嵌入式软件架构概要、详细设计文档

Revision:	2
Status:	Draft
Author:	Huang Xin
Function:	Netease, Inc.
Path and Filename:	
File Type:	Microsoft Word 2017

Revision History

Rev	Status*	Date	Author	Reason for change/description
1	D	2017/08/07	Huang Xin	First draft
2	D	2017/08/15	Huang Xin	1. 修改 DbusWithLibuvInit 参数说明，在第 25 页 2. 修改 DbusWithLibuvCfgInit 参数说明，在第 26 页 3. 修改 EvpAddCryptoTask 参数说明，在第 38 页 4. 修改 OnDBusHeartLost 定义，在第 44 页 5. 修改 ERR_UNK_TASK_TYPE 为 ERR_UNSUP_EVP_TYPE，在第 20 页

* Statuses include: D: Draft, P: Pending review/frozen, U: Update needed, A: Approved

Approved by: **Date:**

1 概述

本文档详细描述智能音箱产品嵌入式软件架构设计。系统架构采用异步和多进程方案实现。文档将详细描述异步架构方案，多进程消息总线以及通用功能模块，例如数据库、加解密、消息摘要算法实现、网络访问等功能。

系统异步架构采用 libuv 开源库，多进程通信利用 dbus 总线实现。数据库采用 SQLite，加解密和消息摘要算法使用 OpenSSL，网络访问使用 libcurl。本软件框架的主要设计在于实现各个功能模块在异步架构库 libuv 中的集成。

2 术语

术语	说明

3 概要设计

3.1 背景信息

智能音箱产品软件基于嵌入式 linux 系统。为了能更迅速地响应 UI 交互以及并发多任务，软件系统采用异步架构。

为了实现模块化开发以及业务程序的稳定性，软件系统将各个相对独立的任务作为一个独立进程。多个进程间进行数据交互和控制采用 dbus 消息总线实现。

每一个进程通常会使用到一些通用扩展功能，比如数据库，加解密，消息摘要算法，校验和算法，网络通信等。软件框架将提供一些统一的 API 接口来方便使用。

3.2 设计目标

3.2.1 高效性

如何让软件运行的更快。

3.2.2 稳定性

如何让系统运行更稳定：减少软件出现异常的可能性，增加系统长时间运行的稳定性。

3.2.3 可靠性

如何让软件执行结果可靠：确保软件的执行结果符合输入数据的预期。不出现无法预料或者多种不同的执行结果。

3.3 技术实现方案

3.3.1 异步框架 libuv

Libuv 是一个专注于异步 IO 的跨平台高性能事件驱动程序库，具有以下主要功能：

- 由 epoll，kqueue，IOCP，事件端口支持的全功能事件循环。
- 异步 TCP 和 UDP 套接字
- 异步 DNS 解析
- 异步文件和文件系统操作
- 文件系统事件通知
- ANSI 转义码控制 TTY
- IPC 与套接字共享，使用 Unix 域套接字或命名管道
- 子进程
- 线程池
- 信号处理
- 高分辨率时钟

- 线程和同步原语

3.3.2 进程间数据总线 DBus

D-Bus 是一个进程间通信及远程过程调用机制，可以让多个不同的计算机程序（即进程）在同一台电脑上同时进行通信。

D-Bus 是一个低延迟、低开销的通信协议。它设计小而高效，使得最小化传输往返时间。协议基于二进制数据，使得可以节省序列化的时间。并且由于只面向本地处理器使用，没有额外进行字节序转换，字节序在每个消息中声明，使得即使将 D-Bus 总线消息通过网络发送到其它主机，依然能被正确识别。

D-Bus 具有多种软件语言封装，使其能在不同的软件设计语言开发下的程序能够统一地运行。

3.3.3 加密算法库 OpenSSL

OpenSSL 提供需要的 AES、Base64、MD5 等算法支持。并且 OpenSSL 还为网络访问库 libcurl 提供安全支持。

3.3.4 数据库 SQLite3

SQLite3 是一个简单，功能强大的嵌入式关系型数据库，也是目前应用最广泛最稳定的嵌入式客户端关系型数据库。它作为一个进程内的库，实现了自给自足的、无服务器的、零配置的、事务性的 SQL 数据库引擎。它具有以下优点：

- 不需要一个单独的服务器进程或操作的系统（无服务器的）。
- SQLite 不需要配置，这意味着不需要安装或管理。
- 一个完整的 SQLite 数据库是存储在一个单一的跨平台的磁盘文件。
- SQLite 是非常小的，是轻量级的，完全配置时小于 400KiB，省略可选功能配置时小于 250KiB。
- SQLite 是自给自足的，这意味着不需要任何外部的依赖。
- SQLite 事务是完全兼容 ACID 的，允许从多个进程或线程安全访问。
- SQLite 支持 SQL92（SQL2）标准的大多数查询语言的功能。

- SQLite 使用 ANSI-C 编写的，并提供了简单和易于使用的 API。
- SQLite 可在 UNIX（Linux, Mac OS-X, Android, iOS）和 Windows（Win32, WinCE, WinRT）中运行。

3.3.5 JSON 库 cJSON

cJSON 是一个轻量级的 JSON 库，它是一个单文件的的，符合 ANSI-C 变准的 JSON 解析器。再配合 struct2json 库进行封装，可以实现结构体和 JSON 字符串之间的自动序列化和反序列化。

3.3.6 网络访问 libcurl

libcurl 是一个支持 DICT, FILE, FTP, FTPS, Gopher, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMTP, SMTPS, Telnet ,TFTP 多种协议的网络访问库。它的主要功能就是利用不同协议来和各种服务器交换数据。

3.4 系统集成

基于异步事件通知的软件系统架构设计，需要把所有内容整合到符合 libuv 规范的框架中。主要是 DBus 总线和 libcurl 网络访问集成到 libuv 中。并且对各种通用模块调用提供同步以及异步封装接口。

3.5 软件架构组成

系统采用模块化设计，向 APP 提供一个包含了异步消息通知，Dbus 总线消息收发，第三方开源软件库整合的 SDK。该 SDK 并没有对 libuv 异步 I/O 库进行进一步封装，只是将 Dbus 消息收发嵌入到了 libuv 事件循环中。同时对第三方开源软件库提供了基于 libuv 的异步调用封装。使得应用程序在既可以像开发其它软件那样使用 libuv 库，也能自动获得软件架构提供的各种功能支持。

软件架构主要由三部分组成，向 APP 提供的 SDK API；基于 libuv 自动收发、封装、解析 DBus 总线消息以及异步网络通信协议服务；扩展第三方开源库的同步、异步调用实现。

3.5.1 SDK

主要提供 SDK 供应用程序使用。完成系统基本的初始化工作，提供应用程序需要的模块的 API 接口。该 SDK 采用动态库实现，可以供多个进程调用，并且支持多线程功能。

3.5.2 基础服务

基础服务包含 DBus 总线消息支持、异步网络通信支持、配置项服务、进程守护服务功能。

DBus 服务实现 DBus 消息的发送、接收、封装、解析。DBus 服务还定义了进程间通信的协议以及数据结构。

异步网络通信服务实现 http/https Web Services 请求和响应，利用 http/https 下载远程文件。

配置项服务提供一个简单、高效的配置项功能实现，详细内容参见文档《系统配置设计文档》。

进程守护服务利用 DBus 总线上面的心跳包，实现了进程的启动、退出监控功能。

3.5.3 开源库封装

系统中使用到多种第三方开源软件，如 OpenSSL，SQLite，libcurl，cJSON 等。利用这些软件来实现加密、解码，数据库、网络协议访问，JSON 格式化、解析等功能。并且和 libuv 框架高度集成，提供异步调用和同步调用封装接口。

4 详细设计

4.1 模型设计

系统采用分层模块化设计，系统设计模型见图 2.1： 。每个进程通过调用相同的 SDK 来访问系统资源。SDK 负责提供接口和服务。API 接口是应用程序和软件框架提供的服务的桥梁。系统按照功能可以分为以下几部分。

4.1.1 异步事件循环框架以及异步 I/O 访问

该功能由 libuv 开源软件库实现和提供接口，如需使用异步事件循环和异步 I/O 访问，定时器，任务队列，多线程等相关功能可以直接调用 libuv 库的 uv_xxxx 函数，软件框架在这一系列功能上面不做任何封装。应用程序使用到的关于文件访问、定时器、多线程等相关功能应当使用 libuv 而不使用 C 标准库注1，这样能更好地符合软件整体架构。避免某些同步调用阻塞系统异步事件主循环。

4.1.2 D-Bus 总线消息收发

SDK 对 D-Bus 总线 IPC 通信进行了深度封装，在 libdbus 开源软件库的基础上，整合了消息的发送、接收、封装、解析、事件通知等功能，并且提供发送消息的同步和异步 API 接口。

SDK 定义了系统中所有进程间通信消息的格式，并且在消息收发的时候进行了自动封装和自动解析功能。应用程序只需要调用 SDK 提供 API 就能发送消息到指定对象。SDK 支持广播、多播以及点对点的消息收发。

消息接收由回调函数实现，当进程有接收到新消息时，经由回调函数通知应用程序。SDK 内部已经实现消息的并发处理，大大简化了应用程序操作。

根据实际应用需要，D-Bus 总线消息内容为格式化后的 JSON 数据，SDK 中同样提供了 JSON 的自动格式化和解析功能。应用程序只需要处理消息的数据结构，不用关心传输的消息的具体格式和内容。

注1：使用 uv_fs_ 族函数代替 open、read、write，使用 uv_thread_、uv_mutex_、uv_queue_ 族函数代替 pthread_ 相关函数等。

4.1.3 网络访问

SDK 提供一组简单的异步网络访问接口用来实现支持 HTTP/HTTPS 的简单的 Web Services 和文件下载功能。得益于 libuv 异步 I/O 框架，网络访问 API 可以实现多任务高并发操作。

4.1.4 进程守护服务

进程守护服务基于 D-Bus 总线上面的心跳数据包实现，它定时检查在一定时间段内是否收到指定进程的心跳数据。从而判断进程状态。当进程心跳丢失后，SDK 利用回调函数通知应用程序进行处理。

4.1.5 按键事件通知

SDK 利用异步 I/O 读取驱动中的按键状态，如果按键状态发生了改变，SDK 利用回调函数通知应用程序进行处理。

4.1.6 AES 加解密

SDK 提供一对进行 AES 加密、解码的接口，支持指定长度密钥，指定 PAD 的 AES 加解密。SDK 还提供一组对应的异步调用接口。

4.1.7 Base64 编解码

SDK 提供一对对 Base64 编码、解码的同步、异步接口。支持 Base64 自动换行功能。

4.1.8 MD5 消息摘要

SDK 提供支持文件 MD5 消息摘要算法的同步、异步接口。

4.1.9 SQLite3 数据库

SDK 中一些功能依赖于 SQLite3 数据库，但 SDK 本身并不对 SQLite3 数据库进行任何封装。应用程序可以按照常规方法在程序中使用 SQLite 数据库。

4.1.10 配置项管理

详细内容参考文档《系统配置设计文档》。

4.1.11 系统日志

提供一个高性能、高并发的系统日志方案。实现日志信息的格式化，存储，传输，展示功能。

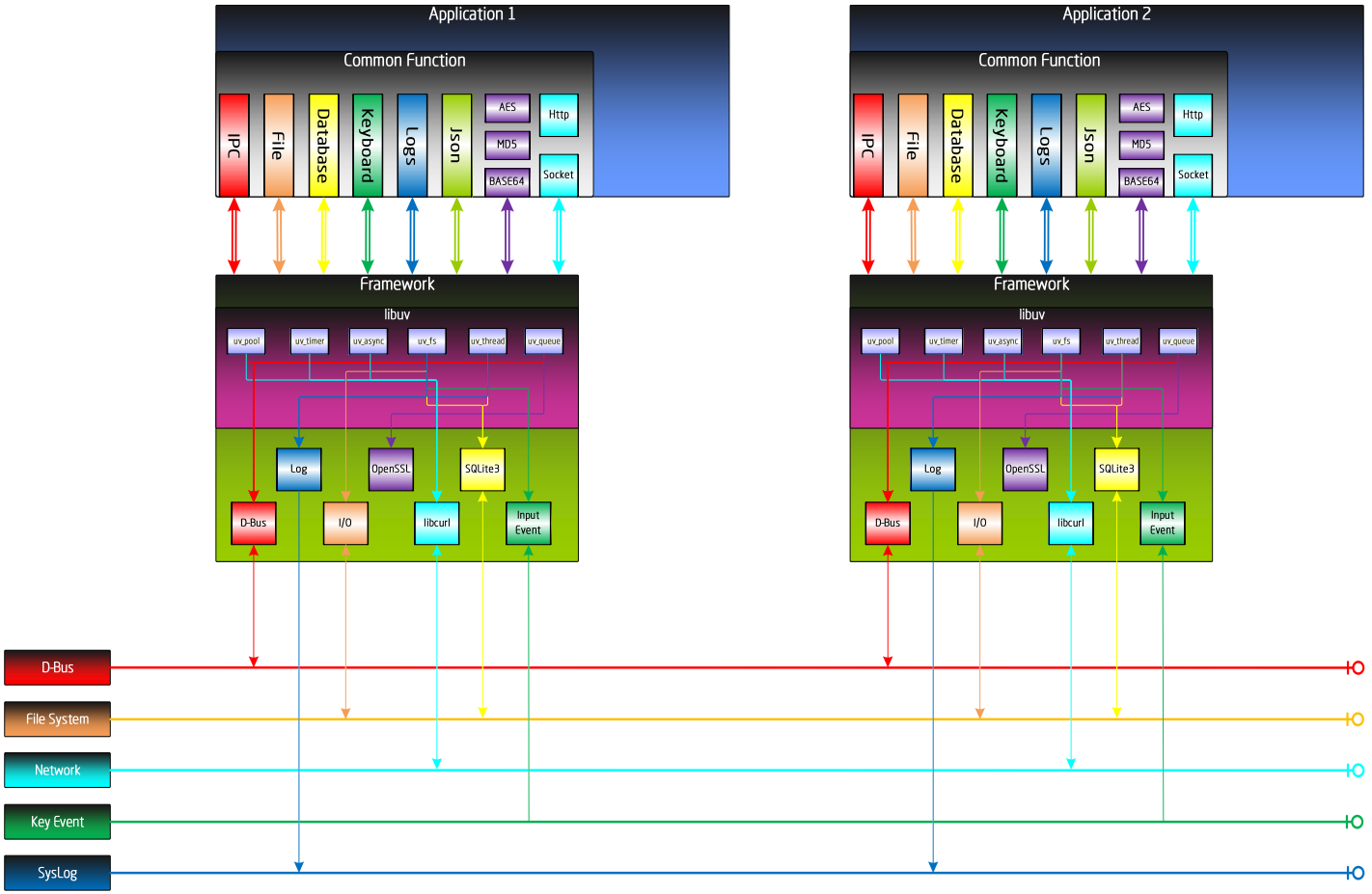
Application 1
Common Function
IPC
File
Database
Keyboard
Logs
Json
AES
MD5
BASE64
Http
Socket
Framework
libuv
uv_pool
uv_timer
uv_async
uv_fs
uv_thread
uv_queue
Log
OpenSSL
SQLite3
D-Bus
I/O
libcurl
Input Event
Application 2
Common Function
IPC
File
Database
Keyboard
Logs
Json
AES
MD5
BASE64
Http
Socket
Framework
libuv
uv_pool
uv_timer
uv_async
uv_fs
uv_thread
uv_queue
Log
OpenSSL
SQLite3
D-Bus
I/O
libcurl
Input Event
D-Bus
File System
Network
Key Event
SysLog

图2.1：　　系统模型图

4.2 开源软件库

4.2.1 异步 I/O 库

异步 I/O 库本质上是为 I/O 提供异步访问和异步事件通知功能。异步 I/O 访问解决 I/O 访问的阻塞问题，异步事件通知提供事件驱动引擎。常用的基于 C 语言的异步 I/O 库有 libevent，libev，Libuv 三种。相比之下，libuv 功能更强大，设计更简洁，移植性更好，使用更方便。目前在 libuv 是更新更快，关注最多，使用最多的异步 I/O 库。Libuv 模型设计详见图 2.2：

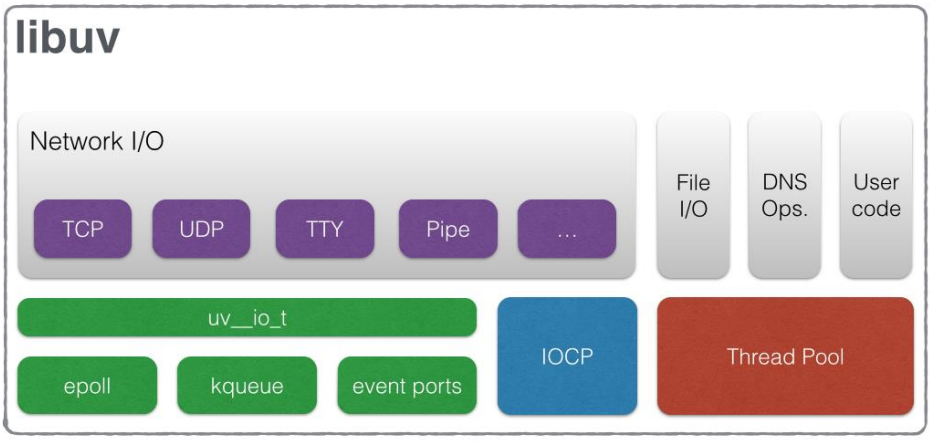

图2.2： Libuv 模块组成图

异步 I/O 循环和事件循环是 libuv 的核心，每一个循环是一个单线程。可以建立多个循环来处理任务，每一个循环运行在一个线程中。每个循环处理的流程图见图 2.3： 。

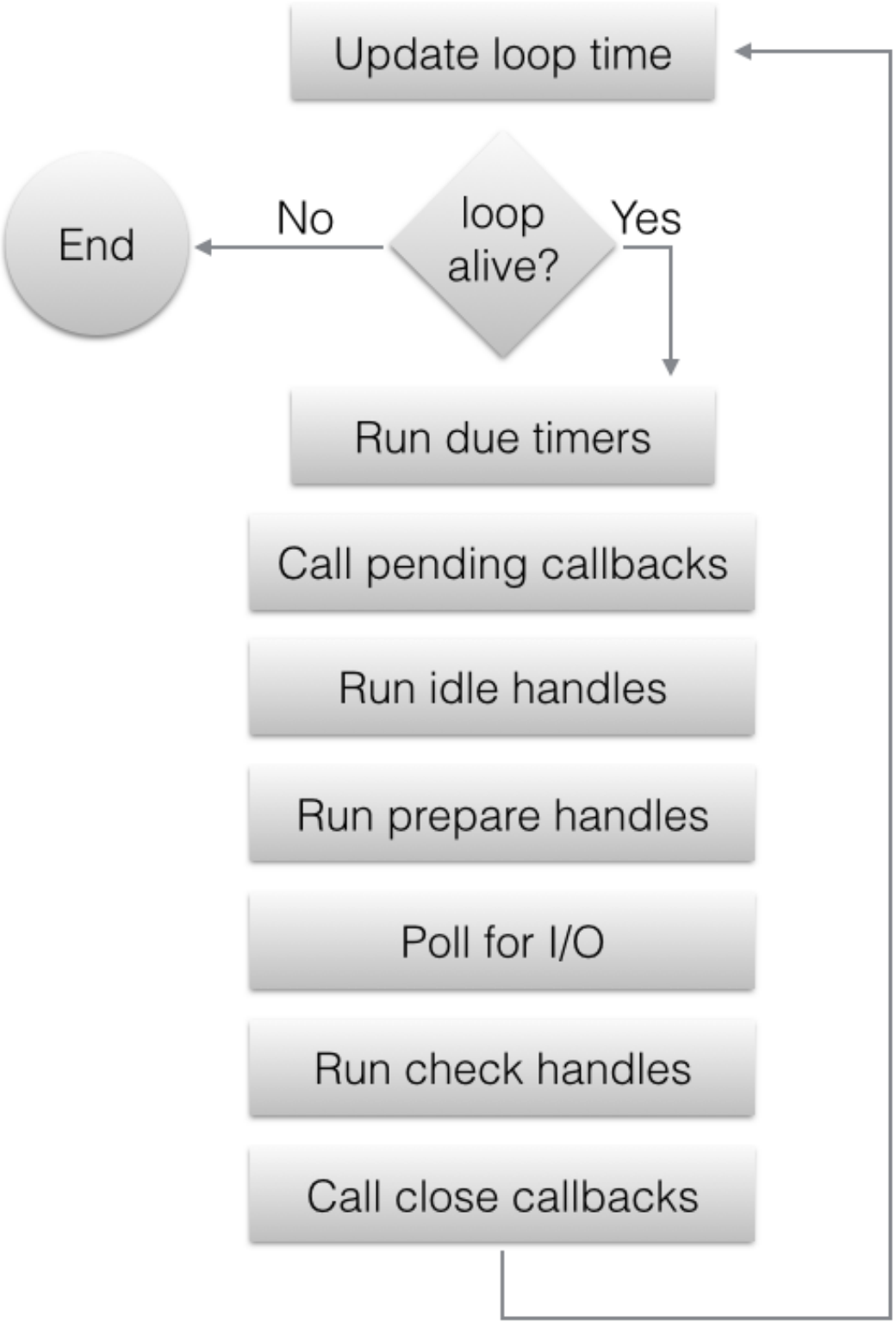

图2.3：　　Libuv 事件循环流程图

4.2.2 IPC 进程通信

进程间通信利用 D-Bus 消息总线实现，可以利用消息总线传输少量数据和命令。

D-Bus 是一个进程间通信及远程过程调用机制，可以让多个不同的计算机程序（即进程）在同一台电脑上同时进行通信。D-Bus 作为 freedesktop.org 项目的一部分，其设计目的是使 Linux 桌面环境（如 GNOME 与 KDE 等）提供的服务标准化。freedesktop.org 项目同时也开发了一个称为 libdbus 的自由及开放源代码软件库，作为规范的参考实现。

D-Bus 抽象为三层：libdbus 让两个应用程序可以互相链接并交换消息的库； dbus-daemon 消息总线的可执行文件，建基于 libdbus，可链接到多个应用程序。这个守护进程可以将消息按特定路径转送给零个或更多个应用程序，从而实现发布/订阅模式；基于特定应用程序框架的封装库。

本系统就采用的是 libdbus。

4.2.3 网络通信

Libcurl 是一个简单易用的 URL 传输库，支持绝大多数网络协议。系统主要利用它对 HTTP/HTTPS 的支持来访问 Web Services 和 HTTP/HTTPS 服务器文件。传统的 socket 进行这些操作太复杂，需要对每种应用协议进行组包和解包操作。Libcurl 可以自动完成协议层处理。并且 libcurl 对异步访问支持完善，和 libuv 兼容。

4.3 交互逻辑

4.3.1 嵌入式端发起操作交互逻辑图

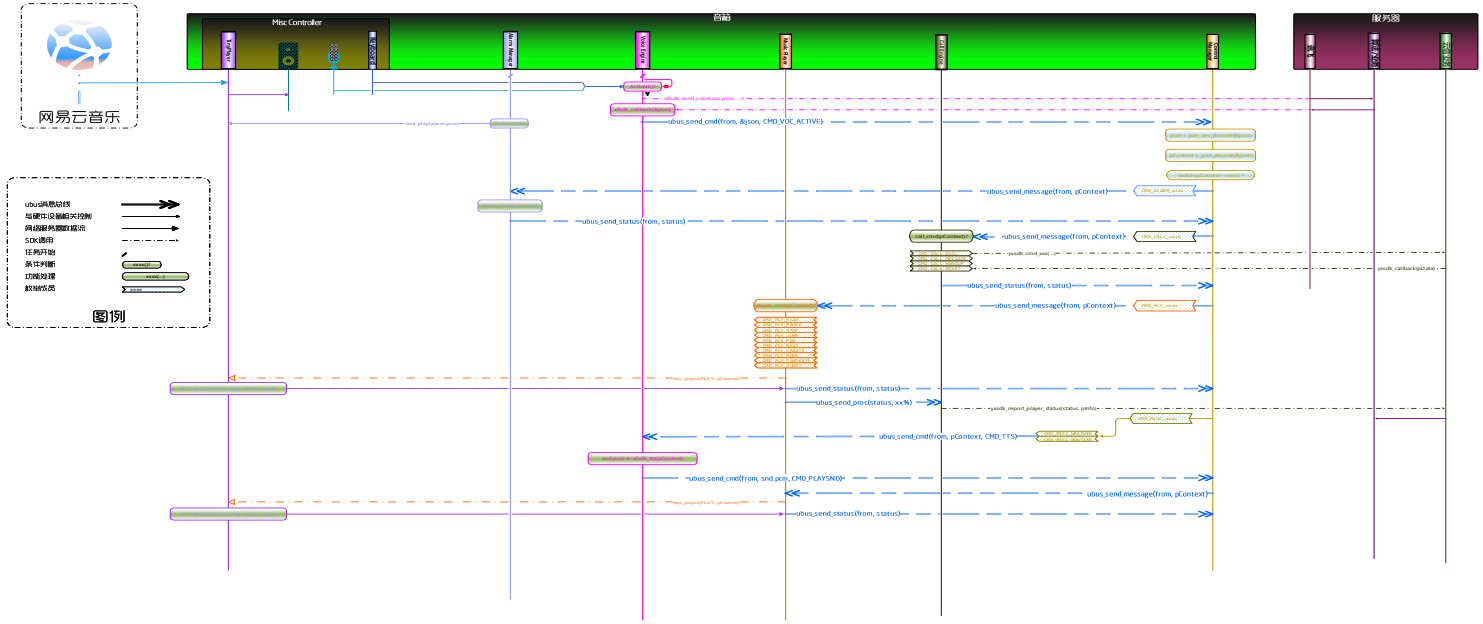

图2.4： 嵌入式端发起操作数据交互逻辑

4.3.2 服务器 Push 消息交互逻辑图

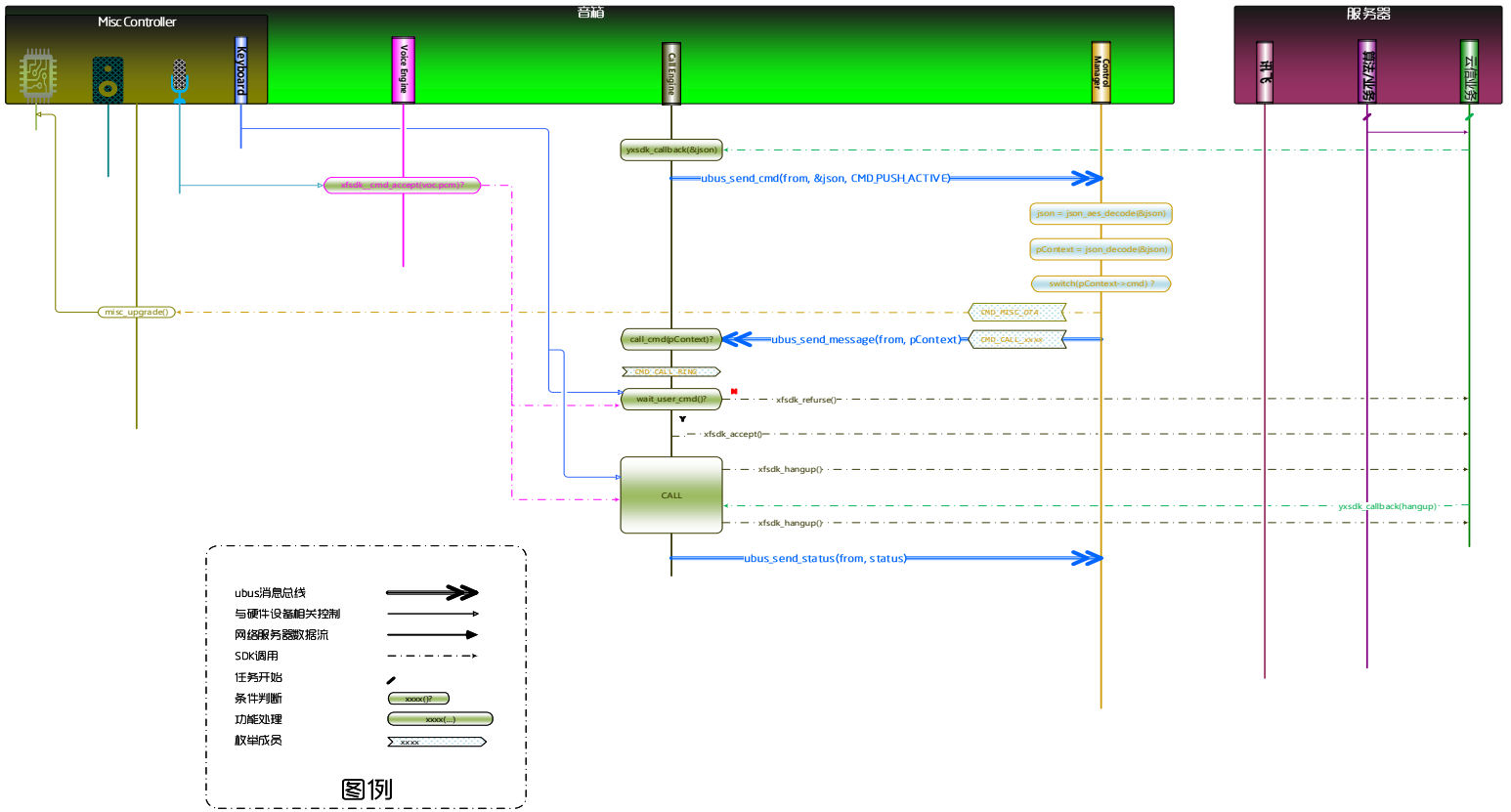

图2.5：　　服务器 Push 数据交互逻辑

4.3.3 嵌入式端与服务端段连接交互逻辑图

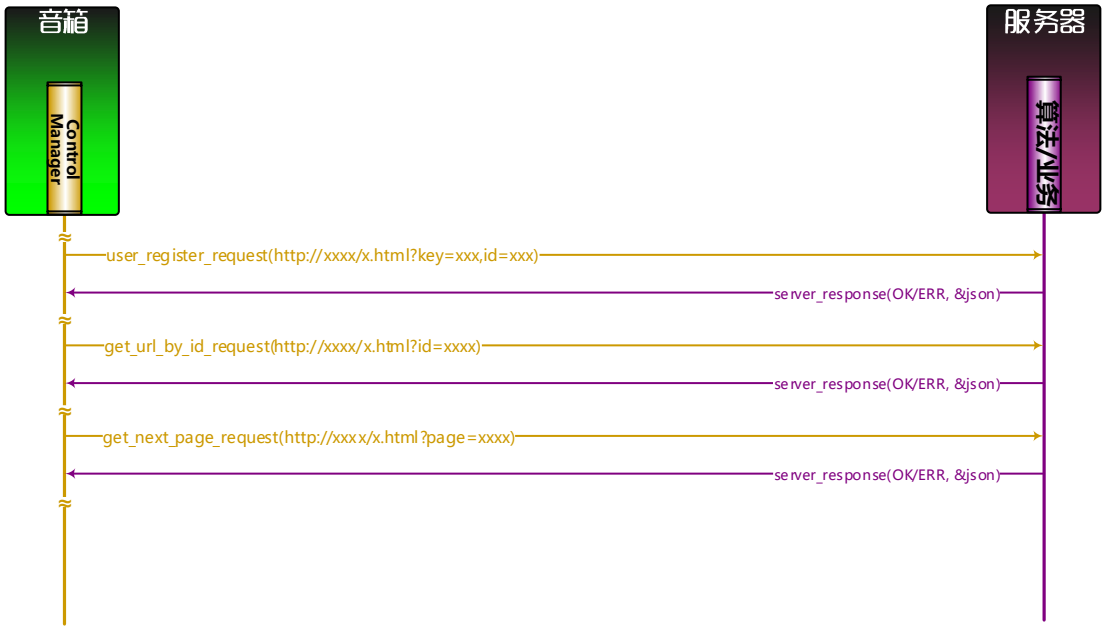

图2.6：　　嵌入式和服务器短链接数据交互逻辑

4.4 SDK API 文档

4.4.1 常量定义

4.4.1.1 错误码

错误码	值
ERR_INPUT_PARAMS	1
ERR_GET_BUS,	2
ERR_DBUS_CONNECTION,	3
ERR_REQUEST_BUS_NAME,	4
ERR_SET_WATCH_FUNCTION,	5
ERR_SET_TIMEOUT_FUNCTION,	6
ERR_BUS_MATCH,	7
ERR_BUS_SET_MSG_CB,	8
ERR_DBUS_CREATE_MSG,	9
ERR_BUS_SEND_MSG,	10
ERR_DBUS_MSG_TO_LARGE,	11
ERR_BUS_RCV_MSG,	12
ERR_ADD_TASK,	13
ERR_UNSUP_EVP_TYPE,	14
ERR_CREATE_MQ,	15
ERR_MQ_SENDMSG,	16
ERR_CREATE_SHM,	17
ERR_MAP_SHM,	18
ERR_EVP_INIT_KEY,	19
ERR_EVP_UPDATE,	20
ERR_EVP_FINALE,	21
ERR_EVP_KEY_SIZE,	22

4.4.1.2 JSON 引擎定义

```
typedef enum
{
//*****************************************
// Player <--> Controller
//*****************************************
   JSON_ENGINE_P2C     = 0,
   JSON_ENGINE_C2P,
//*****************************************
// Configure Req & Rsp
//*****************************************
   JSON_ENGINE_CFG_REQ,
   JSON_ENGINE_CFG_RSP,
//*****************************************
// Alarm Req & Rsp
//*****************************************
   JSON_ENGINE_ALARM_REQ,
   JSON_ENGINE_ALARM_RSP,
//*****************************************
// PING Cmd
//*****************************************
   JSON_ENGINE_PING,
   JSON_ENGINE_MAX,
} JSON_ENGINE_TYPE;
```

4.4.1.3 模块（进程）名称

```
typedef enum
{
   MODULE_CONTROLLER                = 0,
   MODULE_ALARM,
   MODULE_CALL,
   MODULE_VOICEENGINE,
   MODULE_PLAYER,
   MODULE_CONFIGURE,
MODULE_MAX,
} MODULE_NAME;
```

4.4.1.4 配置项

```
typedef struct
{
    char*       pKeyName;
    int         keyType;
    int         keyModule;
    union
    {
        char*           pStrValue;
        int             intValue;
        double          doubleValue;
    };
    UT_hash_handle  hh;         ///< UT Hash handle
} CFG_ITEM, *PCFG_ITEM;
```

4.4.1.5 异步 EVP 操作类型

```
typedef enum
```

```
    {
        CRYPTO_AES_ENCRYPT          = 0,
        CRYPTO_AES_DECRYPT,
        CRYPTO_BASE64_ENCODE,
        CRYPTO_BASE64_DECODE,
    CRYPTO_MD5_FILE,
    } CRYPTO_TYPE;
```

4.4.1.6 日志等级

```
    typedef enum _LOG_LEVEL_
    {
        LOG_Fatal   = (1 << 0),
        LOG_Error   = (1 << 1),
        LOG_Warn    = (1 << 2),
        LOG_Debug   = (1 << 3),
        LOG_Info    = (1 << 4),
        LOG_Test    = (1 << 5),
        LOG_Call    = (1 << 6),
        LOG_Devp    = (1 << 7),
        LOG_Step    = (1 << 8),
        LOG_Unknown = (1 << 9),
        LOG_All     = (0xFFFFFFFF),
        LOG_Close   = 0x0,
    } LOG_LEVEL;
```

4.4.2 结构体定义

4.4.2.1 软件框架对象结构体

```
    typedef struct
    {
        uv_loop_t*          pLoop;              ///< libuv default main loop
        DBusConnection*     pBus;               ///< D-Bus object
        MODULE_NAME         modName;            ///< Process name
        const char*         pBusName;           ///< D-Bus object's interface
name
        const char*         pBusPath;           ///< D-Bus object's path name
        OnDBusMessage       onMsgCb;            ///< D-Bus receive message
callback
        OnKeyEvent          onKeyCb;            ///< Keyboard event callback
        OnDBusHeartLost     onHblCb;            ///< Process HBL event
callback
        OnCfgMsg            onCfgCb;            ///< Configure message
callback
    } LIBUV_DBUS_PARAMS, *PLIBUV_DBUS_PARAMS;
```

4.4.2.2 消息传输协议结构体

```
    typedef struct
    {
        int32_t         msgSrc;             ///< message who send
        uint32_t        msgDests;           ///< who need receive message(not only one)
    #if DBUS_MSG_TIMESTAMP
        uint32_t        tmSec;                  ///< timestamp of second
        uint32_t        tmUsec;                 ///< timestamp of usecond
    #endif
```

```
    uint32_t        busCmd;                 ///< command of message
    uint32_t        msgSize;                ///< how many bytes of this message
    uint32_t        msgKey;                 ///< share key for copy large message
    char*           pMsg;                  ///< message context if has
} DBUS_MSG_PACK, *PDBUS_MSG_PACK;
```

4.4.3 API 接口

4.4.3.1 DbusWithLibuvInit

- **功能描述：**

系统初始化函数，触发系统进行相关配置。初始化成功后，系统所有相关功能才能正常工作。

- **头文件：**

libuv_dbus.h

- **函数原型：**

```
DBusConnection* DBusWithLibuvInit(
            uv_loop_t*            pLoop,
             const char*          pBusName,
             OnDBusMessage        cbOnMsg,
             OnDBusHeartLost      cbOnHbl,
             OnKeyEvent           cbOnKey,
             int*                 pErrno);
```

- **参数说明：**

参数	说明
pLoop	libuv loop 对象指针
pBusName	D-Bus 对象 Interface 名
cbOnMsg	D-Bus 消息接收回调函数，参考 4.4.4.1
cbOnHbl	进程守护服务消息回调函数，参考 4.4.4.2
cbOnKey	按键事件回调函数，参考 4.4.4.3
pErrno	错误码，参考 4.4.1.1

- **返回值：**

成功	DBusConnection
失败	NULL，错误码由参数 pErrno 指示

4.4.3.2 DBusWithLibuvCfgInit

- **功能描述：**

  配置项消息回调函数初始化，指定配置项消息的回调函数。

- **头文件：**

  libuv_dbus.h

- **函数原型：**

```
int DBusWithLibuvCfgInit(OnCfgMsg cbOnCfgMsg);
```

- **参数说明：**

参数	说明
cbOnCfgMsg	配置项消息处理回调函数，参考 4.4.4.4

- **返回值：**

成功	0
失败	<0，具体错误信息参考 4.4.1.1

4.4.3.3 DBusSendToCommand

- **功能描述：**

  利用 D-Bus 发送消息到其它进程。

- **头文件：**

  libuv_dbus.h

- **函数原型：**

```
int DBusSendToCommand(
          DBusConnection    *pBus,
           const char       *pBusName,
           uint32_t             busCmd,
           const char       *pContext);
```

- **参数说明：**

参数	说明
pBus	D-Bus 对象指针，NULL 表示使用系统默认对象
pBusName	D-Bus 对象 Interface 名
busCmd	D-Bus 消息接收回调函数
pContext	字符串消息内容

- **返回值：**

成功	0
失败	<0，具体错误信息参考 4.4.1.1

4.4.3.4 DBusBoardcastCommand

- **功能描述：**

  利用 D-Bus 广播消息。

- **头文件：**

  libuv_dbus.h

- **函数原型：**

```
int DBusBoardcastCommand(
          DBusConnection   *pBus,
          uint32_t            msgToMask,
          uint32_t            busCmd,
          const char       *pContext);
```

- **参数说明：**

参数	说明
pBus	D-Bus 对象指针，NULL 表示使用系统默认对象
msgToMask	D-Bus 消息接收对象掩码，每一位表示一个进程，1：接收 0：不接收
busCmd	D-Bus 消息接收回调函数
pContext	字符串消息内容

- **返回值：**

成功	0
失败	<0，具体错误信息参考 4.4.1.1

4.4.3.5 DBusJsonSendToCommand

- **功能描述：**

  利用 D-Bus 发送 JSON 格式 API 消息到其它进程。

- **头文件：**

  libuv_dbus.h

- **函数原型：**

```
int DBusJsonSendToCommand(
                    DBusConnection        *pBus,
                    const char*            pBusName,
                    uint32_t               busCmd,
                    JSON_ENGINE_TYPE       type,
                    void*                  pStruct,
                    int                    enBase64);
```

- **参数说明：**

参数	说明
pLoop	libuv loop 对象指针
pBusName	D-Bus 对象 Interface 名
busCmd	D-Bus 消息接收回调函数
type	JSON 自动转换引擎 ID，见 4.4.1.2
pStruct	发送消息的数据结构体指针
enBase64	是否使用字符串字段的 Base64 编码： 1：使用 0：不使用

- **返回值：**

成功	0
失败	<0，具体错误信息参考 4.4.1.1

4.4.3.6 DBusBoardcastCommand

- **功能描述：**

  利用 D-Bus 广播 JSON API 消息。

- **头文件：**

  libuv_dbus.h

- **函数原型：**

```
int DBusBoardcastCommand(
        DBusConnection      *pBus,
        uint32_t            msgToMask,
        uint32_t            busCmd,
        JSON_ENGINE_TYPE    type,
        void*               pStruct,
        int                 enBase64);
```

- 参数说明：

参数	说明
pLoop	libuv loop 对象指针
msgToMask	D-Bus 消息接收对象掩码，每一位表示一个进程，1：接收 0：不接收
busCmd	D-Bus 消息接收回调函数
type	JSON 自动转换引擎 ID，见 4.4.1.2
pStruct	发送消息的数据结构体指针
enBase64	是否使用字符串字段的 Base64 编码： 1：使用 0：不使用

- **返回值：**

成功	0
失败	<0，具体错误信息参考 4.4.1.1

4.4.3.7 DBusJsonSendToCommandAsync

- **功能描述：**

  利用 D-Bus 发送 JSON 格式 API 消息到其它进程。

- **头文件：**

  libuv_dbus.h

- **函数原型：**

```
int DBusJsonSendToCommandAsync (
          DBusConnection*        pBus,
          const char*            pBusName,
          uint32_t               busCmd,
          JSON_ENGINE_TYPE       type,
          void*                  pStruct,
          int                    iSize,
          OnDBusAsyncSendTo      cbSendTo,
          int                    enBase64);
```

- **参数说明：**

参数	说明
pLoop	libuv loop 对象指针
pBusName	D-Bus 对象 Interface 名
busCmd	D-Bus 消息接收回调函数
type	JSON 自动转换引擎 ID，见 4.4.1.2
pStruct	发送消息的数据结构体指针
iSize	pStruct 结构体大小
cbSendTo	消息发送结果回调函数
enBase64	是否使用字符串字段的 Base64 编码： 1：使用 0：不使用

- **返回值：**

成功	0
失败	<0，具体错误信息参考 4.4.1.1

4.4.3.8 EvpAESEncrypto

- **功能描述：**

  对数据进行 AES 加密。

- **头文件：**

  crypto.h

- **函数原型：**

```
int EvpAESEncrypto(
            unsigned char*   pInBuf,
            int              iSize,
            unsigned char*   pOutBuf,
            int*             pOutSize,
            unsigned char*   pKey);
```

- **参数说明：**

参数	说明
pInBuf	待加密数据指针
iSize	待加密数据大小（字节数）
pOutBuf	加密后数据指针
pOutSize	加密后数据大小（字节数）
pKey	密钥

- **返回值：**

成功	0
失败	<0，具体错误信息参考 4.4.1.1

4.4.3.9 EvpAESDecrypto

- **功能描述：**

  对 AES 加密的数据进行解密。

- **头文件：**

  crypto.h

- **函数原型：**

```
int EvpAESDecrypto (
            unsigned char*   pInBuf,
            int              iSize,
            unsigned char*   pOutBuf,
            int*             pOutSize,
            unsigned char*   pKey);
```

- 参数说明：

参数	说明
pInBuf	待解密数据指针
iSize	待解密数据大小（字节数）
pOutBuf	解密后数据指针
pOutSize	解密后数据大小（字节数）
pKey	密钥

- **返回值：**

成功	0
失败	<0，具体错误信息参考 4.4.1.1

4.4.3.10 EvpBase64Encode

- **功能描述：**

对字符串进行 Base64 编码，函数返回编码后的 Base64 格式字符串。使用后需要显示地释放其占用的内存。

- **头文件：**

crypto.h

- **函数原型：**

```
const char* EvpBase64Encode(const char* pSrc);
```

- **参数说明：**

参数	说明
pSrc	需要进行 Base64 编码的字符串

- **返回值：**

成功	Base64 编码后的字符串。
失败	NULL

4.4.3.11 EvpBase64Decode

- **功能描述：**

对 Base64 编码字符串进行解密，函数返回 Base64 解码后的格式字符串。使用后需要显示地释放其占用的内存。

- **头文件：**

crypto.h

- **函数原型：**

```
const char* EvpBase64Decode(const char* pBase64);
```

- **参数说明：**

参数	说明
pBase64	需要进行 Base64 解码的字符串

- **返回值：**

成功	Base64 解码后的字符串。
失败	NULL

4.4.3.12 EvpMD5HashFile

- **功能描述：**

  计算文件的 MD5 校验和。

- **头文件：**

  crypto.h

- 函数原型：

```
const char* EvpMD5HashFile (const char* pFileName);
```

- **参数说明：**

参数	说明
pFileName	需要计算 MD5 值的文件名

- **返回值：**

成功	MD5 消息摘要字符串。
失败	NULL

4.4.3.13 EvpAddCryptoTask

- **功能描述：**

  系统密码、消息摘要、Base64 等操作的异步调用接口。

- **头文件：**

  crypto.h

- **函数原型：**

```
int EvpAddCryptoTask(CRYPTO_TYPE type,
        unsigned char* pInBuf,
        int         iSize,
        unsigned char* pOutBuf,
        char*           pKey,
        OnEVPCrypto  onEvpCryptCb);
```

- **参数说明：**

参数	说明
type	异步操作类型，详见 4.4.1.5
pInBuf	数据指针
iSize	加密大小（字节数）
pOutBuf	操作完成后的数据指针
pKey	加，解密密钥
onEvpCryptCb	异步操作完成后的回调函数，参考 4.4.4.5

- 返回值：

成功	0
失败	<0，具体错误信息参考 4.4.1.1

4.4.3.14 CfgGetKeyValue

- **功能描述：**

  获取配置项的值。

- **头文件：**

  config_engine.h

- **函数原型：**

```
int CfgGetKeyValue(const char* pKeyName,
            PCFG_ITEM* pItem);
```

- **参数说明：**

参数	说明
pKeyName	配置项名称
pItem	配置项值：具体内容参考 4.4.1.4

- **返回值：**

成功	0
失败	<0，具体错误信息参考 4.4.1.1

4.4.3.15 CfgChangeKeyValue

- **功能描述：**

  改变配置项的值。

- **头文件：**

  config_engine.h

- **函数原型：**

```
int CfgChangeKeyValue(const char *pKeyName,
                      PCFG_ITEM pItem,
                      int saveToDB);
```

- **参数说明：**

参数	说明
pKeyName	配置项名称
pItem	配置项值：具体内容参考 4.4.1.4
saveToDB	是否保持到数据库中（掉电保存）： 1：保存到磁盘中 0：不保存到磁盘中，重启后丢失

- **返回值：**

成功	0
失败	<0，具体错误信息参考 4.4.1.1

4.4.3.16 CfgAddKeyValue

- **功能描述：**

  增加一个新的配置项。

- **头文件：**

  config_engine.h

- **函数原型：**

```
int CfgAddKeyValue (const char *pKeyName,
                    PCFG_ITEM pItem,
                    int saveToDB);
```

- **参数说明：**

参数	说明
pKeyName	配置项名称
pItem	配置项值：具体内容参考 4.4.1.4
saveToDB	是否保持到数据库中（掉电保存）： 1：保存到磁盘中 0：不保存到磁盘中，重启后丢失

- **返回值：**

成功	0
失败	<0，具体错误信息参考 4.4.1.1

4.4.3.17 LOG_EX

- **功能描述：**

  LOG 日志接口。

- **头文件：**

  log.h

- **函数原型：**

```
#define LOG_EX(level, format, args...);
```

- **参数说明：**

参数	说明
level	日志等级，详见 4.4.1.6
format	格式字符串，参考 printf
args...	可变参数值，参考 printf

- **返回值：**

成功	无
失败	无

4.4.4 回调函数

4.4.4.1 OnDBusMessage

- **功能描述：**

  D-Bus 消息接收回调函数

- **头文件：**

  libuv_dbus.h

- **函数原型：**

```
typedef PDBUS_MSG_PACK (*OnDBusMessage)(
     uv_loop_t*      pLoop,
     DBusConnection*     pConn,
     PDBUS_MSG_PACK  pMsg);
```

- **参数说明：**

参数	说明
pLoop	libuv loop 对象指针
pConn	D-Bus 对象指针，NULL 表示使用系统默认对象
pMsg	D-Bus 消息接收回调函数

- **返回值：**

成功	NULL
失败	NULL

4.4.4.2 OnDaemonMsg

- **功能描述：**

  进程守护服务事件回调函数

- **头文件：**

  libuv_dbus.h

- **函数原型：**

```
typedef void (*OnDaemonMsg)(MODULE_NAME modName,
                            int status);
```

- **参数说明：**

参数	说明
modName	模块名称，详见 4.4.1.3
status	进程状态： 0：Connected 1：Disconnected

- **返回值：**

成功	无
失败	无

4.4.4.3 OnKeyEvent

- **功能描述：**

  按键消息回调函数

- **头文件：**

  libuv_dbus.h

- **函数原型：**

```
typedef void (*OnKeyEvent)(
        uint16_t uType,
        uint16_t uKey,
        int32_t iValue);
```

- **参数说明：**

参数	说明
uType	事件类型： EV_KEY：按键事件
uKey	按键值，指示那个按键产生的事件
iValue	按键动作： 1：按下 0：松开

- **返回值：**

成功	无
失败	无

4.4.4.4 OnCfgMsg

- **功能描述：**

  配置项消息回调函数

- **头文件：**

  config_engine.h

- **函数原型：**

```
typedef void (*OnCfgMsg)(
        DBUS_CMD      cmd,
        PCFG_ITEM     pMsg,
        int           err);
```

- **参数说明：**

参数	说明
cmd	消息类型
pMsg	配置项值：具体内容参考 4.4.1.4
err	错误码：具体内容参考 4.4.1.1

- **返回值：**

成功	无
失败	无

4.4.4.5 OnEVPCrypto

- **功能描述：**

  配置项消息回调函数

- **头文件：**

  crypto.h

- **函数原型：**

```
typedef void (*OnEVPCrypto)(
          CRYPTO_TYPE           type,
          const unsigned char*    pData,
          int                       iSize,
          const unsigned char*    pSrcData,
          int                       iError);
```

- **参数说明：**

参数	说明
type	异步操作类型，详见 4.4.1.5
pData	结果数据指针
iSize	结果数据大小（字节数）
pSrcData	源数据指针
iError	错误码，参考 4.4.1.1

- **返回值：**

成功	无
失败	无

5 Review Information

Topic of Review: **Date**:

Related Documents:

Meeting Moderator:

Meeting Scribe:

Invitees	Attended (Y/N)

Approval required from anyone who didn't attend?

If so, from whom?

Agenda/Issues discussed:

1.
2.
3.
4.

Outcome:

Action items

Item	Owner	Date Due	Status	Closed?