LoRaGW SDK 使用手册

修改历史

版本	修改内容	时间	作者
2.3.0	初版	2018-11-22	和璞
2.4.0	更新部分描述	2019-03-25	陈文兵、和璞

目录

1. 引言

1.1. 文档目的

LoRaGW SDK 是 Link WAN 为合作伙伴（网关设备厂商）提供的 LoRa 网关 SDK。本文档 LoRaGW SDK 的使用参考手册。

1.2. 版权声明

1.3. 名词解释

本文档使用到的名词如下：

Link WAN：广域网连接管理平台，支持设备接入、设备管理、网络管理、数据安全

LoRa Server：LoRa 服务端

LoRaGW： LoRa Gateway 的简写

packet_forwarder： LoRa 网关协议栈

pktfwd：packet_forwarder 的缩写，表示 LoRa 网关协议栈

mqtt：负责 LoRa 网关内和 Link WAN Server 连接的模块

watchdog：负责 LoRa 网关进程运行监护的模块

monitor： 负责 LoRa 网关系统状态监护的模块

libalilog: 阿里巴巴提供的日志模块

iotkit：阿里巴巴提供的 IoT 连接套件

2. 网关 SDK 模块和目录

2.1. 模块介绍

SDK 提供 LoRa 网关相关的基础库、功能模块，以及 LoRa 网关协议栈（packet_forwarder）和连接 Link WAN 平台能力（mqtt）等。

目前包含的内容如下：

模块	类型	描述
libalilog	基础库	阿里巴巴提供的日志基础库，提供源代码。
libipcbus	基础库	基于 dbus 实现的模块通讯库，提供源代码。
libwatchdog	模块库	watchdog 功能的调用库。watchdog 可以在线程级别检测网关的运行状态，当线程运行异常时，重启网关，提供源代码。
watchdog	执行程序	watchdog 功能监护程序，提供源代码。
iotkit	IoT 套件库	阿里巴巴提供的物联网连接套件，mqtt 模块使用该套件连接 Link WAN LoRa Server，提供源代码。
mqtt	执行程序	连接 Link WAN LoRa Server 模块，提供源代码
monitor	执行程序	检测网关运行状态模块，提供源代码
pktfwd	执行程序：packet_forwarder 的缩写	在 semtech 官方的 LoRa 协议栈上提供相关的功能扩展 patch。合作伙伴需要 merge 自己修改的部分到 SDK 中，然后编译自己的 packet_forwarder。
remote_debug	执行程序	远程调试功能模块，在 Link WAN 后台开启该功能（网关设备商暂不具有权限），使用三元组远程 SSH 登录网关，提供源代码。
update-deamon	执行程序	OTA 守护进程（OTA 机制和原来有较大改动，需要单独运行该 OTA 守护进程），提供源代码。
security	deploy_sst：执行程序 irot_service：执行程序 keychain_service：执行程序 libkeychain.a：静态库 libsst.a： 静态库	阿里巴巴提供的安全存储方案，包括进程 deploy_sst、irot_service、keychain_service。 目前 SDK 中只有 mqtt 使用了安全存储保存一些关键数据。 libkeychain.a 和 libsst.a 是提供的静态库，合作伙伴可以集成该功能到其他模块内，提供源代码。

LoRaGW SDK 使用的第三方库开源库包括：

- libcjson：开源的 cjson 库。
- dbus 和 expat： security、update、libipcbus 和 watchdog 功能基于 dbus 实现，这部分依赖的库包括 dbus 库和 libexpat。

- hiredis ： redis 库。

2.2. 目录结构

SDK 的目录结构如下：

```
$tree -L 2
.
|-- build.sh
|-- docs
|   |-- loragw_sdk_manual v2.4.0.pdf
|   |-- 阿里云 Link WAN 网关 SDK 设备接入快速开始.pdf
|   `-- 阿里云 Link WAN 网关接入规范 v1.8.0.pdf
|-- external
|   |-- cJSON-1.5.5.tar.gz
|   |-- dbus-1.10.18.tar.gz
|   |-- hiredis-0.13.3.tar.gz
|   |-- libexpat-2.2.3.tar.bz2
|   |-- nopoll-0.4.4.tar.gz
|   `-- openssl-1.0.2.tar.gz
|-- libraries
|   |-- iotkit-embedded
|   |-- ipc-bus
|   `-- libalilog
|-- LICENSE
|-- make.settings
|-- modules
|   |-- monitor
|   |-- mqtt
|   |-- pktfwd
|   |-- remote_debug
|   |-- security
|   |-- update-deamon
|   `-- watchdog
|-- README.md
|-- scripts
|   `-- dbus-setup
`-- tools
    `-- ota_ref_env
```

1. build：在编译过程中产生的输出目录。
2. build.sh：编译 SDK 的命令脚本。
3. external: SDK 使用的第三方开源库，包括：cJSON、dbus、hiredis、libexpat、nopoll、和 openssl。
4. libraries：阿里巴巴提供的库，包括：iotkit、ipc-bus 和 libalilog。

5. docs：SDK 文档，包括 SDK 使用手册，SDK 开始开始手册和网关接入规范。
6. make.settings：SDK 编译配置脚本。
7. modules：LoRa Gateway 模块，包括：monitor、mqtt、pktfwd、remote_debug、security、update-deamon 和 watchdog。
8. tools：OTA 增强版签名工具及打包工具。

3. 网关 SDK 编译

3.1. 配置文件说明

make.settings 为编译配置脚本，配置选项说明如下：

选项	说明	默认值
PATH	系统环境变量，需要合作伙伴修改为本地 toolchain 的路径	请添加本地的 toolchain 路径到 PATH。
BUILDHOST	编译 toolchain 的 HOST	arm-linux-gnueabihf，请修改为本地 toolchain 的 HOST。
BUILDROOT	SDK 的根目录	$(pwd)，无需修改。
BUILDTMP	SDK 编译的临时目录	默认值为 SDK 根目录下 tmp 文件，无需修改。
BUILDOUT	SDK 编译结果输出文件	默认值为 SDK 根目录下 out 文件，无需修改。
BUILDOUTPUT	SDK 编译依赖文件	默认值为 SDK 根目录下 build 文件，无需修改。
dbus_address	SDK dbus 监听地址	不需要修改。原系统没有运行 dbus 的厂商可以忽略该选项。
ENABLE_ALILOG	是否使能能 libalilog	默认为 true，打开 Log 日志功能，合作伙伴可以在调试阶段可选择关闭。
ENABLE_WATCHDOG	是否使能 watchdog 功能	默认 true，打开 watchdog 功能，合作伙伴可以在调试阶段可选择关闭。
ENABLE_MONITOR	是否使能 monitor 功能	默认 true，打开 monitor 功能，合作伙伴可以在调试阶段可选择关闭。
BUILD_PKTFWD_BIN	是否编译 packet_forwarder	默认 true，SDK 中 packet_forwarder 基于 semtech 官方代码已打上阿里修改的相关 patch。合作伙伴可以根据自己的 LoRa 协议栈进行 merge，然后选择是否编译。
ENABLE_ADVANCED_	是否开启增强 OTA 功能	默认 true，采用增强版 OTA 功能。

OTA		OTA 有两种方式：基础版和增强版。参考 4.2.7 章节：OTA 增强。
ENABLE_ADVANCED_SECURITY	是否开启安全存储服务	默认 true， 合作伙伴可以在调试阶段可选择关闭。

3.2.SDK 编译

a) **Setp 1：设置环境变量**

修改 make.settings，参考下面设置，指定你的 toolchain 路径和编译 HOST：

export PATH=/home/XXXXXXX/XXXX/bin:$PATH

export BUILDHOST=arm-linux-gnueabihf

如果是 openwrt toolchain 请额外指定：

export STAGING_DIR=/your/toolchain/staging_dir/

b) **setp 2 ： 设置编译模块配置**

export ENABLE_ALILOG=true

export ENABLE_WATCHDOG=true

export ENABLE_MONITOR=true

export BUILD_PKTFWD_BIN=true

export ENABLE_ADVANCED_OTA=true

export ENABLE_ADVANCED_SECURITY=true

在初始调试阶段可以选择关闭 ENABLE_ALILOG、ENABLE_WATCHDOG 和 ENABLE_MONITOR 选项。正式提测阶段，这三个模块选项都需要打开，集成运行。

BUILD_PKTFWD_BIN：可以根据 packet_forwarder 的 merge 情况选择是否打开。

c) **setp 3： merge mqtt 到 SDK 代码**

mqtt 需要合作伙伴实现本地的接口，gwiotapi.c 需要合作伙伴去实现。请将本地实现的 gwiotapi.c 拷贝到 modules/mqtt/sample_libgwiotapi 目录，然后重新编译。目前源码中包含了一套 gwiotapi 的参考实现，建议合作伙伴在参考代码上适配实现。

d) **setp 4： merge pktfwd 代码到 SDK**

SDK 中 packet_forwarder（包括 libloragw）模块，是阿里巴巴基于 semtech 官方代码修改的。阿里巴巴还提供了相关的 patch list。这部分需要合作伙伴和自己的 packet_forwarder 代码完成 merge。

注意：完成 merge 时，请保证 packet_forwarder 和 lora_pkt_fwd 目录下 Makefile 不被修改。

e) **setp 5：执行编译**

1) 完整编译

./build.sh all

2) 单模块编译

$./build.sh

Invalid command:

Usage:

build.sh lib [third libalilog iotkit ipc-bus]

build.sh libraries

build.sh module [watchdog mqtt pktfwd monitor remote_debug]

```
build.sh modules
build.sh all
build.sh clean
```

4. 网关 SDK 集成和运行

4.1. SDK 编译输出及集成

SDK 编译的结果输出在 out 目录下：

```
$tree out/
out/
├── bin
│   ├── auth_key.json
│   ├── dbus-daemon
│   ├── deploy_sst
│   ├── cron_watchdog.sh
│   ├── dev_info.json
│   ├── filter_conf.json
│   ├── global_conf.json
│   ├── irot_service
│   ├── keychain_service
│   ├── local_conf.json
│   ├── lora_pkt_fwd
│   ├── mbusd
│   ├── mbusd.conf
│   ├── monitor
│   ├── mqtt
│   ├── publicKey.pem
│   ├── remote_debug.ini
│   ├── reset_lgw.sh
│   ├── root.pem
│   ├── sshd_agent
│   ├── update-deamon
│   └── watch_dog
├── etc
└── lib
    ├── libalilog.so
    ├── libcjson.so
    ├── libdbus-1.so
    ├── libexpat.so
    ├── libgwiotapi.so
    ├── libkeychain.a
```

```
├──── libsst.a
└──── libwatchdog.so
```

lib 目录下包含了需要集成的库文件。将这些库拷贝到文件系统，并保证能够被系统自动搜索到即可。

lib 库文件说明：

文件	说明	调用
libalilog.so	阿里巴巴提供的日志库	头文件为 log.h，具体调用参考头文件说明。
libcjson.so	开源的 cjson 库	源码在 SDK 下 external 路径。
libdbus-1.so	开源 dbus 的动态库	源码在 SDK 下 external 路径，需要头文件和其他 dbus 的 bin 文件请编译完成后，在 SDK build 目录查找。
libexpat.so	dbus 依赖的 libexpat 库	源码在 SDK 下 external 路径。
libgwiotapi.so	合作伙伴需要适配的 gwiotapi。	目前阿里巴巴提供了一套参考代码，源码在 modules/mqtt/sample_libgwiotapi。
libwatchdog.so	watchdog 功能库	源码在 modules/watchdog，API 请参考 SDK 中 watch_dog_export.h。
libsst.a	安全存储 sst 功能库	目前不提供源码，API 请参考 SDK 中，sst.h
libkeychain.a	安全存储 keychain 功能库	目前不提供源码，API 请参考 SDK 中，keychain.h

bin 目录包含了需要集成的程序执行文件。具体如下。

mqtt 运行相关文件：

文件	说明	存放路径
mqtt	mqtt 进程运行文件	合作伙伴自定义 运行： ./mqtt &
libgwiotapi.so	mqtt 依赖库，合作伙伴集成实现。	系统库搜索路径内。
auth_key.json	被 libgwiotapi 调用，保存加密过的网关三元组信息，加密机制请参考 libgwiotapi 的参考实现。	默认路径为：mqtt 运行路径。 合作伙伴可自定义，这部分是参考实现，只要适配好 libgwiotapi，不一定需要 auth_key.json 文件。但是建议网关三元组信息以单独文件存储，方便后续 OTA 更换三元组信息。
local_conf.json	被 libgwiotapi 和 lora_pkt_fwd 调用，保存网关的 EUI。	默认路径为：mqtt 运行路径， 合作伙伴可自定义。注意路径配置，libgwiotapi 和 lora_pkt_fwd 要都能访问到。
global_conf.json	被 libgwiotapi 和 lora_pkt_fwd 调用，用于	合作伙伴可自定义。注意路径配置，libgwiotapi 和 lora_pkt_fwd

	保存网关 LoRa 协议栈相关配置。	要都能访问到。
dev_info.json	被 libgwiotapi 调用，保存网关的合作伙伴信息。合作伙伴需适配该文件内容。	默认路径为：mqtt 运行路径，合作伙伴可自定义。这部分是参考实现，只要适配好 libgwiotapi，不一定需要 dev_info.json 文件。

packet_forwarder 运行相关文件：

文件	说明	默认路径
lora_pkt_fwd	LoRa 网关协议栈	合作伙伴自定义。
filter_conf.json	节点过滤设置，网关合作伙伴不需要修改该文件。	默认路径为 lora_pkt_fwd 运行目录。
local_conf.json	被 libgwiotapi 和 lora_pkt_fwd 调用，保存网关的 EUI。	默认路径为 lora_pkt_fwd 运行目录。
global_conf.json	被 libgwiotapi 和 lora_pkt_fwd 调用，用于保存网关 LoRa 协议栈相关配置。	默认路径为 lora_pkt_fwd 运行目录。
reset_lgw.sh	reset semtech 基带芯片的参考脚本	合作伙伴可以根据实际情况进行集成。

dbus 运行相关文件

文件	说明	默认路径
dbus-daemon	dbus 守护进程	合作伙伴自己管理，建议存放在系统 PATH 路径。
mbusd	阿里巴巴提供的 dbus 启动脚本。	合作伙伴自定义。 运行： ./mbusd
mbusd.conf	dbus 运行配置脚本，合作伙伴不需要修改。	mbusd 脚本使用了./mbusd.conf 进行 dbus 配置，默认情况下请保证 mbusd.conf 和 mbusd 在同一个目录。

watchdog 运行文件：

文件	说明	路径
watch_dog	watchdog 监护进程，依赖 dbus 库和 libalilog。	合作伙伴自定义。 运行： ./watch_dog &
cron_watchdog.sh	监护 watchdog 运行状态的 cron 任务脚本	需要和 watch_dog 在同一目录。如果使用硬狗看护 watchdog 进程，则不需要该脚本。

monitor 运行文件：

文件	说明	路径

monitor	monitor 运行进程，依赖 libalilog	合作伙伴自定义。 运行： ./monitor &

远程调试运行文件：

文件	说明	路径
sshd_agent	远程调试运行进程，在需要时，由 mqtt 进程拉起。	合作伙伴自定义，但必须和 mqtt 在同一目录。
root.pem	远程调试进行需要的秘钥文件	合作伙伴自定义，但必须和 mqtt 在同一目录。
remote_debug.ini	远程调试的初始化文件	合作伙伴自定义，但必须和 mqtt 在同一目录。

增强 OTA 进程运行文件：

文件	说明	路径
update-deamon	OTA 机制和原来有较大改变，update-deamon 为 OTA 守护进程	合作伙伴自定义，最好在其他模块启动前运行
publicKey.pem	update-deamon 进行签名验证的公钥	必须和 update-deamon 在同一路径下。 每个厂商提供一对公钥\私钥，请联系阿里巴巴获取公钥。

安全存储运行文件：

文件	说明	路径
deploy_sst	安全存储的初始化进程	合作伙伴自定义，但必须在 irot_service 和 keychain_service 之前运行 运行： ./deploy_sst -f 或者 ./deploy_sst
irot_service	安全存储(支持 sst 模式)后台进程	合作伙伴自定义，但必须在 keychain_service 之前运行 运行： ./irot_service &
keychain_service	安全存储（支持 keychain 模式）后台进程	合作伙伴自定义 运行： ./ keychain_service &

模块启动顺序

首先启动 dbus，然后启动 watchdog 和安全存储相关进程，接着启动 update-deamon，最后启动 monitor，mqtt，pktfwd 模块。

参考的执行脚本如下：

./ mbusd

sleep 1

./watch_dog &

sleep 1

./deploy_sst -f

sleep 1

./irot_service &

sleep 1

./keychain_service &

sleep 1

./update-deamon &

sleep 1

./monitor &

sleep 1

./mqtt &

sleep 3

./lora_pkt_fwd

这是一个全功能启动的参考实现，请合作伙伴依据实际情况进行删减。

4.2. 模块功能介绍和注意事项

4.2.1 watchdog 功能介绍和注意事项

4.2.1.1 watchdog 功能介绍

功能	描述	备注
线程进行喂狗操作	Watchdog 监护该线程运行状态，当线程喂狗超时，watchdog 会 reboot 网关	处理逻辑：喂狗线程在不断执行一个 loop，当喂狗超时，说明该线程的 loop 执行出

		现问题。
线程取消喂狗	线程取消喂狗操作，watchdog 不再监护该线程	取消喂狗后，watchdog 将无法感知线程运行异常。
检测 dbus 运行状态	当 watchdog 检测不到 dbus-daemon 时，watchdog 会 reboot 网关	dbus-daemon 是 dbus 连接的保障，当 dbus 通讯出现问题时，watchdog 机制将不工作，这时将重启网关。
linux 系统看护 watch_dog 进程状态	watchdog 运行出现异常时，网关 reboot	使用硬狗监控 watchdog 进程，watchdog 会在每分钟进行一次喂硬狗操作（/dev/watchdog）。

4.2.1.2 watchdog 集成注意事项

1. watchdog 必须要以 root 权限运行。
2. 完善的 watchdog 功能，必须使用硬狗/dev/watchdog，否则 watchdog 机制将无法得到保障。使用硬件看门狗时请保证一下几点：
   - 硬件看门狗文件路径名为：/dev/watchdog。如果硬件看门狗设备文件路径和/dev/watchdog 不一致，请用 ln 命令将/dev/watchdog 链接到看门狗设备。
   - 保障/dev/watchdog 能被 watchdog 使用。如果硬件看门狗已经被合作伙伴使用，并且不支持用户空间多句柄打开。建议将硬件看门狗让给 watch_dog 进程使用，这样可以尽大可能的监控 LoRa 网关软件的运行状态。
   - 为了保证硬件看门狗健壮性，防止 watch_dog 进程被杀后无法看护系统进程，请使能内核编译选项 CONFIG_WATCHDOG_NOWAYOUT。
3. watchdog 在 dbus 启动后运行。
4. watchdog 对 dbus 版本依赖。SDK 中 dbus 版本为 1.10.18，合作伙伴可能会将 SDK 中 dbus 版本进行替换。如果替换的版本过低，可能会造成 watchdog 运行异常。原因是较低版本的 dbus 不支持“org.freedesktop.DBus”服务，导致 watchdog 无法通过该服务请求 dbus_daemon 的运行 PID。如果合作伙伴遇到类似现象，请自行修改 watchdog 的 Makefile 文件，关闭 DBUS_SUPPORT_SERVER_PID，重新编译 watchdog。

备注：如果网关无硬狗设备，watchdog 仍能运行，但是 watchdog 机制将无法保障，因为 watchdog 退出时，整个看护机制也就无效了。对于无硬狗的情况，我们推荐合作伙伴集成 cron 服务到网关，使用 cron 监护 watchdog。cron 监护 watchdog 的功能，在 watchdog 中已有实现，合作伙伴只需保证 cron 在 linux 启动后正常运行即可。

watchdog 提供了 libwatchdog.so，合作伙伴可以选择将自己的模块加入到 watchdog 监护中，这个过程不需要额外配置，详细 API 请查看 watch_dog_export.h。

4.2.1.3 cron 服务集成

当无法使用硬件看门狗监护 watchdog 进程运行状态时，我们推荐合作伙伴集成 cron 服

务到网关，使用 cron 定时去检测 watchdog 进程的运行状态，这样能够最大限度的保障 watchdog 机制的可靠性。

在 watchdog 中，我们在/etc/cron.d/（系统必须集成 cron 服务）目录下写入了 gateway_watchdog 文件，该文件配置 cron 每分钟去执行一次 cron_watchdog.sh 脚本。

cron_watchdog.sh 内容如下：

```
#!/bin/bash
pgrep watch_dog;
if [[ $? -ne 0 ]];
then
/sbin/reboot;
fi
```

通过 cron 检测 watch_dog 进程的运行状态，如果 watch_dog 不在运行，则重启系统。

注意：开启 cron 检测 watchdog 功能，一定要注意 cron 和 watch_dog 进程启动的时间间隔（watch_dog 不要晚于 cron 超过一分钟）。

4.2.1.4 watchdog 的扩展能力

目前喂狗超时后，watchdog 会对网关进行重启。但是 watchdog 具备了第二个超时处理机制能力——重新拉起喂狗超时的进程。使用 API：thread_feeddog_with_operation 可以选择喂狗超时后出的操作。即选择喂狗超时后重启网关后者仅仅重新拉起喂狗超时的进程。

拉起进程的机制依赖 dbus 的 service 服务。集成方法：编写一个进程 service 描述，并且拷贝到 dbus_daemon 配置文件的 service 目录。例如 dbus_daemon 运行配置 service 目录为:<servicedir>/tmp/var/run/mbusd/service</servicedir>(具体的配置请自行确定)。

以 watchdog 为例，为 watchdog 编写一个 service 描述：watchdog.service，其内容如下：

```
[D-BUS Service]
Name=iot.gateway.watchdog
Exec=/your_watchdog_bin_dir/watch_dog
```

将 watchdog.service 文件拷贝到/tmp/var/run/mbusd/service。进程被拉起时，类似调用下面命令：/your_watchdog_bin_dir/watch_dog，如果你的程序需要 cd 到某个目录再执行 bin，这时候需要额外注意目录问题。

4.2.2 远程调试模块注意事项

远程调试模块和其配置文件，一定要放在 mqtt 程序所在的同目录。

1. 注意给 sshd_agent 增加可执行权限：

```
chmod a+x sshd_agent
```

2. 注意给 remote_debug.ini、root.pem 增加可读权限：

```
chmod a+r root.pem
chmod a+r remote_debug.ini
```

远程调试模块在服务端开启功能后，会被 mqtt 自动拉起，这部分 mqtt 中已经完成实现，合作伙伴只需保证 sshd_agent 和 root.pem、remote_debug.ini 在 mqtt 所在目录及权

限正确即可。

注意：远程调试功能服务和 gwiotapi.c 中需要实现的 ssh 开关为不同实现，不要混淆。ssh 开关是指网关本地的 ssh 服务，例如 dropbear 等。

4.2.3 alilog 功能和使用注意事项

libalilog 提供本地日志服务，可以保存日志到文件和数据库。网关 mqtt、monitor 和 watchdog 目前都使用了 libalilog。

合作伙伴在使用 libalilog 要注意：目前 libalilog 输出到文件、数据库日志的同时，会将日志输出到 stdout 中。在默认情况下 stdout 关联了系统的终端设备，通常就是网关的串口，因此系统调用 libalilog 输出日志速度不要超过串口速率（一般情况下约 14KB/s）。如果打印日志超过该速率，可能会造成日志打印阻塞，影响程序运行效率。

4.2.4 monitor 功能介绍

网关 monitor 是 LoRa 网关 Linux 系统上执行的一个网关状态监视工具，该工具会定时采集系统信息，网关信息以及报警信息上报给服务端。

具体功能点描述如下：

- 获取网关文件系统空间使用率
- 获取最近系统启动时间
- 获取网关回传网方式
- 获取网关 IP 地址
- 获取网关网络延时
- 获取网关网络收发包总流量
- 获取网络传输速率
- 获取网关 SX1301 异常告警
- 获取网关重启原因

monitor 目前不需要合作伙伴实现任何 API，保证 monitor 能够在网关运行即可。

4.2.5 OTA 基础版功能介绍和注意事项

OTA 基础版功能实现了 OTA 功能，采用了阿里巴巴 IOT 套件的 OTA 机制。在 mqtt 中完成 OTA 包的下载和更新。mqtt 代码中有一个 ota 包的参考实现：lora_ota.tar.gz。

基础版 OTA 逻辑较为简单，下载完成 lora_ota.tar.gz 后，解压执行脚本完成文件系统更新，然后 reboot 网关完成更新。

基础版不限制 OTA 的修改内容，但是 OTA 基础版没有提供可靠加密签名机制（可能产生 OTA 包混淆、冒充），也没有提供一些异常处理、容错机制及回滚操作，使用基础版的合作伙伴，对于固件的管理（重点关注版本依赖问题）和线上网关的 OTA 操作一定要慎重！

4.2.6 OTA 增强功能介绍和注意事项

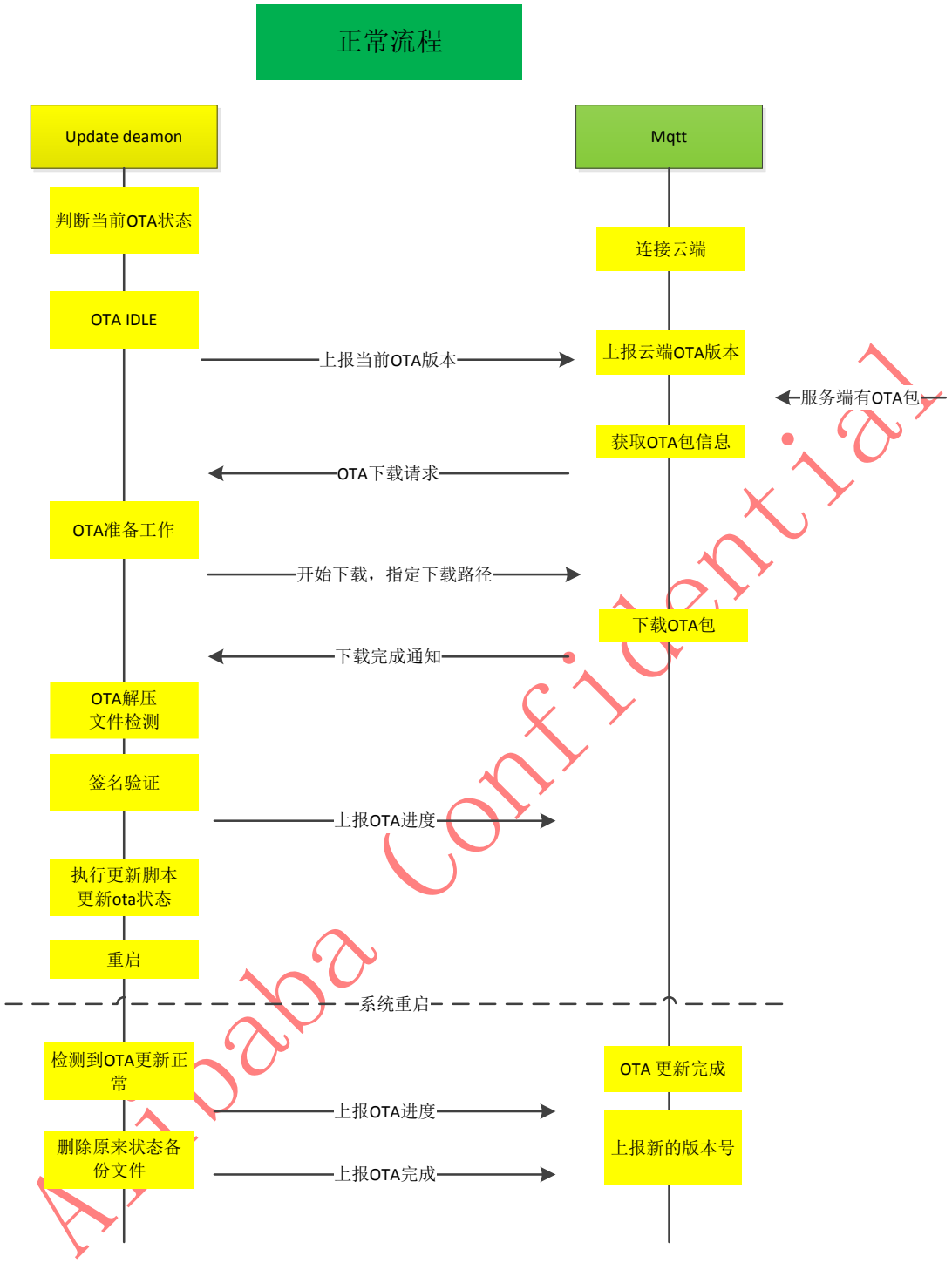
新的 OTA 机制保留了原来的 OTA 机制的优点：

支持进行差分升级，不限制对文件系统修改，支持用户添加脚本操作。同时在下面几点做了提升：

1. 健壮的 OTA 策略——OTA 过程断电恢复、OTA 过程异常后回滚。
2. OTA 自检——升级后网关能够功能自检，自检不通过时，系统回滚到更新前版本。
3. 完善的固件签名机制——一厂商一秘钥（非对称加密），防止固件被篡改或混淆。
4. 添加厂商硬件、软件版本检测和解决 OTA 软件版本依赖问题等。

4.2.6.1 OTA 流程

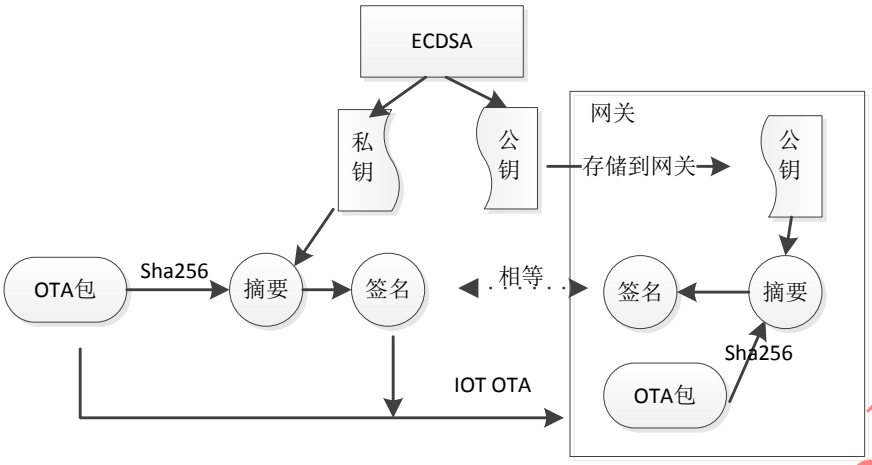
正常的 OTA 流程如下：



OTA 的流程的控制、签名验证、文件系统更新、异常回滚、更新后自检等功能均在 update-deamon 中实现。mqtt 负责服务端固件状态的检测，以及固件的下载功能。

4.2.6.2 Packages 包格式约定

OTA 加入了签名验证机制，签名机制 ECDSA + sha256 算法。签名的原理如下：



通过 ECDSA 算法对 OTA 包的 sha256 签名进行加密生成 OTA 包的签名。OTA 时，OTA 包和签名一起打包，通过 IoT 套件后台的 OTA 功能更新到网关。网关使用公钥进行签名验证。

阿里巴巴为每一个合作伙伴（网关设备商）提供了一套参考的 OTA packages 打包环境和签名工具，合作伙伴可以使用签名工具生成自己的私钥和公钥。建议每个合作伙伴使用一对共钥私钥即可。合作伙伴务必妥善保管私钥。

packages 文件打包：

我们将以一个 packages 文件进行举例说明：

```
$ls packages/
echo_test_rollback.sh    echo_test.sh    lora_pkt_fwd    monitor    mqtt
test_file002  test_file003  update.json
```

其中 update.json 为 OTA packages 的描述信息。如果 packages 内没有这个文件，或者该文件格式不正确，或者内容描述信息不对，都会导致 OTA 失败。

其他文件——为 OTA 包的更新内容。我们将结合 update.json 详细说明。

update.json 内容如下：

```
$cat update.json
{
    "ota-info": {
        "manufacturer": "xxx.xxx.xxx",
        "hw_version": " xxx.xxx",
        "sw_version": "xxx",
        "current_version" : "2.2.1",
        "depend_version": "2.2.0"
    },
    "file-list": [
        { "name": "mqtt", "type": "file", "path": "/home/root/", "operation":
"modify" },
        { "name": "monitor", "type": "file", "path": "/home/root/",
"operation": "modify" },
        { "name": "lora_pkt_fwd", "type": "file", "path": "/home/root//",
"operation": "modify" },
        { "name": "test_file001", "type": "file", "path": "/home/",
"operation": "delete" },
```

```
            {  "name":  "test_file002",  "type":  "file",  "path":  "/home/",
"operation": "new" },
            {  "name":  "test_file003",  "type":  "file",  "path":  "/home/",
"operation": "modify" },
            {   "name":   "echo_test.sh",   "type":   "exec",   "rollback":
"echo_test_rollback.sh" }
        ]
    }
```

ota-info ： 包括厂商信息（manufacturer），厂商网关硬件版本号（hw_version），网关软件版本号（sw_version），当前的 OTA 软件版本号（current_version），依赖的 OTA 软件版本号（depend_version）。

备注：网关软件版本号，由网关厂商出厂设置并维护后续更新。OTA 版本号是在 OTA 时，特有的版本号，同样由网关厂商维护，但是建议采用下面命名规则：在 SDK 版本号后进行扩展命名，例如，SDK 版本号为 2.3.0，厂商使用该版本的 SDK，OTA 版本号命名为 2.3.x 或者 2.3.x.x。

ota-info 内的信息必须和网关信息完全匹配，否则会升级失败，这点需要网关厂商注意。

为了防止版本依赖问题出现，厂商应该慎重管理依赖版本号 depend_version，每次 OTA 时须进行先验，然后填写正确的 depend_version。OTA 依赖版本号，可以是多个，请使用‘;’隔开，例如：

"depend_version": "2.2.0;2.2.1"

"depend_version": "2.2.0;2.2.1;2.2.2"。

depend_version 所填写的版本，网关厂商必须要都经过先验后才能填写，这一点请务必保证。

file-list：OTA 更新包的文件列表信息。

文件列表信息包括：

name	type	path	operation	rollback
文件名称	文件类型，包括：file ——文件 exec ——单次执行的脚本	文件所在的路径	文件操作： new ——添加文件 delete ——删除文件 modify ——修改原有文件	exec 类型的节点特有的回滚操作脚本。

文件列表包括了两种类型文件：file —— 文件；exec ——单次执行的脚本。

file——文件节点表示对一个文件进行删除、添加、修改等操作，这类节点需要告知文件的路径，file 节点可以指定某个文件，也可以指定文件目录。

exec——单次执行的脚本文件：允许用户在不仅可以修改文件，还可以执行特殊脚本操作。这样大大提高了 OTA 的灵活性，可以覆盖到一些文件更新无法解决的升级情况。在 exec 类型的更新下，用户需要提供一个执行脚本的逆操作（rollback），以防止更新异常后，系统无法回滚到更新前状态。

4.2.6.3 OTA 过程的异常处理

OTA 包解析失败和更新后网关自检失败都会导致整个 OTA 流程失败。

一、 OTA 包解析失败

下面情况下，OTA 包将解析失败，将导致升级失败。

1. sign 签名不对
2. 更新包压缩名称不对（必须为 lora_ota.tar.gz）
3. lora_ota.tar.gz 解压后，没有 update.json
4. update.json 格式不对：json 格式不对，没有 ota_info 节点，没有 file_list 节点等。
5. update.json 内 ota_info 和网关不匹配：设备商、硬件版本、软件版本、依赖 OTA 版本必须完全符合；
6. file_list 节点不符合规范。
7. file_list 与文件内容不对：例如添加/修改文件的操作，但是 OTA 包内却没有该文件；单次执行脚本，缺失对应的回滚脚本等。

注意：不强求添加、修改文件的操作和路径一定要正确。当修改一个文件系统不存在的文件时，将自动纠正为新建一个文件；当添加一个文件，但是该文件已存在时，自动纠正为修改该文件。

OTA 包解析成功后，系统进行文件系统更新，更新完成后，网关会进行一次重启，启动后进入自检模式。

二、 更新后网关自检失败

下面情况将认为自检不通过，OTA 失败，并进行文件系统回滚（回滚到更新前状态）。

1. 在 300s 内，没有检测到 pktfwd, mqtt, monitor（如果使能了 monitor）进程没有运行。
2. 在 300s 内，mqtt 没有正常连接云端（有上行包且有下行包）。
3. 在 180s 内，monitor 检测（检测 3 次）到内存或 cpu 占用率超过 80%。
4. 在 300s 内，pktfwd 和服务端没有心跳包连接。

自检失败后，将进行回滚操作：包括恢复原来的文件状态，执行 exec 脚本的回滚脚本。自检通过后，将删除原来保存的备份文件，更新 OTA 版本号到本地，并上报新的版本号到服务端，OTA 更新成功。

更新过程异常及恢复

更新过程异常包括：更新过程中网关断电，或者系统突然重启，网络突然中断等情况。在异常断电、重启时，能够识别上次状态，恢复 OTA 状态上下文，继续完成 OTA 操作。在网络中断情况下，可能会造成下载失败，或者自检失败，这时可能会回滚，并上报 OTA 失败信息到服务端。服务端需再次进行 OTA。

4.2.6.4 现有 OTA 机制升级

目前接入的网关仍然保留了原来的 OTA 机制，在首次更新本版本的 OTA 机制时，还是采用原来的 OTA 机制，将新增加的模块和文件以及启动脚本更新到网关，严格测试后（更新后原有的 OTA 机制将不可用），进行全面更新。

4.2.6.5 OTA 注意事项

1. 每个厂商提供一对共钥/私钥，请不要混用，SDK 默认提供了 semtech 版本的公钥，合作伙伴需要联系阿里巴巴获取自己的公钥。
2. OTA 文件目录及剩余空间确认。OTA 文件下载目录为 update-deamon 的执行目录。OTA 文件解压和验证默认保存的目录为 “/usr/tmp/lora_ota/”（2.3.0 SDK 默认为/var/tmp/lora_ota/,因为在有的网关下，该目录为临时目录，现进行更换，/var/tmp/lora_ota/不是临时目录的，可以忽略此次更改），请保证这些目录有足够的空间，以完成 OTA 包的下载和解压。在 OTA 过程中，会对更新的文件进行备份操作，请确保有足够的空间完成备份。
   合作伙伴可以对 OTA 文件的保存目录进行修改，修改方法：a. 代码上直接修改宏定义 OTA_STORE_DIR；b. 修改 update-deamon 的 Makefile 文件，使能 ENABLE_CUSTOM_OTA_PATH，并且修改 CUSTOM_OTA_PATH 为自定义目录。
3. 更新文件必须严格按照格式。
4. 必须保证更新文件的文件属性是正确的。例如：更新可执行程序，请一定保证更新的文件有执行权限。
5. 无特殊情况，不建议在 OTA 包内使用脚本操作。如果需要使用，请保证更新的脚本操作及逆操作脚本是正确的，且经过仔细验证的。
6. OTA 版本号，建议按照在 SDK 版本号后扩展的规则维护，每次 OTA 包制作都需要指定依赖的 OTA 版本号，依赖版本号必须是经过验证的。
7. 使用 lora_sign 生成的签名，需要将公钥放在网关上（和 update-deamon 一个目录）。lora_sign 生成的公钥为 PublicKey.pem，拷贝到网关时请注意修改首字母大小写（publicKey.pem）。

4.2.6.6 OTA 更新网关设备商自测

合作伙伴需要将 update-deamon 和开启 ENABLE_ADVANCED_OTA 编译选项的其他进程（monitor，mqtt，pktfwd）集成到系统内。使用打包环境生成 OTA 包，然后登陆到 iot 套件后台中，先自验 OTA 包的正确性（可以保留网关 GWEUI，将三元组替换为 IOT 套件后台申请的三元组，然后登陆 IOT 套件后台进行 OTA）。

SDK 内提供了一个参考的打包环境实现，目录为：tools/ota_ref_env/ 。合作伙伴可以将目录下的公钥私钥替换为自己的公钥私钥（公钥私钥很重要，请不要使用 SDK 内默认 key）。秘钥及签名工具为 lora_sign，用户可以使用 lora_sign_src 下的源码编译本地的 lora_sign 工具。秘钥工具及密钥对就绪后，修改打包环境目录下的 packages 内容，添加自己要 OTA 的文件即可。最后调用 gen_ota.sh 生成 OTA 文件。

合作伙伴从 IOT 套件进行 OTA 验证的时候，可能会出现 OTA 自检错误。因为三元组不是 LinkWan 支持的三元组，所以 OTA 过程的自检环节可能会无法通过，导致 OTA 失败，如果是自检环节失败，合作伙伴可以认为 OTA 已经顺利完成。

4.2.7 安全存储功能介绍

安全存储提供了一套 key-value 加密存储机制，目前还在迭代开发中。提供了两套安全

存储服务 sst 和 keychain。

sst 实现带路径的加密存储操作，只要知道路径及 key name 和存储路径就可以对加密数据进行添加删除等操作。

keychain 基于 sst 实现，屏蔽了 sst 中路径操作，只需要关注 key - value 存储。具体 API 参考 sst.h 和 keychain.h。

运行注意事项

注意运行顺序 deploy_sst -> irot_service -> keychain_service。

deloly_sst 不是后台程序，负责运行前的初始化操作，运行时最好添加参数'-f'：./deploy_sst -f 。

irot_service 和 keychain_service 是后台运行的服务进程,依赖 dbus 库。

目前 mqtt 使用了安全存储（只是 NS 下沉的场景才需要），使用 ENABLE_ADVANCED_SECURITY（make.settings）控制 mqtt 是否开启该功能。

keychain 中的全局数据操作 api 功能尚不完善，需要使用时，请先与阿里巴巴联系。

4.3. SDK 使用 Q&A

1. 合作伙伴的 toolchain 应该提供哪些能力？

toolchain 应该默认绑定 sysroot（如果没有绑定 sysroot，请在 makefile 中指定 sysroot 路径，export TOOLCHAIN_SYSROOT=/xxx/xxx/xxx/xxx/sysroot/，指定后编译不过时，请联系阿里巴巴），toolchain 应该支持 libssp（如果不支持，请合作伙伴集成 libssp 到 toolchain，或者提供不支持 libssp 的 libnopoll 库，否则将影响远程调试模块的编译）。

2. 合作伙伴务必不使用版本过低的 toolchain

我们对 gcc 版本没有强制要求，但是版本最好不要太低。版本过低的 toolchain 在编译时容易造成兼容问题，导致编译不过，影响集成进度。

3. 网关原有第三方库和 SDK 使用版本冲突，如何解决？

建议使用 SDK 版本的第三方库，替换原来的版本。如果无法替换原来的第三方库，可以将第三库的版本在 SDK 中进行替换。替换方法：删除 external 目录下 SDK 的第三方库，修改 SDK build.sh，适配编译选项。

我们以替换 dbus 版本为例。删除 SDK external 目录下的 dbus-1.10.18.tar.gz，将新版本的 dbus 软件包拷贝到 external。

修改 build.sh dbus 编译部分：

```
function build_dbus()
{
    #start build dbus
    cd ${BUILDROOT}/external
    --if [ ! -d "./dbus-1.10.18" ]; then
    ++if [ ! -d "./dbus-1.8.20" ]; then

        tar -zxf dbus-1*
    fi
    --cd dbus-1.10.18/
    ++d dbus-1.8.20/
```

…

4. dbus 版本不兼容问题

有一些网关系统内已经集成 dbus，从兼容性的角度考虑，我们建议更新到 SDK 的 dbus 版本。如果更新 dbus 版本存在困难（例如一些文件系统使用了 systemd，强依赖特定版本的 dbus），合作伙伴需要如下操作：

- 确认系统 dbus 软件版本，下载一样版本的 dbus 软件包；
- 集成到 SDK 编译环境内（参见 Q&A 3），然后完成 SDK 重新编译。

SDK 中 dbus 默认的运行配置为 mbusd.conf，总线的监听地址为："unix:path=/tmp/var/run/mbusd/mbusd_socket"。如果系统内已经运行了 dbus-daemon（监听其他的地址），合作伙伴有两种选择：再运行一个 dbus-daemon，配置文件为 mbusd.conf；或者使用系统原来的 dbus-daemon 进行通信，这时要修改 make.settings 的 dbus_address（修改为系统 dbus-daemon 配置的监听地址，配置文件一般在/etc/目录下，需要合作伙伴自行确认），然后重新编译整个 SDK。

watchdog 功能使用了请求 dbus-daemon PID 的功能，有些版本过低的 dbus 可能没有该服务，实测遇到问题时，请参考 4.2.1.2 watchdog 集成注意事项。

5. 日志文件需要多大，过多如何清理

每个进程日志目录存在大小限制， pkfwd、mqtt 是限制 40M，其它进程限制是 5M。日志会同时输出到标准输出和文件，合作伙伴可以自行修改各个模块日志大小，或者日志等级（减少过多日志打印或者日志占用大小，搜索 log_init 函数进行修改）。

如果不需要将日志输出到标准输出，可以修改 log_print 函数，删除第 100 行：

```
--if (lvl >= g_log_lvl) {
--      fprintf(stderr, "%s", buf);
--}
```

6. make.seting 那些模块应该使能？

需要根据合作伙伴认证网关的级别来确认，例如认证的是运营级别网关，需要强制打开使能必选的相关模块。对于可选模块功能，可以根据是否需要使用而选择打开。

7. SDK 编译错误

我们适配了市面上 LoRa 网关常见的 CPU 平台和不同版本的 gcc，但是还是无法完全保证新接入的 gcc 能够顺利完成编译。我们在适配的过程中，见到许多编译错误问题，需要帮助时，可以联系阿里巴巴。如果合作伙伴遇到编译问题，我们鼓励先自行解决，同时将编译问题反馈给我们，一起完善 SDK。

8. 加密 key 及必要修改的信息

三元组加密可以参考 gwiotapi.c 中的实现。请不要使用 SDK 默认的加密 key 值。合作伙伴可以在 gwiotapi.c 替换成自己的加密 key 值。SDK 提供了一套参考的实现，加密过的三元组请更新到 auth_key.json，GWEUI 请更新到 local_conf.json 和 global_conf.json 中。厂商硬软件相关的版本信息请更新到 dev_info.json。

同样适配 OTA 时，也务必不要使用 SDK 提供的共钥、私钥，一定要重新生成一对。

9. OTA 版本号和软件版本号关系

OTA 版本号是为了方便 OTA 管理，专门定义的版本号。命名方式建议在 SDK 版本号后面进行扩展。软件版本号和硬件版本号是合作伙伴自己管理的软硬件信息，和 OTA 版本号无关。注意：OTA 时会检查 OTA 包内的厂商信息、硬件版本号是否和网关保存的一致。