OsGpsClose(void);
Function	Close GPS module.
Parameters	None
Return	None
Instruction	<ol style="list-style-type: none"> GPS module can only be closed by the same application which was used to open it firstly. If GPS module is not open, this function does not do anything. As long as RET_OK is returned after calling OsGpsOpen(), OsGpsClose() must be called to close GPS module when program exits.

24 Base

24.1 Introduction

Newly added base APIs are used for operating base-set, including detecting whether the handset is on base and opening/detecting/closing the communication mode of handset-base.

When the communication mode is enabled on the handset-base, functions will be extended on base; (B920 extends the Ethernet function, and Px Dock extends printer and scanner functions). When the communication mode is closed on the handset-base, extended functions are invalid on base.

Mode instructions

- **Local mode:** The handset is used as an independent device. In this mode, applications can only access to the local physical device on the phone (such as printer, scanner and Ethernet).
- **Communication mode:** The handset is used as a handset with base cooperatively. And then the base works as a extention device of the handset, Prolin applications can access to local physical device on base by calling OSAL API, such as B920 Ethernet, Px Dock printer and scanner.

24.2 Macro Definition

Add two return values for handset-base model.

Macro	Value	Description
ERR_BASE_ABSENT	-1025	<i>Base is absent.</i>
ERR_BASE_COMM	-1026	<i>Communication failure</i>
ERR_LIB_NOT_FOUND	-1028	<i>Bluetooth library does not exist</i>
ERR_USB_MODE	-1029	<i>USB mode error</i>

Add virtual net channel for base.

Macro	Value	Description
NET_LINK_TAP	8	<i>tap vlan</i>

24.3 API

24.3.1 OsOnBase

Prototype	int OsOnBase(void);	
Function	Check whether the handset is connected to the base.	
Parameters	None	
Return	RET_OK ERR_BASE_ABSENT ERR_SYS_NOT_SUPPORT ERR_TIME_OUT ERR_SYS_BAD	Handset is on the base. Base is absent. System does not support this function. Timeout System error
Instruction	This function only applies to handsets connected to the base via USB or serial ports.	

24.3.2 OsBaseOpen

Prototype	int OsBaseOpen(void);
Function	Open the communication mode on the handset.

Parameters	None	
Return	RET_OK	Succeeded
	ERR_SYS_NOT_SUPP ORT	System does not support this function.
	ERR_SYS_BAD	System error.
	ERR_TIME_OUT	Timeout
	ERR_NOT_CONNECT	Bluetooth has not connected.
	ERR_DEV_BUSY	Device is busy.
	ERR_BASE_ABSENT	Base is absent.
	ERR_USB_MODE	USB mode error
	Others	Return values of the Bluetooth module
	<ol style="list-style-type: none"> After opening the handset mode, API of serial port, modem and Ethernet will be switched to the port which is corresponding to the operation base side; When the connection mode between the handset and the base is USB, please ensure the USB works in single channel mode, otherwise the interface will return ERR_USB_MODE. 	

24.3.3 OsBaseCheck

Prototype	int OsBaseCheck(void);	
Function	Check the base state.	
Parameters	None	
Return	BASE_STATE_CONNECTED	Bluetooth is connected but base is not open.
	BASE_STATE_CONNECTING	Connecting
	RET_OK	Base is open.
	ERR_DEV_NOT_OPEN	Base is not open.
	ERR_SYS_NOT_SUPP ORT	System does not support this function.
	ERR_SYS_BAD	System error.
	ERR_TIME_OUT	Timeout
	ERR_NOT_CONNECT	Bluetooth has not connected.
	Others	Return values of the Bluetooth

	module
Instruction	<p>Call this function in the Bluetooth base, the Bluetooth connecting state and base state will be returned.</p> <p>Enumeration as follow:</p> <pre>enum { BASE_STATE_OPENED = 0, /* Base is open */ BASE_STATE_CONNECTING, /* Connecting */ BASE_STATE_CONNECTED, /* Bluetooth is connected but base is not open. */ };</pre>

24.3.4 OsBaseClose

Prototype	int OsBaseClose(void);				
Function	Close the communication mode on the handset.				
Parameters	None				
Return	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 45%;">RET_OK</td> <td>Succeeded</td> </tr> <tr> <td>ERR_SYS_NOT_SUPPORT</td> <td>System does not support this function.</td> </tr> </table>	RET_OK	Succeeded	ERR_SYS_NOT_SUPPORT	System does not support this function.
RET_OK	Succeeded				
ERR_SYS_NOT_SUPPORT	System does not support this function.				
Instruction					

24.3.5 OsBaseScan

Prototype	int OsBaseScan(ST_HandsetLinkInfo *links, int n, int timeout);	
Function	Scan the Bluetooth base information.	
Parameters	links	Address of the incoming base information structure
	n	Number of the incoming structure arrays, the maximum value is 32.
	timeout	Timeout period, [unit: second]
Return	>=0	The number of the scanned base information. 0 indicates no base has been scanned.
	ERR_SYS_NOT_SUPPORT	System does not support this function.
	ERR_SYS_BAD	System error

	ERR_TIME_OUT	Timeout
	ERR_INVALID_PARAMETER	Invalid parameter.
	ERR_MEM_FAULT	Memory application failed.
	ERR_DEV_BUSY	Device is busy.
	ERR_LIB_NOT_FOUND	Bluetooth library cannot be found.
	Others	Return values of the Bluetooth module
Instruction		

Sample code

```
typedef struct _HandsetLinkInfo {
    int type;                                //Reserved for future use
    union {
        struct {
            char name[32];      //Bluetooth name
            char mac[32];      //Bluetooth MAC address
        } bt;               //Bluetooth base information
        unsigned char reserved[160];
    } u;
} ST_HandsetLinkInfo;
```

CAUTION

The timeout period of Bluetooth scanning is not accurate.

24.3.6 OsBaseConnect

Prototype	int OsBaseConnect(ST_HandsetLinkSet lk, int timeout);	
Function	Connect to the Bluetooth base.	
Parameters	lk	Base connection information.
	timeout	Timeout period
Return	RET_OK	Succeeded.
	ERR_SYS_NOT_SU	System does not support this function.

	PPORT	
	ERR_SYS_BAD	System error
	ERR_TIME_OUT	Timeout
	ERR_INVALID_PARAMETER	Invalid parameter.
	ERR_DEV_BUSY	Device is busy.
	ERR_LIB_NOT_FOUND	Bluetooth library cannot be found.
	Others	Return values of the Bluetooth module
Instruction	This function is non-blocking. When 0 is returned, it indicates the calling succeeded. Application can check the connecting state by calling OsBaseCheck(). When BASE_STATE_CONNECTING is returned, it indicates the Bluetooth base is in the connecting process.	

Sample code

```
typedef struct _HandsetLinkSet {
    int type;          //Reserved for future use
    union {
        struct {
            char mac[32];      //Bluetooth MAC address
        } bt;
        unsigned char reserved[1024];
    } u;
} ST_HandsetLinkSet;
```

NOTE



Bluetooth connection result is returned through the callback function, so the timeout parameter is temporarily ignored and reserved for future use.

24.3.7 OsBaseDisconnect

Prototype	int OsBaseDisconnect(void);	
Function	Disconnect the Bluetooth base connection.	
Parameters	None	
Return	RET_OK	Succeeded.

	ERR_SYS_NOT_SUPPORT	System does not support this function.
	ERR_SYS_BAD	System error
	ERR_TIME_OUT	Timeout
	ERR_DEV_BUSY	Device is busy.
Instruction	Only the process which has opened the base can close the base.	

24.3.8 OsCheckPortStatus

Prototype	int OsCheckPortStatus(int Channel);	
Function	Check if a physical connection exists at the specified communication port.	
Parameters	Channel[Input]	PORT_BASE_DEV: usbdev module of base PORT_BASE_ETH0: Ethernet module of base PORT_BASE_WLAN0: WIFI module of base
Return	RET_OK ERR_NOT_CONNECT ERR_INVALID_PARAM ERR_SYS_NOT_SUPPORT ERR_BASE_ABSENT ERR_SYS_BAD	Succeeded. It is not connected. Invalid parameter System does not support this function Base is absent. System error
Instruction	<ol style="list-style-type: none"> 1. Please ensure the handset is connected to the base before using it. 2. When <i>channel</i>= PORT_BASE_WLAN0, it is used to check whether the WIFI module of base exists, 0 is present, and -1012 is absent. 	

24.3.9 OsGetBaseInfo

Prototype	int OsGetBaseInfo(const char *Key, char *Value);
Function	Get the contents of the system configuration specified by the base.

Parameters	Key [Input]	System configuration name, ending with “\0”.
	Value [Output]	Parameter value. The valid string length must be more than 64 bytes.
Return	>=0	String length.
	ERR_INVALID_PARAM	Invalid Parameter
	ERR_NOT_CONNECT	Not connected
	ERR_BASE_ABSENT	Base is absent.
	ERR_TIME_OUT	Timeout
	ERR_SYS_NOT_SUPP ORT	System does not support
	ERR_SYS_BAD	System error
Instruction	<ol style="list-style-type: none"> 1. Please ensure that the handset is connected to the base before use. 2. For more values configured by system, refer to Appendix 3 Registry. 3. Additional system configurations supported by the base are as follows: 	
	System configuration name	Description
	ro.fac.digital_io	Whether there is a number IO(None means it does not exist by default.)
	ro.fac.com1	Whether com1 port that corresponds to PORT_BASE_A port exists, (None means it does not exist by default.)
	ro.fac.com2	Whether com2 port that corresponds to PORT_BASE_B port exists, (None means it does not exist by default.)

24.3.10 OsBaseCmd

Prototype	int OsBaseCmd(int cmd, void *inparm, void *outparm);
Function	Send commands to base.

Parameters	cmd[Input]	BASE_CMD_REBOOT: Restart base.
	inparm[Input]	Reserved for future use.
	outparm[Output]	Reserved for future use.
Return	RET_OK	Succeeded.
	ERR_INVALID_PARAM	Invalid Parameter
	ERR_NOT_CONNECT	Not connected
	ERR_BASE_ABSENT	Base is absent.
	ERR_TIME_OUT	Timeout
	ERR_SYS_NOT_SUPP ORT	System does not support ORT
	ERR_SYS_BAD	System error
Instruction	Please ensure that the handset is connected to the base before use.	

25File System

Prolin file system adopts standard ANSI.C file operation API.

26System Information

In Prolin, the system messages of hardware device are implemented by asynchronous notification. The system provides two kinds of hardware system messages:

- SIGMAG (magnetic card message)
- SIGICC (IC card message)

The SIGMAG is valid only when the magnetic stripe reader is open.

To register asynchronous notification function, the sample code is shown as follows:

Example

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <pthread.h>
#include <cosal.h>
#include <ccutils/log.h>
#define printf(...) LOGI(__VA_ARGS__)
static sigset_t mask;
void * handler_sigwait(void * arg)
{
```

```

int ret, signo;
while(1){
    ret = sigwait(&mask, &signo);
    if(ret != 0){
        printf("sigwait err, ret=%d\n", ret);
        break;
    }
    switch(signo){
    case SIGMAG:
        printf("Capture msr signal\n");
        break;
    case SIGICC:
        printf("Capture icc signal\n");
        break;
    default:
        printf("Capture other signal %d\n", signo);
        break;
    }
}
int main()
{
    int ret;
    sigset_t oldmask;
    pthread_t tid;
    ret = OsMsrOpen();
    if (ret < 0)
        exit(-1);
    ret = OsIccOpen(ICC_USER_SLOT);
    if (ret < 0){
        OsMsrClose();
        return -1;
    }
    sigemptyset(&mask);
    sigaddset(&mask, SIGMAG);
    sigaddset(&mask, SIGICC);
    ret = pthread_sigmask(SIG_SETMASK, &mask, &oldmask);
    if(ret != 0){
        printf("pthread_sigmask error, ret=%d\n", ret);
        exit(-1);
    }
}

```

```
}

ret = pthread_create(&tid, NULL, handler_sigwait, 0);
if(ret != 0){
    printf("pthread_create error, ret=%d\n", ret);
    exit(-1);
}
pthread_join(tid, NULL);

OsMsrClose();
OslccClose(ICC_USER_SLOT);

}
```

27 Audio

Prolin audio device is a speaker. In general, the volume settings are unified and can be set in TM interface.

27.1 Return Code List

Table 27.1 Return code list

Macro	Value	Description
<i>RET_RECORDING</i>	1	<i>In the recording.</i>
<i>ERR_NOT_RECORD</i>	-1027	<i>No recording</i>

27.2 OsRecordOpen

Prototype	int OsRecordOpen(void);	
Function	Establish a connection with Prolin recording service to obtain the permission of the recording device.	
Parameters	None.	
Return	0	Succeeded. ERR_DEV_NOT_EXI Device does not exist.

	ST	
	ERR_DEV_BUSY	Device is occupied by other application programs.
Instruction		<ol style="list-style-type: none"> 1. OsRecordOpen() should be called before calling OsRecordStart() or OsRecordStop(). 2. After all the operations have been completed, OsRecordClose() should be called to release the voice recording device.

27.3 OsRecordStart

Prototype	<pre><code>int OsRecordStart(const char *FileName, const char *Type, unsigned int Channel, const char *Format, unsigned int Rate, unsigned int Duration);</code></pre>	
Function	Start recording and save it as a specified audio file after the recording.	
Parameters	FileName[Input]	Path name to the recording file, including path and file name. It cannot be NULL. It is recommended to be stored in the local directory of the application.
	Type[Input]	Audio format, “wav” is supported. The format is “wav” when it is NULL.
	Channel[Input]	The number of channels, “1” indicates single track, “2” indicates dual track. Currently, Q90 only supports single track.
	Format[Input]	Sample format, it supports “U8” or “S16_LE”. The sample format is “S16_LE” when it is NULL. “S” indicates signed, “U” indicates unsigned; “LE” indicates little endian; “8/16” indicates sample length.
	Rate[Input]	Sample frequency, [Unit: Hz], the valid value can be 8000, 11025, 16000, 22050, 24000, 32000, 44100 or 48000.
	Duration[Input]	Recording time, [Unit: s], the valid value ranges from 1 to 600.
Return	RET_OK	Successed.
ERR_ACCESS_DENY	Access denied.	

	ERR_DEV_NOT_OPE N	Recording not available.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_NO_SPACE	Insufficient space available in system.
Instruction	<ol style="list-style-type: none"> 1. The audio file occupies a large space. For example, sample in the format of S16_LE, 44100Hz, single track, the recording time is 600s, and it is saved as "wav" format, then the audio file occupies about 50MB space. It is not recommended to record for too long time. And it is recommended to clear the unnecessary audio files in time to release the storage space. The recording will be terminated when the remaining space is less than 10M. 2. In the recording, OsRecordStop() can be called to stop recording, otherwise, the recording will keep up until the specified recording time (<i>Duration</i>) or the storage space is insufficient. 	

Audio sample code:

Example

```
if (OsRecordOpen())
    return -1;
OsRecordStart("/data/app/MAINAPP/test.wav", NULL, 1, NULL, 44100, 20);
while (1)
{
    if (!XuiHasKey())
        continue;
    key = XuiGetKey();
    if (key == XUI_KEY1)
        OsRecordStop();
    else if (key == XUI_KEY2)
        ret = OsRecordCheck();
    else if (key == XUI_KEYCANCEL)
        break;
}
OsRecordClose();
```

27.4 OsRecordStop

Prototype	int OsRecordStop(unsigned int *Time);	
Function	Stop recording.	
Parameters	Time[Input]	Duration of audio file.
Return	RET_OK	Success.
	ERR_SYS_BAD	System error.
	ERR_DEV_NOT_OP EN	Recording not available.
	ERR_NO_SPACE	Insufficient space available in system.
Instruction	<ol style="list-style-type: none"> 1. In the recording, OsRecordStop() can be called to stop the recording, and the duration of the audio file will be returned. 2. When calling OsRecordStop(): <ol style="list-style-type: none"> 1) if the recording has been stopped due to the arrival of the specified recording time (Duration), RET_OK will be returned. 2) if the recording has been stopped due to the insufficient storage space, ERR_NO_SPACE will be returned. 	

27.5 OsRecordCheck

Prototype	int OsRecordCheck(void);	
Function	Get the current recording status.	
Parameters	None	
Return	RET_RECORDING	In the recording.
	RET_OK	Finish recording.
	ERR_NOT_RECORD	No recording.
	ERR_SYS_BAD	System error.
	ERR_NO_SPACE	Insufficient space available in system.
Instruction	<ol style="list-style-type: none"> 1. If no application calls OsRecordStart(), RET_NOT_RECORD will be returned. 2. If the recording has been stopped due to the insufficient storage space, ERR_NO_SPACE will be returned. 	

27.6 OsRecordClose

Prototype	void OsRecordClose(void);	
Function	Release the access to the recording device and disconnect the device from the Prolin recording service.	
Parameters	None	
Return	None	
Instruction		

27.7 OsPlayWave

Prototype	int OsPlayWave(const char *Buf, int Len, int Volume, int DurationMs);	
Function	Play a specified wav audio file.	
Parameters	Buf [Input]	Audio data buffer.
	Len[Input]	Length of the data buffer.
	Volume[Input]	Volume. The valid volume ranges from 0 to 4; 0 means mute.
	DurationMs[Input]	Play duration.[Unit:ms].
Return	RET_OK Success. ERR_FILE_FORMAT File format error. ERR_ACCESS_DENY Access denied. ERR_INVALID_PARAM Invalid parameter. ERR_USER_CANCEL The user stop operating.	
Instruction	1. The value range of Volume is from 0 to 4. When Volume<0, it will be set to the value of <i>persist.sys.sound.volume</i> , which is the system volume and can be set in TM; When Volume>4, it will be set to 4 automatically. 2. The play duration has three cases: a) If DurationMs>Len, loop playback; b) If DurationMs<Len, DurationMs will be used; c) If DurationMs=0, the actual length of the audio data will	

- be used.
3. The system supports single track and double track WAVE audio file.
 4. The system supports 8-bit sampling and 16-bit sampling WAVE audio file. The supported sample frequencies are 8000 Hz, 11025 Hz, 16000 Hz, 22050 Hz, 24000 Hz, 32000 Hz, 44100 Hz and 48000 Hz.

Audio sample code (FILENAME is the name of an audio file):

Example

```

int fd, ret = 0;
char *buff;
int len;
struct stat state;
stat(FILENAME, &state);
len = state.st_size;
buff = (char *) malloc(len * sizeof(char));
fd = open(FILENAME, O_RDONLY);
if(fd<0)
printf("Open File Fail\n");
ret = read(fd, buff, len);
ret = OsPlayWave(buff, len, 3, 0);
if(ret != RET_OK)
printf("PlayWave Fail\n");
close(fd);
free(buff);

```

27.8 OsStopPlayWave

Prototype	int OsStopPlayWave(void);	
Function	Stop playing audio.	
Parameters	None	
Return	RET_OK	Succeeded.
	ERR_SYS_BAD	System error.
	ERR_DEV_NOT_OP EN	Audio device is not open.
	ERR_SYS_NOT_SU	System does not support this feature.

	PPORT
Instruction	<ol style="list-style-type: none"> When a thread calls OsPlayWave() to play an audio, OsStopPlayWave() can be called by another thread to stop playing the audio. If no audio is playing, ERR_DEV_NOT_OPEN will be returned.

27.9 OsPlayAudio

Prototype	int OsPlayAudio(const char *Buf, int Len, int Volume);	
Function	Play the specified MP3 audio.	
Parameters	Buf [Input]	The buffer area for audio data.
	Len[Input]	The length of data buffer
	Volume[Input]	Volume value, and the value range is [0,4], 0 represents mute.
Return	RET_OK	Succeeded.
	ERR_FILE_FORMAT	File format error.
	ERR_ACCESS_DEN Y	Failed to access device.
	ERR_INVALID PARA M	Invaild parameter.
	ERR_USER_CANCE L	User canceled the operation.
Instruction	The value range of <i>Volume</i> is [0,4], if the value is less than 0, the system volume (persist.sys.sound.volume) will be taken, and it can be set in TM; if the value is larger than 4, it will be set to 4 automatically.	

The sample code of playing MP3 audio is shown as below, the *FILENAME* is the audio file name.

Sample code

```
int fd = 0, ret = 0;
char *buf;
struct stat state;
fd = open(FILENAME, O_RDONLY);
```

```

if (fstat(fd, &stat) == -1 || stat.st_size == 0) {
    printf("audio file error\n");
    return -1;
}
buf = mmap(0, stat.st_size, PROT_READ, MAP_SHARED, fd, 0);
if (buf == MAP_FAILED)
    return -1;
ret = OsPlayAudio(buf, stat.st_size, 1);
printf("PlayAudio result = %d\n", ret);
munmap(buf, stat.st_size);
close(fd);

```

27.10 OsStopPlayAudio

Prototype	int OsStopPlayAudio(void);	
Function	Stop playing MP3 audio.	
Parameters	None	
Return	RET_OK	Succeeded. ERR_DEV_NOT_OP EN Audio device is not open.
Instruction	<ol style="list-style-type: none"> When a thread calls OsPlayWave() to play an audio, OsStopPlayWave() can be called by another thread to stop playing the audio. If no audio is playing, ERR_DEV_NOT_OPEN will be returned. 	

28Barcode and Camera

28.1 General Definition

Prolin supports one-dimensional, two-dimensional barcode, scanning barcode and taking photos through camera.

One-dimensional code is composed of vertical black and white bars of different widths, and generally there are letters or digits at the bottom of these bars. The code is commonly used to identify the basic information of products, such as name, price, etc.

Two-dimensional code is in a dot matrix form usually with square structure. Chequered with black and white bars of different widths, the code is dotted with polygonous images within its code area. In addition to the identification function, two-dimensional code can contain much more detailed contents.

Camera's scanning barcode function is implemented by acquiring image information through camera module, and it supports operations of supplementary light and positioning light.

Photography is implemented by calling API to operate camera module to open and capture the image content of current shooting, and return the content with specified image data format.

28.2 Data Definition

28.2.1 Macro Definition of Camera

Table 28.1 Camera definition list

Macro	Value	Description
USE_BACK_CAMERA	1	<i>Use the back camera.</i>
USE_FRONT_CAMERA	2	<i>Use the front camera.</i>
OPEN_BACK_ADDBRILAMP	10	<i>Open the back supplementary light.</i>
CLOSE_BACK_ADDBRILA MP	11	<i>Close the back supplementary light.</i>
OPEN_BACK_POSITIONL AMP	12	<i>Open the back positioning light.</i>
CLOSE_BACK_POSITIONL AMP	13	<i>Close the back positioning light.</i>
OPEN_FRONT_ADDBRILA MP	20	<i>Open the front supplementary light.</i>
CLOSE_FRONT_ADDBRILA MP	21	<i>Close the front supplementary light.</i>
OPEN_FRONT_POSITIONL AMP	22	<i>Open the front positioning light.</i>
CLOSE_FRONT_POSITION LAMP	23	<i>Close the front positioning light.</i>
OPEN_SCAN_LEDLIGHT	24	<i>Open the scanning LED</i>
CLOSE_SCAN_LEDLIGHT	25	<i>Close the scanning LED</i>

28.2.2 Image Resolution Macro Definition of Photography

Table 28.2 Image resolution definition list

Macro	Value	Description
RESOLUTION_WIDTH_640_HEIGHT_480	$1\ll 0$	<i>Image resolution is 640*480, and it is the only supported resolution.</i>
RESOLUTION_WIDTH_1024_HEIGHT_480	$1\ll 1$	<i>Image resolution is 1024*480, and it is the only supported resolution.</i>

28.2.3 Image Data Format Macro Definition of Photography

Table 28.3 Image data format definition list

Macro	Value	Description
CAMERA_PIXEL_FORMAT_YUYV	1<<0	The captured image data format is yuyv.
CAMERA_PIXEL_FORMAT_RGB565	1<<1	The captured image data format is rgb565.
CAMERA_PIXEL_FORMAT_SBGGR8	1<<2	The captured image data format is sbggr8.

28.2.4 The Attribute Structure of the Photography Parameter

CameraAttribute

```
typedef struct SCameraAttribute
{
    int SupportResolution; /*Read only, describes all the resolutions which the device supports in bits.*/
    int CurrentResolution; /*Read and write, it is the current resolution.*/
    int SupportPixelFormat; /*Read only, it is the format of output pixel data, and it only supports yuyv and rgb565 formats.*/
    int CurrentPixelFormat; /* Read and write, it is the format of current output data, and it only support yuyv format*/
    char Reserved[128]; /*Reserved for future use.*/
}CameraAttribute;
```

28.2.5 Return Code List

Table 28.4 Return code list

Macro	Value	Description
ERR_BAR_CAMER_INIT_FAILED	-3601	Failed to initialize camera.
ERR_BAR_ZSDECODE_INIT_FAILED	-3602	Failed to initialize zsdcode
ERR_BAR_SRECODE_INIT_FAILED	-3603	Failed to initialize srcode.
ERR_BAR_NOT_ENOUGH_BUFFER	-3604	Output buffer is not enough.
ERR_BAR_READ_CAMERA_BLOCK	-3605	Failed to read the data in

UFFE_FAILED	<i>camera buffer.</i>	
ERR_BAR_DECODE_FAILED	-3606	<i>Failed to decode.</i>

28.3 Basic Interface

28.3.1 OsScanSetParam

Prototype	int OsScanSetParam(unsigned int Param);	
Function	Open/Close the supplementary light, positioning light or scanning LED of camera.	
Parameters	Param [Input]	<ul style="list-style-type: none"> ● OPEN_BACK_ADDBRILAMP: Open the back supplementary light. ● CLOSE_BACK_ADDBRILAMP: Close the back supplementary light. ● OPEN_BACK_POSITIONLAMP: Open the back positioning light. ● CLOSE_BACK_POSITIONLAMP: Close the back positioning light. ● OPEN_FRONT_ADDBRILAMP: Open the front supplementary light. ● CLOSE_FRONT_ADDBRILAMP: Close the front supplementary light. ● OPEN_FRONT_POSITIONLAMP: Open the front positioning light. ● CLOSE_FRONT_POSITIONLAMP: Close the front positioning light. ● OPEN_SCAN_LEDLIGHT: Open the scanning LED ● CLOSE_SCAN_LEDLIGHT: Close the scanning LED ● USE_BACK_CAMERA: Use the back camera. ● USE_FRONT_CAMERA: Use the front camera.
Return	=0 Succeeded. ERR_INVALID_PARAMETER Invalid parameter. ERR_DEV_NOT_OPEN Device is not open.	
Instruction	1. QR55 has front supplementary light; D190/D195/D220/Q92 has back supplementary light; IM10/IM20 has scanning LED. 2. This function should be called after OsScanOpen()	

- succeeds.
3. If device only has one camera, this function will return 0 when the front or back camera is set, and scanning code function will be normal.

28.3.2 OsScanOpen

Prototype	<code>int OsScanOpen(void);</code>	
Function	Open the barcode scanning module.	
Parameters	None.	
Return	RET_OK ERR_DEV_BUSY ERR_DEV_NOT_OP EN ERR_DEV_NOT_EXI ST	Succeeded. Device is busy. Device is not open. Device does not exist.
Instruction	Because taking photo and scanning barcode are using the same camera device, if this function is called after OsCameraOpen(), <i>ERR_DEV_BUSY</i> will be returned, and vice versa.	

28.3.3 OsScanRead

Prototype	<code>int OsScanRead(char *Buf, int Len, int TimeoutMs);</code>	
Function	Scan and read the barcode in specified time.	
Parameters	Buf [Output] Len[Input]	Buffer of barcode data. For one-dimensional barcode, the buffer size is recommended to be more than 512 bytes; For two-dimensional code, the buffer size is recommended to be more than 3072 bytes. Buffer length.
	TimeoutMs[Input]	Timeout of barcode reading.[unit:ms] The valid timeout value ranges from 1500ms to 36000ms. Any timeout value that is less than 1500ms will be set to 1500ms; while more than 36000ms, it will be set to 36000ms. There may be an error less than 1 second between the actual timeout and the set value.

		The timeout is recommended to be 3000ms.
Return	>=0 ERR_DEV_NOT_OPE N ERR_TIME_OUT ERR_INVALID_PARA M	The actual length of barcode data. Device is not open. Timeout. Invalid parameter
Instruction		

28.3.4 OsScanClose

Prototype	void OsScanClose(void);	
Function	Close scanning device.	
Parameters	None.	
Return	None.	
Instruction		

28.3.5 OsCameraOpen

Prototype	int OsCameraOpen(int Index);	
Function	Open camera device.	
Parameters	Index[Input]	Camera index, 0 represents rear camera; 1 represents front camera.
Return	RET_OK ERR_INVALID_PARA M ERR_DEV_NOT_EX IST ERR_DEV_BUSY	Succeeded. Invalid parameter. Device does not exist. Camera device is busy.
Instruction	Because taking photo and scanning barcode are using the same camera device, if this function is called after OsScanOpen(), <i>ERR_DEV_BUSY</i> will be returned, and vice versa.	

28.3.6 OsCameraClose

Prototype	void OsCameraClose(void);
------------------	----------------------------------

Function	Close camera device.	
Parameters	None.	
Return	None.	
Instruction		

28.3.7 OsCameraGetParam

Prototype	<code>int OsCameraGetParam(CameraAttribute *CameraParam);</code>	
Function	Obtain camera parameters.	
Parameters	CameraParam[Output]	Camera parameters, including the attribute settings of camera.
Return	RET_OK ERR_DEV_NOT_OPEN ERR_INVALID_PARAM	Succeeded. Camera device is not open. Invalid parameter.
Instruction		

28.3.8 OsCameraSetParam

Prototype	<code>int OsCameraSetParam(CameraAttribute *CameraParam);</code>	
Function	Set camera parameters.	
Parameters	CameraParam[Input]	Camera parameters, including the attribute settings of camera.
Return	RET_OK ERR_DEV_NOT_OPEN ERR_INVALID_PARAM ERR_SYS_NOT_SUPPORTED	Succeeded. Camera device is not open. Invalid parameter. System does not support.
Instruction	This function can only set the read-write member in <i>CameraAttribute</i> structure. For example, the value of <i>CurrentResolution</i> must be supported by camera. Before setting camera parameter, it is recommended to call <i>OsCameraGetParam()</i> to obtain all the resolution types supported by system.	

28.3.9 OsCameraCapture

Prototype	int OsCameraCapture(char* Buf, int Size);	
Function	Take photo through camera.	
Parameters	Buf[Output]	The data buffer of photography.
	Size[Input]	The memory size of <i>Buf</i> .
Return	<p>>0 Actual collected data length.</p> <p>ERR_DEV_NOT_OPEN Camera device is not open.</p> <p>ERR_INVALID_PARAM Invalid parameter.</p> <p>ERR_DEV_BUSY Device is busy.</p> <p>ERR_BAR_CAMER_INIT_FAILED Camera initialization failed.</p> <p>ERR_BAR_NOT_ENOUGH_BUFFER Insufficient buffer.</p> <p>ERR_BAR_READ_CAMERA_BUFFER_FAILED Obtain photography data failed.</p>	
Instruction	<ol style="list-style-type: none"> 1. <i>Buf</i> is used to buffer the photography data, the calculation formula of the data size is [height]* [width]* [the number of bytes occupied by each pixel]. For example, the <i>Size</i> of an image with 480*640 resolution in yuyv/ rgb565 format (each pixel occupies 2byte) should be $480*640*2 = 614400$; while the <i>Size</i> of an image with 1024*480 resolution in sbggr8 format (each pixel occupies 1byte) should be $1024*480*1 = 491520$; the <i>Size</i> of an image with 640*480 resolution in NV21 format (each pixel occupies 3byte) should be $640*480*3/2 = 460800$. 2. For image data in rgb565, the rgb information is stored by the byte stream. For example, the first pixel point is stored by <i>Buf[0]</i> and <i>Buf[1]</i>, and the rgb information is shown as below: $r = \text{Buf}[0] \& 0xF8;$ $g = ((\text{Buf}[0] << 5) \& 0xE0) ((\text{Buf}[1] >> 3) \& 0x1C);$ $b = \text{Buf}[1] << 3;$ or <i>Buf[0]</i> and <i>Buf[1]</i> can be combined into a temporary variable in short data type: $\text{unsigned short tmp_pixel} = ((\text{unsigned short})\text{Buf}[0] << 8) \text{Buf}[1];$ $r = (\text{tmp_pixel} >> 8) \& 0xF8;$ $g = (\text{tmp_pixel} >> 3) \& 0xFC;$ 	

	<pre>b = tmp_pixel << 3;</pre> <p>3. For image data in yuyv, all Y values will be assigned to r.g.b, and U and V can be ignored, each 4-byte yuyv data corresponds to two pixels:</p> <pre>int *yuyv = (int*)Buf; int yy = *yuyv; //The first pixel char y1 = (char)yy; r = y1; g = y1; b = y1; // The second pixel char y2 = (char)(yy >> 16); r = y2; g = y2; b = y2; //Keep traversing the yuyv data yuyv++;</pre> <p>4. For image data in sbgr8, each byte is gray value, that is, the value is assigned to r.g.b byte by byte:</p> <pre>char* ptr = Buf; r = *ptr; g = *ptr; b = *ptr; // Keep traversing the data ptr++;</pre> <p>5. For image data in NV21, the first 640*480 bytes of the <i>Buf</i> are the Y values of each pixel, that is, the value is assigned to r.g.b byte by byte:</p> <pre>char* y = (char*)Buf; r = y; g = y; b = y; // Keep traversing the data y++;</pre>
--	---

28.3.10 OsScanDecodeBuf

Prototype	<pre>int OsScanDecodeBuf(const unsigned char *DataIn, int DataInLen, unsigned char *DataOut, int MaxLen, int Width, int Height, int Format);</pre>
Function	Decode image data which containing the barcode data.
Parameters	DataIn[Input] Image data.
	DataInLen[Input] Length of the image data.
	DataOut[Output] The buffer used to store barcode data. For one-dimensional code, the

		buffer size is recommended to be more than 512 bytes; for two-dimensional code, the buffer size is recommended to be more than 3072 bytes.
	MaxLen[Input]	The size of the <i>DataOut</i> .
	Width[Input]	Image width. The value range is from 320 to 1280.
	Height[Input]	Image height. The value range is from 320 to 1280.
	Format[Input]	The format of image data: CAMERA_PIXEL_FORMAT_YUYV.
Return	>=0	Read the byte length of the barcode data from the image data.
	ERR_SYS_NOT_SUPP ORT	System does not support.
	ERR_BAR_DECODE_F AILED	Decoding failed.
	ERR_INVALID_PARAM	Invalid parameter.
Instruction	The parameters DataInLen, Width and Height conform to the formula “ DataInLen=Width * Height * 2 ”.	

28.3.11 OsCameraDetectMotion

Prototype	<code>int OsCameraDetectMotion(int threshold, int interval, int timeout);</code>	
Function	Motion detection, the camera captures images at specified intervals and checks for differences in two consecutive images.	
Parameters	threshold[Input]	The threshold of image difference ranges from 0 to 100. When the difference between two consecutive frames captured by the camera is not less than the value of this parameter, the function returns the actual difference.
	interval[Input]	The interval between images captured by the camera. [unit: milliseconds]
	timeout[Input]	Interface blocking timeout in seconds. The interface will return

		ERR_TIME_OUT if the interface blocking has reached timeout and the image difference has not been detected to reach the set threshold.
Return	>=0	Actual difference value of two consecutive frames of images captured by the camera (ranges from threshold to 100).
	ERR_DEV_NOT_OPEN	Camera is not open.
	ERR_INVALID_PARAM	Invalid parameter.
	ERR_TIME_OUT	Timeout.
Instruction		<ol style="list-style-type: none"> 1. Call OsCameraOpen() to open the camera before calling this function, otherwise ERR_DEV_NOT_OPEN will be returned, and call OsCameraClose() to close the camera after the detection is finished. 2. The range of detected difference value is 0~100, so the parameter threshold must be within this range, otherwise ERR_INVALID_PARAM will be returned. In the actual application scenario, the recommended value range is 20~50. 3. The value of the parameter interval that captures the image determines the sensitivity of the detection. In the actual application scenario, the recommended value range is 500~2000 in milliseconds. 4. The unit of the parameter timeout is second. When timeout is set to 0, this function immediately returns the comparison difference value after capturing two frames of images. When timeout is greater than 0, interface blocking works. If the difference value of two consecutive frames of images is not less than the threshold parameter, the difference value will be returned; otherwise, ERR_TIME_OUT will be returned after blocking time.

29 Power Management

29.1 Return Code List

Macro	Value	Description
BATTERY_LEVEL_0	0	<i>Battery level is 0.</i>
BATTERY_LEVEL_1	1	<i>Battery level is 1.</i>
BATTERY_LEVEL_2	2	<i>Battery level is 2.</i>
BATTERY_LEVEL_3	3	<i>Battery level is 3.</i>
BATTERY_LEVEL_4	4	<i>Battery level is 4.</i>
BATTERY_LEVEL_CHARGE	5	<i>Battery is being charged.</i>
BATTERY_LEVEL_COMPLETE	6	<i>Battery is fully charged.</i>
BATTERY_LEVEL_ABSENT	7	<i>Battery is absent.</i>

29.2 Data Structure

PM_MSG_T: events broadcasted by pm.

PM_MSG_T:

```
typedef enum {
    PM_MSG_NO_EVENT,          /* No event.*/
    PM_MSG_ENTER_SLEEP,       /* Device enters sleep mode.*/
    PM_MSG_EXIT_SLEEP,        /* Device exits sleep mode.*/
    PM_MSG_ENTER_SCREENSAVER, /* Device enters screensaver
mode.*/
    PM_MSG_EXIT_SCREENSAVER,  /* Device exits screensaver
mode.*/
    PM_MSG_ENTER_POWEROFF,    /* Device starts to power off*/
    PM_MSG_POWER_ABNORMAL,   /* Device power is abnormal*/
    PM_MSG_BATTERY_DAMAGE,   /* Battery is out of service */
} PM_MSG_T;
```

PM_REQ_T: action requested by client.

PM_REQ_T:

```
typedef enum  {
    PM_FORBID_SLEEP,      /* Forbid device from sleeping. */
    PM_ALLOW_SLEEP,        /* Allow device to sleep.*/
    PM_FORBID_SCREENSAVER, /*Forbid device from entering
screensaver mode. */
    PM_ALLOW_SCREENSAVER,  /* Allow device to enter screensaver
mode.*/
    PM_FORBID_POWEROFF,    /*Forbid device from powering off. */
    PM_ALLOW_POWEROFF,     /*Allow device to power off.*/
} PM_REQ_T;
```

POWER_TYPE: the type of power supply.

POWER_TYPE:

```
typedef enum {
    POWER_ADAPTER = 1,    /*Supplied by adapter.*/
    POWER_USB,            /*Supplied by USB external device.*/
    POWER_BATTERY,        /*Supplied by battery.*/
    POWER_WPC             /*Supplied by wireless base.*/
} POWER_TYPE;
```

WAKEUP_SOURCE: the wakeup source

WAKEUP_SOURCE:

```
typedef enum {
    WAKEUP_SRC_NONE = 0, /* No wakeup has been done, it has no
                          wakeup source. */
    WAKEUP_SRC_KP, /* Key pressing wakeup */
    WAKEUP_SRC_RTC, /* Timer wakeup */
    WAKEUP_SRC_BT, /* Bluetooth wakeup */
    WAKEUP_SRC_CHC, /* Power wakeup */
    WAKEUP_SRC_WIFI, /* WIFI wakeup */
    WAKEUP_SRC_MSR, /* MSR wakeup */
    WAKEUP_SRC_SMARTCARD0 = 8, /* IC card wakeup */
    WAKEUP_SRC_UART = 12, /* serial port wakeup*/
    WAKEUP_SRC_ETHER, /* Ethernet wakeup */
    WAKEUP_SRC_GENERAL1 = 1000, /* general wakeup 1 */
    WAKEUP_SRC_GENERAL2, /* general wakeup 2 */
    WAKEUP_SRC_GENERAL3, /* general wakeup 3 */
    WAKEUP_SRC_GENERAL4, /* general wakeup 4 */
    WAKEUP_SRC_GENERAL5, /* general wakeup 5 */
} WAKEUP_SOURCE;
```

29.3 OsCheckBattery

Prototype	int OsCheckBattery(void);	
Function	Check the battery level.	
Parameters	None.	
Return	BATTERY_LEVEL_0 0~5%, low battery and need to be charged immediately. Transaction, wireless communications and printing are not recommended. When the battery is too low, the system will automatically power off. BATTERY_LEVEL_1 5%~15% BATTERY_LEVEL_2 15%~40% BATTERY_LEVEL_3 40%~70% BATTERY_LEVEL_4 70%~100%	

	BATTERY_LEVEL_CHARGE	Battery is being charged.
	BATTERY_LEVEL_COMPLETE	Battery is fully charged; external power supply.
	BATTERY_LEVEL_ABSENT	Battery does not exist; external power supply.
	ERR_SYS_NOT_SUPPORTED	System does not support checking battery level. Only terminal without battery can return this value.
Instruction	<ol style="list-style-type: none"> When the terminal is being charged with external power supply, it can detect whether the battery is fully charged or not but cannot obtain the battery level. The battery level can be detected only when the device is powered by built-in battery. It is not recommended to call this function during the process of searching RF card, connecting wireless network or printing, as the returned battery level might bounce. When BATTERY_LEVEL_0 is returned, wireless module, printer and RF module may fail to work, the battery should be charged immediately. 	

29.4 OsCheckPowerSupply

Prototype	int OsCheckPowerSupply(void);	
Function	Check the type of power supply.	
Parameters	None.	
Return	POWER_BATTERY	Powered by built-in battery.
	POWER_ADAPTER	Powered by power adapter.
	POWER_USB	Powered by USB device, such as PC.
	POWER_WPC	Powered by wireless base.
Instruction	Only D220 supports powering by wireless base.	

29.5 OsSysSleep

Prototype	int OsSysSleep(void);	
Function	Get the terminal to enter sleep mode.	
Parameters	None.	
Return	RET_OK ERR_SYS_NOT_SU PPORT	Succeeded. Device doesn't support. Device is busy.
Instruction	<p>1. The device will fail to enter sleep mode and return ERR_DEV_BUSY in the following situations:</p> <ol style="list-style-type: none"> 1) OsPmRequest() has been called to forbidd sleeping. 2) POS is being used as USB device <p>2. During the sleep, CPU will stop and the screen will be turned off. After waken up by key, the screen will display the same content as that before sleep and the system will continue to run from where it stopped.</p> <p>3. After Modem dialed successfully, calling OsSysSleep() or OsSysSleepEx(2) will return ERR_DEV_BUSY, and the system cannot sleep. After Modem hanged up, the system can sleep by calling sleep function.</p> <p>4. After the terminal enters sleep mode, Keypad and communication module will not be powered off.</p> <p>5. P^X5, P^X7, Q30, SP200, IM500, IM700 and IM310 don't support this function.</p>	

NOTE


1. It is not recommended to start sleep when using RF card. Otherwise, after the system wakes up, OsPiccClose() and OsPiccOpen() must be called before performing other card functions.
2. It is recommended to disconnect PPP connection by calling OsWILogout() before system enters sleep state, and to set up PPP connection by calling OsWILogin() after system wakes up.
3. After system wakes up, OsScanClose() and OsScanOpen() must be called in advance to process scanning operation.

29.6 OsSysSleepEx

Prototype	int OsSysSleepEx(int Level);	
Function	Set terminal power management mode.	
Parameters	Level[Input]	<p>Sleep level, value range is [0, 2].</p> <p>0: System runs normally;</p> <p>1: Screensaver mode. CPU works normally; LCD, key backlight, touch key and touch screen can be woken up by plastic button and application program.</p> <p>2: System sleeps. CPU is in standby mode, all modules are closed, parts of modules can be awakened by plastic button; and some of them can be awakened by WiFi or power button.</p>
Return	<p>RET_OK Succeeded.</p> <p>ERR_INVALID_PARAM Invalid parameter.</p> <p>ERR_SYS_NOT_SUPPO RT Device doesn't support.</p> <p>ERR_DEV_BUSY Device is busy.</p>	
Instruction	<ol style="list-style-type: none"> When Level=1, the system will enter screensaver mode. In one of the following scenarios, it will fail to enter the screensaver mode and return ERR_DEV_BUSY. <ol style="list-style-type: none"> OsPmRequest() has been called to forbid screensaver. The system is in PED mode. When Level=2, it is the same as OsSysSleep(). The opened handles will not be closed in any sleep level. Neither Level 0 nor Level 1 will change the current working state of cards and communication modules. While in Lever 2, all the modules will be closed except the plastic button, communication module and the module which has been set the wake-up source. Normally, the screensaver mode will be activated if there is no input within the specified interval whose default value is 60 seconds. The interval can be set by <i>persist.sys.backlighttime</i> (unit: sec). <i>persist.sys.backlighttime=0</i> means turning off the screensaver. P^X5, P^X7, Q30, SP200 , IM500, IM700 and IM310 don't 	

	support the Lever=2 sleep mode.
--	---------------------------------

29.7 OsSysSleepTime

Prototype	int OsSysSleepTime(int Time);	
Function	Enter sleep mode and wakeup automatically within specified Time.	
Parameter	Time[Input]	Sleep duration.[Unit: sec] The valid sleep duration ranges from 60 to 43200 (that is, 12 hours). For Prolin-2.4 and Prolin-phoenix-2.5: If the sleep duration <128, it will be set to 128 automatically. For Prolin-cygnus-2.6 and higher version:If the sleep duration <60, it will be set to 60 automatically; if the sleep duration>43200, it wil be set to 43200 automatically.
Return	RET_OK ERR_SYS_NOT_SUP PORT	Suceeeded. Not supported by device.
Instruction	<ol style="list-style-type: none"> The timing error of Prolin-2.4 and Prolin-phoenix-2.5 is $\pm 128s$; the timing error of Prolin-cygnus-2.6 and higher version is $\pm 1s$. ERR_SYS_NOT_SUPPORT will be returned, if this function is called by Px5, Px7, Q30, SP200, IM500, IM700 or IM310; If OsPmRequest() is called or terminal is connected as a USB device, calling this function will return ERR_DEV_BUSY and the terminal cannot enter sleep mode. 	

29.8 OsReboot

Prototype	int OsReboot(void);	
Function	Reboot the terminal.	
Parameters	None.	
Return	ERR_SYS_BAD	System error.
Instruction	This function works in blocking mode. If this function is	

executed successfully, the terminal will reboot directly without any return value.

29.9 OsPowerOff

Prototype	int OsPowerOff(void);	
Function	Power off the terminal.	
Parameters	None.	
Return	ERR_SYS_BAD	System error.
Instruction	This function works in blocking mode. If this function is executed successfully, the terminal will be powered off directly without any return value.	

29.10 OsPmGetEvent

Prototype	PM_MSG_T OsPmGetEvent(int TimeoutMs);	
Function	Get the events sent by PM module.	
Parameters	TimeoutMs [Input]	Timeout.[Unit: ms] The value range of TimeoutMs includes 0 and number ranging from 100 to 3600000. TimeoutMs=0 means to get the event in nonblocking mode.
Return	ERR_INVALID_PARAMETER >=0	Invalid parameter Please refer to PM_MSG_T
Instruction	Each time OsPmGetEvent is called, a power management event can be got. It is recommended that the <i>TimeoutMs</i> parameter should not be more than 1000ms.	

NOTE



1. There must be only one OsPmGetEvent() in a process.
2. Before an event occurs, call OsPmGetEvent(0) to establish a link between the application process and the daemon service, after establishment, the subsequent power

management events can be sent to the application process. And OsPmGetEvent() can be called to get event directly when the power management event occurs. If OsPmGetEvent() is not called for a long time, the events will be lost.

29.11 OsPmRequest

Prototype	int OsPmRequest(PM_REQ_T ReqType);	
Function	The client requests device to enter into a specified mode	
Parameters	ReqType[input]	The corresponding meanings of specified events please refer to PM_REQ_T structure. For example: PM_FORBID_SLEEP: Forbid the device from entering sleep mode. PM_ALLOW_SLEEP: Allow the device to enter sleep mode.
Return	ERR_INVALID_PARAM	Parameter error.
	RET_OK	Succeeded.
Instruction	<ol style="list-style-type: none"> 1. The “forbid” and “allow” action don’t have to be used in pairs. For example: RET_OK will always be returned no matter how many times PM_FORBID_SLEEP is called. 2. The device will enter sleep mode as long as PM_ALLOW_SLEEP is called. 	

29.12 OsWakeupSource

Prototype	int OsWakeupSource(void);	
Function	Get the wakeup source which makes the terminal wake up.	
Parameters	None	
Return	>=0	Enumeration values of the wakeup source. Please refer to the enumeration structure WAKEUP_SOURCE for the

	<p>corresponding wakeup source of each value.</p> <p>ERR_SYS_NOT_SU The system does not support. PPORT</p>
Instruction	Because different models have different hardware configurations, a model may support only one or several wakeup sources.

29.13 OsCheckPowerStatus

Prototype	int OsCheckPowerStatus(int *PowerStatus);					
Function	Check the power status.					
Parameters	PowerStatus[Output]	<p>Power status:</p> <p>Bit0: Battery voltage status, 0-normal; 1-abnormal;</p> <p>Bit1: Battery charging status, 0-normal; 1-abnormal;</p> <p>Bit2: Charging IC status, 0-normal; 1-abnormal;</p> <p>Bit3: Whether the battery is damaged, 0-no, 1-yes;</p> <p>Bit4: Whether the charging cycle exceeds the limitation, 0-no, 1-yes;</p> <p>Bit5: SOH, 0-health, 1-aging seriously; Bit6~Bit31: Reserved.</p>				
Return	RET_OK ERR_SYS_NOT_SU PPORT ERR_INVALID_PARA M	<p>Succeeded</p> <p>Device does not support.</p> <p>Invalid parameter.</p>				
Instruction	<ol style="list-style-type: none"> This function only supports terminals with battery, such as S920, D190, D195 and terminals with voltmeter (Q80, Q92, QR68, QR65); Due to the differences in model and hardware, the system may not be able to detect all the abnormal types in the above list. <table border="1"> <thead> <tr> <th>Model</th> <th>Exception type that support detection</th> </tr> </thead> <tbody> <tr> <td>S920</td> <td>S920 V20: Bit0-Bit5</td> </tr> </tbody> </table>		Model	Exception type that support detection	S920	S920 V20: Bit0-Bit5
Model	Exception type that support detection					
S920	S920 V20: Bit0-Bit5					

	Other S920 with voltmeter: Bit0-Bit2
D190, D195	D190 V06 and higher version: Bit0-Bit1 Other D190 version and D195: Bit0
Q80	Bit0, Bit3-Bit5
Q92	Bit0-Bit5
QR68, QR65	Bit0



CAUTION In case of abnormal battery damage of Bit3, the system will periodically broadcast the PM_MSG_BATTERY_DAMAGE event and stop charging, and the battery voltage is controlled within 3.6V. Please be sure to replace the battery. Other abnormal events are broadcast with PM_MSG_POWER_ABNORMAL.

29.14 OsCheckBMSMode

Prototype	<code>int OsCheckBMSMode(int*CurrentMode, int *Capacity, int *FullCharge, int *Recharge);</code>	
Function	Get information about battery management system.	
Parameters	CurrentMode[Output]	Current mode of the terminal.
	Capacity[Output]	Current battery capacity, it ranges from 0 to 100%.
	FullCharge[Output]	Full charge, it ranges from 0 to 100%.
	Recharge[Output]	Recharge, it ranges from 0 to 100%.
Return	RET_OK	Succeeded
	ERR_INVALID_PARA M	Invalid parameter.
	ERR_SYS_NOT_SU PPORT	System does not support.
Instruction		

29.15 Get the Power Management Information

```
PM_MSG_T msg;
while(1) {
    msg = OsPmGetEvent(100000);
    if (msg > 0) {
        switch (msg) {
            case PM_MSG_ENTER_SLEEP:
                break;
            case PM_MSG_EXIT_SLEEP:
                break;
            /*add other case*/
            default:
                break;
        }
    }
}
```

29.16 Client Sends Request

```
OsPmRequest(PM_FORBID_SLEEP);      /* Forbid system from sleeping.*/
OsPmRequest(PM_ALLOW_SLEEP);       /* Allow system to sleep.*/
OsPmRequest(PM_FORBID_SCREENSAVER); /* Forbid screensaver.*/
OsPmRequest(PM_ALLOW_SCREENSAVER);  /* Allow screensaver.*/
OsPmRequest(PM_FORBID_POWEROFF);   /*Forbid system from powering off.*/
OsPmRequest(PM_ALLOW_POWEROFF);    /*Allow system to power off.*/
```

Appendix 1 PIN Block Format

Format 0 PIN block

This PIN block is constructed by modulo-2 addition of two 64-bit fields: the plain text PIN field and the account number field. The formats of these fields are described in 1.1.1 and 1.1.2 respectively.

The format 0 PIN block shall be reversibly enciphered when transmitted.

Plain text PIN field

The plain text PIN field shall be formatted as follows.

Bit

1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 64



where

C = Control field: shall be binary 0000;

N = PIN length: 4-bit binary number with permissible values of 0100(4) to 1100(12);

P = Pin digit: 4-bit field with permissible values of 0000(zero) to 1001(9);

P/F = PIN/Fill digit: designation of these fields is determined by the PIN length field;

F = Fill digit: 4-bit field value 1111(15).

Account number field

The account number field shall be formatted as follows.

Bit

1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 64

0	0	0	0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
---	---	---	---	----	----	----	----	----	----	----	----	----	-----	-----	-----

Wherein,

0 = Pad digit: a 4-bit field with the only permissible value of 0000(zero);

A1...A12 = Account number: content is the 12 rightmost digits of the primary account number (PAN) excluding the check digit. A12 is the digit immediately preceding the PAN's check digit. If the PAN excluding the check digit is less than 12 digits, the digits are right justified and padded to the left with zeros. Permissible values are 0000 (zero) to 1001 (9).

Format 1 PIN block

This PIN block is constructed by concatenation of two fields: the plain text PIN field and the transaction field.

The format 1 PIN block should be used in situations where the PAN is not available.

The format 1 PIN block shall be reversibly enciphered when transmitted.

The format 1 PIN block shall be formatted as follows.

Bit

1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 64

C	N	P	P	P	P	P/T	T	T							
---	---	---	---	---	---	-----	-----	-----	-----	-----	-----	-----	-----	---	---

Wherein,

C = Control field: shall be binary 0001;

N = PIN length: 4-bit binary number with permissible values 0100(4) to 1100 (12);

P = PIN digit: 4-bit field with permissible values 0000 (zero) to 1001 (9);

P/T = PIN/Transaction digit: designation of these fields is determined by the PIN length field;

T = Transaction digit: 4-bit binary number with permissible values of 0000 (zero) to 1111 (15).

The transaction field is a binary number formed by [56-(N*4)] bits. This binary shall be unique (except by chance) for every occurrence of the PIN block and can, for example, be derived from a transaction sequence number, time stamp, random number or similar.

The transaction field should not be transmitted and is not required in order to translate the PIN block to another format since the PIN length is known.

Format 2 PIN block

The format 2 PIN block has been specified for local use with IC cards. The format 2 PIN block shall only be used in an offline environment and shall not be used for online PIN verification.

Format 3 PIN block

Format 3 PIN block construction

The format 3 PIN block is the same as format 0 PIN block except for the fill digits.

This PIN block is constructed by modulo-2 addition of two 64-bit fields: the plain text PIN field and the account number field. The formats of these fields are described in 1.4.2 and 1.4.3 respectively.

Plain text PIN field

The plain text PIN field shall be formatted as follows.

Bit

1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 64



Wherein,

C = Control field: shall be binary 0011;

N = PIN number: 4-bit binary number with permissible values of 0100 (4) to 1100 (12);

P = PIN digit: 4-bit field with permissible values of 0000 (zero) to 1001 (9);

P/F = PIN/Fill digit: designation of these fields is determined by the PIN length field;

F = Fill digit: 4-bit field, with values from 1010(10) to 1111(15), where the

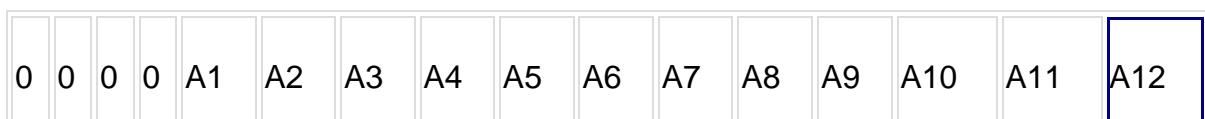
Fill-digit values are randomly or sequentially selected from this set of six possible values, such that it is highly unlikely that the identical configuration of fill digits will be used more than once with the same account number field by the same PIN device.

Account number field

The account number field shall be formatted as follows.

Bit

1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 64



For more details related to PIN Block format please refer to ISO 9564-1:2002(E).

Appendix 2 EPS PINBLOCK Format

String format: “1234”+ISN [6 bytes] +PIN [byte bit];

If PIN is less than 6 bytes, add “0” at the beginning ofPIN;

Convert the above data into BCD code and use TPK to encrypt the BCD code with DES/TDES algorithm.

Appendix 3 Registry

The table below is a registry of functions which mainly begin with “ro.fac.” and “persist.sys.”. The “ro.fac.” are read-only and “persist.sys.” can be both read and written.

System configuration name	Description
ro.fac.bootver	Boot version information.
ro.fac.hwver	Hardware version number. Main board- interface board.
ro.fac.mach	Product model name.
ro.fac.boardid	Boardid information, including the product model, hardware version number, etc.
ro.fac.conf.ver	Version information of configuration files.
ro.fac.pn	PN number.
ro.fac.sn	SN number.
ro.fac.prolin_debug_level	<i>Prolin_debug_level</i> information, this value is 0 and 1 for release system and debug system, respectively.
ro.fac.eth	Whether there is a network cable port.(0: does not exist; 1: exist)
ro.fac.usb.host	Whether there is a main device interface.(0: does not exist; 1: exist)
ro.fac.usb.device	Whether there is an USB device interface.(0: does not exist; 1: exist)
ro.fac.usb.otg	Whether there is an USB OTG interface.(0: does not exist; 1: exist)
ro.fac.leddt	Whether there is a LED digital tube.(0: does not exist; 1: exist)
ro.fac.keyboard	Key types.(0: have no key; 1: physical button; 2: touch-screen button)
ro.fac.buzzer	Whether there is a Buzzer module.(0: does not exist; 1: exist)
ro.fac.simsocket	The number of SIM card slot.
ro.fac.battery	Whether there is a battery.(0: does not exist; 1: exist)
ro.fac.coulomb_counter	Whether there is a coulombmeter. (None means it does not exist by default.)

ro.fac.wifi	Name of WiFi module.(None means it does not exist by default.)
ro.fac.bt	Name of <i>Bluetooth</i> module.(None means it does not exist by default.)
ro.fac.radio	Wireless module information and parameter information (optional). Used for multiple wireless modules; separated. (None means it does not exist by default.)
ro.fac.modem	Name of <i>Modem</i> module. (None means it does not exist by default.)
ro.fac.printer	Name of <i>Printer</i> module. (None means it does not exist by default.)
ro.fac.pcd	Name of PCD module. (None means it does not exist by default.)
ro.fac.sci	Name of IC card reader. (None means it does not exist by default.)
ro.fac.msr	Name of MSR reader. (None means it does not exist by default.)
ro.fac.videocard	Name of LCD module. (None means it does not exist by default.)
ro.fac.audioCARD	Name of audio card module. (None means it does not exist by default.)
ro.fac.touchesCREEN	Name of Touch-screen module. (None means it does not exist by default.)
ro.fac.sdHC	Supported specification, capacity range and speed level of SD card. (None means it does not exist by default.)
ro.fac.barCODE	Name of barcode module
ro.fac.camera_number	Number of camera(None means it has no camera)
ro.fac.camera_front	Name of front camera
ro.fac.camera_back	Name of back camera
ro.fac.barcode_cipher_chiP	Name of cipher chip (Used for camera scanning code.)
ro.fac.cipher_chip	Name of SM ciper chip(None means it does not exist by default.)
ro.fac.pcd.param1	PCD antenna parameter 1. (None means PCD does not exist.)
ro.fac.pcd.param2	PCD antenna parameter 2. (None means PCD does not exist.)

ro.fac.pcd.param3	PCD antenna parameter 3. (None means PCD does not exist.)
ro.fac.lcd.rotate	LCD clockwise rotation degrees. (0,90,180, 270)
ro.fac.security_level	Security level, the value can be 1 or 2. When updating firmware, the security level of the new firmware should be higher than the old one and Boot.
ro.fac.security_mode	Security mode, the value can be 0, 1 or 2. 0 represents production state, it only exists in the process of factory production; 1 represents factory state, it indicates non-authorization scheme; 2 represents security state, it indicates authorization scheme.
ro.security_version	Version information of PCI. (This information is available for models certified by PCI)
persist.sys.eth0.dhcp	DHCP is open or not. (true: open; false or none: closed)
persist.sys.eth0.ip	Ethernet ip address.
persist.sys.eth0.mask	Ethernet subnet mask.
persist.sys.eth0.gateway	Ethernet gateway.
persist.sys.eth0.speed	Ethernet network port speed.(<i>eth_auto</i> : automatic configuration; <i>eth_10mhd</i> : 10M half-duplex; <i>eth_10mfd</i> : 10M full-duplex; <i>eth_100mhd</i> : 100M half-duplex; <i>eth_100mfd</i> : 100M full-duplex)
persist.sys.prolin	Prolin system version information.
persist.sys.language	System language.
persist.sys.backlighttime	Time interval to enter screensaver mode from normal mode automatically. (The time interval ranges from 0 to 7200 sec. It will be set to 60 sec by default if screensaver time is not more than 30 sec; it is the set value if more than 30 sec. The system will automatically enter screensaver mode if there is no operation for specified time. 0 means closing the screensaver mode.)
persist.sys.sleeptime	Time interval to enter sleep mode from screensaver mode automatically. (The valid value ranges from 0 to 60s, and the default value is 0, which means the auto-enter sleep mode is disabled)
persist.sys.sleepwaittime	Wait time for user process to deal with events before the system enters sleep mode. (The value range is from 0 to 15s, the default value is 1, which means the user is notified that system will sleep and is

	given 1 second to handle events before system goes to sleep.)
persist.sys.delayshutdown	After long pressing the power key or OsReboot() is called, the system will power off after <i>persist.sys.delayshutdown</i> seconds. (The <i>persist.sys.delayshutdown</i> value ranges from 0~600 seconds, and the default value is 0.) It is invalid when OsPowerOff() is called.
persist.sys.confirmshutdown	Setting it to 1 means the screen will display the shutdown confirmation interface after pressing the power key for 3 seconds, press “Enter” key to continue the shutdown or press “Cancel” key to cancel the shutdown. Setting it to 0 means the terminal will power off immediately after pressing power key for 3 seconds. By default, this value is 0 for Prolin-2.4 system and Prolin-phoenix-2.5 and 1 for Prolin-cygnus-2.6 and higher version.
persist.sys.sound.enable	Whether to open the keypad tone. (False means closed; true means open) The setting will take effect after reboot.
rt.app.key.tone	The status of current key tone (0 means off; 1 means open. It is not recommended to modify the keyword. Only prolin-peng-2.8 supports this keyword)
persist.sys.keypad.duty_cycle	The duty cycle of PWM (valid range is from 1 to 99). It is used to adjust the buzzer volume when pressing the button, and the setting will take effect after reboot.
persist.sys.key.backlight	Whether the button backlight is open or not. (0 means closed; 1 means open)
rt.app.key.backlight	The status of current key backlight (0 means off; 1 means open. It is not recommended to modify the keyword. Only prolin-peng-2.8 supports this keyword)
persist.sys.lcd.brightness	LCD brightness. (The valid brightness ranges from 1 to 10 and the higher, the brighter.)
persist.sys.lcdsaverbrightness	LCD brightness in screensaver mode(This value ranges from 0 to 10, the effect of setting this value is the same as calling OsScrBrightness.This variable only applies to auto-screensaver and OsSysSleepEx(1), and it will take effect immediately after setting this value.)
persist.sys.sound.volume	Audio/video sound volume. (The valid volume

	ranges from 0 to 4.)
persist.sys.auto.mount.enable	Mount u disk or sd card automatically or not.
persist.sys.xcb.enable	Open or close XCB service. (0: close; 1: open)
persist.sys.xcb.tcp.port	Network port number of XCB service. (It is the network port number when using network; it is -1 when using other ways.)
persist.sys.xcb.rs232	USB or COM of XCB service. (When using USB, it is /dev/ttydev; when using COM port, it is /dev/ttyAMA* (* means the represented value, refer to [ro.kernel.CONSOLE]: [ttyAMA3, 115200] as it may vary from model to model); it is null when using network.)
persist.sys.autowaketime	<p>System sleeps for <i>autowaketime</i> seconds and then automatically wakes up.</p> <p>If <i>autowaketime</i> has not been set, it means the autowake function is disabled, and the system will not wake up automatically.</p> <p>If <i>autowaketime</i> is set to less than 0, the system will not wake up automatically.</p> <p>If the <i>autowaketime</i> is set to be greater than 0 and less than 128s, then systems will automatically wake up after 128s;</p> <p>If the <i>autowaketime</i> is set to be greater than 12*60*60s, which is 12 hours, the system will automatically wake up after 12 hours.</p> <p>If application calls OsSysSleep() or OsSysSleepEx() to enter the sleep state, this variable will not work, and the system will not wake up automatically.</p>
persist.sys.console.enable	Open/close console (0: close, 1: open)
persist.sys.wifi.roam.dhcp	Enable or disable the function of DHCP when the terminal is in roaming state. (0: disable, 1: enable) and it is enabled by default.
persist.sys.tm.imei	The IMEI value of wireless module
persist.sys.timezone.tz	Set time zone. For tz values of various regions, please refer to standard Linux tz value. For more information on how to set time zone, please refer to "Prolin Application Development Manual"
persist.sys.continue.print	Enable/disable the function of continuing to print after a shutdown. (0: disable, 1:enable)
persist.sys.autouload.ena	Enable/disable USB flash driver auto-downloading

ble	function at the POS terminal startup.(0: disable, 1:enable)
persist.sys.hostname	Terminal host name (If the host name is not filled, it means this term does not exist. Developers can use the internet or Linux standard method to acquire the hostname. The default hostname is “product name-series number”. After this value is being set, it will not take effect until the terminal is rebooted, and the display of hostname requires the terminal to connect to the internet again.)
persist.sys.wnet.version	Acquire wireless module firmware version information.
persist.sys.autostartup.enable	Enable/disable the function of starting the terminal automatically when terminal is being charged. (0: disable, 1:enable, and the default startup method will be used when this value is not filled in. Prolin-2.4 and Prolin-phoenix-2.5 doesn't support this configuration.)
persist.sys.enable.debug	When setting it to 1, if PED tampers, XCB service will be enabled; if POS enters TM, XCB service will also be enabled, and after entering the application, XCB service will restore to the original status which is before POS entering TM.
persist.sys.mainapp.restart	Whether to restart the application automatically after MAINAPP abnormally crashes. (1 represents Yes, and the application can be restarted 10 times at most; 0 represents No)
rt.sys.mainapp.restart	Whether to restart the main application automatically after a normal exit. (1 represents Yes; 0 represents No)
persist.sys.ped.keypad.mask	When inputting PIN, mask the specified keys on soft keypad, and this function has no effect on physical key. Currently, it only supports passing “cancel” into QR55, and masking “cancel” key while inputting PIN.
persist.sys.ped.keypad.type	<p>Switch the soft keypad style of inputting PIN, after setting the registry value, it will take effect on the next time the PIN is inputted.</p> <ul style="list-style-type: none"> when it is set to “0”, the default soft keypad will be used; when it is set to “1”, the 1PIN keypad customized for Q20 will be used (“displaying on top and inputting at bottom” soft keypad in disordered mode);

	<ul style="list-style-type: none"> ● when it is set to “2”, the 1PIN keypad customized for D220 will be used; ● When it is set to “3”, the 2PIN keypad customized for D220 will be used; ● when it is set to “4”, the 2PIN keypad customized for Q20 will be used; ● When it is set to “5”, the 3PIN keypad customized for Q20 will be used; ● when it is set to “6”, the 4PIN keypad customized for Q20 will be used; ● when it is set to “7”, the 5PIN keypad customized for Q20 will be used; ● when it is set to “8”, the 6PIN keypad customized for Q20 will be used.
persist.sys.batterylow.oftcnt	Record the times of auto shutdown caused by low battery.
persist.sys.reboot.cnt	Record the times of calling OsReboot().
persist.sys.powerkey.oftnt	Record the times of shutdown by long pressing on the power key.
persist.sys.dns.static	Whether to use static DNS server address. It defaults to “0”, that is, if DHCP is enabled, DNS server address acquired by DHCP will be used; If it set to “1”, OS will ignore the DNS server address acquired by DHCP and use the static DNS server address set by users, and DHCP will not modify the static DNS server address.
persist.sys.xcb.installapp.lock	Whether to disable XCB to install AIP/AUP. If it is set to “1”, the function of installing AIP/AUP through XCB is disabled; if it is set to “0”, the function of installing AIP/AUP through XCB is enabled.
persist.sys.ped.pin.icon.align	<p>Whether to enable the dynamic center display function of PIN ICON.</p> <ul style="list-style-type: none"> ● when it is set to “1”, while inputting PIN, the foreground image which is passed in after calling OsPedSetPinIconLayout() will be centered and displayed dynamically, and the background image will not be displayed at the same time; ● when it is set to “0”, the foreground image will restore to normal display, and the default value is “0”.
persist.sys.usbd.mode	<p>Set the mode for USB device, and it will take effect after restart.</p> <ul style="list-style-type: none"> ● When it is set to “1”, the POSVCOM single

	<p>channel mode with old Product ID and old Vender ID will be used, and it's the default mode;</p> <ul style="list-style-type: none"> ● When it is set to “2”, CDC single channel mode will be used; ● When it is set to “3”, the POSVCOM single channel mode with new Product ID and old Vender ID will be used; ● When it is set to “4”, ECM mode will be used, after the device is connected to the upper computer, a USB0 virtual network card will be enumerated at both sides; ● When it is set to “5”, the dual-channel mode will be used; ● When it is set to “6”, CDC ACM + CDC ECM mode will be used, and this mode is compatible with the USB dual-channel mode and has a virtual network port usb0; ● When it is set to “7”, the POSVCOM three-channel mode will be used; ● When it is set to “8”, the CDC ACM three-channel mode will be used.
persist.sys.ped.reboot.cycle	Whether to restart the terminal regularly. <ul style="list-style-type: none"> ● When it is set to an integer between 1 and 48, the set value (Unit: hour) will be used as the period for restarting the terminal. This feature will not take effect until the registry is set successfully and the terminal is restarted. This value defaults to "0", the function of restarting the terminal regularly will be disabled at this time. ● When it is set to more than 48h, the system will set the restart period to 48h. When the continuous running time of the system reaches the restart period, a countdown prompt interface will appear in advance of 5s, and then the terminal will be automatically restarted.
persist.sys.ped.pinwait.retain	Whether it is forbidden to press the cancel key to exit the PIN waiting interface when entering PIN. <ul style="list-style-type: none"> ● The default value is "0", that is, pressing the cancel key to exit the pin waiting interface is allowed. ● If the value is set to "1", it is forbidden to press the cancel key to exit the PIN waiting interface when entering PIN. After setting this value, it will take effect when entering the pin waiting

	interface next time.
persist.sys.tm.cpu	CPU information of the terminal.
persist.sys.tm.flash	Flash size of the terminal. (For example, "128MB ")
persist.sys.tm.ram	Memory size of the terminal. (For example, "64MB ")

Appendix 4 Validity of Models and Contents

Allowing for the configuration differences for different models, some OSAL interfaces may be invalid for certain kinds of model. For the validity of different models and chapters, please refer to the table below.

Note: Whether there is Wireless module, Modem module or Ethernet module depends on the model configuration. (Refer to the POS PN number)

Chapters	S300	S800	S900	S920	D200	P ^X X	Q80	D220
Thread	√	√	√	√	√	√	√	√
System Function	√	√	√	√	√	√	√	√
Encryption and Decryption	√	√	√	√	√	√	√	√
PED	√	√	√	√	√	√	√	√
LCD	240*32 0, rotate 90° in 90° in clockwi	320*240 , rotate 90° in clockwis	240*320 , rotate 90° in clockwis	240*320, rotate 90° in clockwise direction	240*32 0, no rotation	800*480 no rotation	480*80 0 no rotation	480*80 0 no rotation

	se direc tio n	direction	direction					
Keyboard	√	√	√	√	√	√	√	NA
Touch Screen	√	NA	√	√	NA	√	√	√
Signature Pad	√	NA	√	√	NA	√	√	√
Printer	NA	√	√	√	NA	NA	√	NA
Font Library	√	√	√	√	√	√	√	√
Code	√	√	√	√	√	√	√	√
MSR	√	√	√	√	√	√	√	√
IC Card Reader	√	√	√	√	√	√	√	√
RF Reader	√	√	√	√	√	√	√	√
Communi	PORT	PORT_	PORT_	PORT_U	PORT	PORT_C	PORT	PORT

Function		_COM	COM1	COM1	SBDEV	_COM	OM2	_COM	_USB
Port		1	PORT_	PORT_	PORT_U	1	PORT_U	1	DEV
	USB DEV	PORT	COM2	USBDE	SBHOST	PORT	SBDEV	PORT	PORT
	PORT _USB HOST	PINPAD	PORT_	V	—	USBD	PORT_U	_PINP AD	_USB HOST
	PORT _USB HOST	PORT_	USBHO	USBHO	EV	PORT	SBHOST	PORT	PORT
	PORT _USB HOST	USBDE	ST	ST	—	PORT	—	_USB DEV	_USB DEV
	PORT _USB HOST	V	PORT_	USBHO	—	HOST	—	PORT	PORT
	PORT _USB HOST	USBHO	ST	ST	—	—	—	_USB HOST	_USB HOST
MODEM	N/A	√	N/A	N/A	N/A	N/A	NA	√	NA
Network Communication	√	√	√	N/A	N/A	√	√	√	√
Network Configuration	√	√	√	N/A	N/A	√	√	√	√
GPRS/CDMA	N/A	√	√	√	N/A	√	√	√	√
						(optional)			

WiFi	N/A	N/A	√	√	√	√	√	√
File System	√	√	√	√	√	√	√	√
System Information	√	√	√	√	√	√	√	√
Audio	√	√	√	√	N/A	√	√	√
Power Management	N/A	N/A	√	√	√	√	√	√
Barcode	N/A	N/A	√	N/A	N/A	NA	NA	NA
Bluetooth	N/A	N/A	N/A	√	√	√	√	√