EC2x&AG35-QuecOpen

JSON-XML 应用指导

LTE 系列

版本：EC2x&AG35-QuecOpen_JSON-XML_应用指导_V1.0

日期：2017-12-21

状态：临时文件

上海移远通信技术股份有限公司始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市徐汇区虹梅路 1801 号宏业大厦 7 楼　　邮编：200233
电话：+86 21 51086236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：
http://www.quectel.com/cn/support/sales.htm

如需技术支持或反馈我司技术文档中的问题，可随时登陆如下网址：
http://www.quectel.com/cn/support/technical.htm
或发送邮件至：support@quectel.com

前言

上海移远通信技术股份有限公司提供该文档内容用以支持其客户的产品设计。客户须按照文档中提供的规范、参数来设计其产品。由于客户操作不当而造成的人身伤害或财产损失，本公司不承担任何责任。在未声明前，上海移远通信技术股份有限公司有权对该文档进行更新。

版权申明

文档历史

修订记录

版本	日期	作者	变更表述
1.0	2017-12-21	高飞虎	初始版本

目录

1 JSON

1.1. JSON-C 介绍

JSON-C 实现了一个引用计数对象模型，它允许在 C 中构造 JSON 对象，将它们输出为 JSON 格式的字符串，并将 JSON 格式的字符串解析为 JSON 对象的 C 表示，遵循 RFC 7159。

Json-c 提供了一组 json 格式的处理 API，Quectel 模组集成了开源 json-c,用户在编写应用程序时可直接使用；需要包含相关头文件，并链接 libjson-c 库，参考 1.3

1.2. JSON-C API 介绍

枚举：

```
typedef enum json_type {
json_type_null,
json_type_boolean,
json_type_double,
json_type_int,
json_type_object,
json_type_array,
json_type_string,
} json_type;
```

1.2.1. 创建 Json 对象

创建一个 Json 对象：

```
struct json_object * json_object_new_object (void)
```

创建一个 Json 数组对象：

```
struct json_object * json_object_new_array (void)
```

1.2.2. 将 json 格式的 string 转化为 json 对象

```
struct json_object * json_tokener_parse(const char *str)
```

1.2.3. 将 json 对象转化为 json 格式的 string

const char * json_object_to_json_string (struct json_object *obj)

1.2.4. 向 json_object 内的增加，查询，删除

添加：
void json_object_object_add (struct json_object *obj, const char *key, struct json_object *val)

查询：
json_bool json_object_object_get_ex(struct json_object* obj, const char *key, struct json_object **value);

删除：
void json_object_object_del (struct json_object *obj, const char *key)

1.2.5. 将其他基础数据类型转化为 json 基础类型

struct json_object * json_object_new_int (int i)
struct json_object * json_object_new_double (double d)
struct json_object * json_object_new_string (const char *s)
struct json_object * json_object_new_boolean (boolean b)
struct json_object * json_object_new_string_len (const char *s, int len)

1.2.6. 基础数据类型对象转化

将 json 对象转化为另外一个 json 对象
struct json_object * json_object_get (struct json_object *obj)

将 json_object 转化为 lh_table
struct lh_table * json_object_get_object (struct json_object *obj)

将 json_object 转化为 array_list
struct array_list * json_object_get_array (struct json_object *obj)

将 json_object 转化为 boolean
boolean json_object_get_boolean (struct json_object *obj)

将 json_object 转化为 int
int json_object_get_int (struct json_object *obj)

将 json_object 转化为 double
double json_object_get_double (struct json_object *obj)

将 json_object 转化为 char *
const char * json_object_get_string (struct json_object *obj)

1.2.7. 销毁一个 json 对象

void json_object_put (struct json_object *obj)

1.2.8. 判断 json_object 的类型是否为基础数据类型 type

返回值：是：0；否：1
int json_object_is_type (struct json_object *obj, enum json_type type)

1.2.9. 获得 json_object 的基础类型

enum json_type json_object_get_type (struct json_object *obj)

1.2.10. json 数组

创建 json 数组：
struct json_object * json_object_new_array (void)

将 json 数组转化为 arraylist：
struct array_list * json_object_get_array (struct json_object *obj)

获得 json 数组的长度：
int json_object_array_length (struct json_object *obj)

向 json 数组 obj 内添加一个 json_object：val，其中 obj 为 json 数组，val 为需要添加的对象
int json_object_array_add (struct json_object *obj, struct json_object *val)

向 obj 中的 idx 位置，update 或 insert 对象 val
int json_object_array_put_idx (struct json_object *obj, int idx, struct json_object *val)

获得 json 数组中的某一特定的对象：
struct json_object * json_object_array_get_idx (struct json_object *obj, int idx)

1.2.11. 参考资料

https://github.com/json-c/json-c/wiki

1.3. Quectel json 示例

参考：example/json
编译 example_json.c，上传 example_json 到模块，./example_json

详细例子请参考：https://gist.github.com/jasoner/6e39c168650206ce18dcf954846cd971

2 XML

2.1. LIBXML2 介绍

Libxml2 是为 Gnome 项目开发的 XML C 解析器和工具包，它是在 MIT 许可下可用的免费软件。XML 本身是一种用于设计标记语言的元语言，也就是文本语言，在这种语言中，语义和结构被附加在尖括号内的额外“标记”信息添加到内容中。尽管这个库是用 C 语言编写的，但是很多语言绑定都可以在其他环境中使用。众所周知，Libxml2 是非常可移植的，库可以在各种系统上构建和工作。Libxml2 提供了一组 xml 格式的处理 API, Quectel 模组集成了开源 libxml2,用户在编写应用程序时可直接使用；需要包含相关头文件，并链接 libxml2 库，参考 2.3

2.2. LIBXML2 API 介绍

2.2.1. 去除空白字符

int xmlKeepBlanksDefault(int val)

函数功能：在分析 XML 数据时，去除空白字符。如果不去除空白字符，则这些字符也会被当做一个 node 来处理

参数说明: val：0 或者 1。0 表示去除空白字符，1 表示不去除;

返回值：0 表示设置失败，1 表示设置成功;

2.2.2. XML 文件载入和保存函数

xmlDocPtr xmlParseFile(const char * filename)

函数功能：将 XML 文件从硬盘上载入到内存中，并且生成 DOM 树。使用完毕之后，需要用 xmlFreeDoc()来释放资源

参数说明: filename：XML 文件名称;

返回值：如果载入成功，则返回这个文档的根节点。否则返回 NULL;

int xmlSaveFormatFileEnc(const char * filename, xmlDocPtr cur, const char * encoding, int format)

函数功能：将内存中的 DOM 树，保存到硬盘上，生成一个带格式的 XML 文件

参数说明: filename：需要保存的文件的名称

cur：需要保存的 XML 文档

encoding：导出文件的编码类型，或者为 NULL

format：是否格式化。0 表示不格式化，1 表示需要格式化。注意：只有当 xmlIndentTreeOutput 设置为 1，或者 xmlKeepBlanksDefault(0)时，format 设置为 1 才能生效

返回值：写入文件中的字节数量

2.2.3. XML 内存载入和输出函数

xmlDocPtr xmlParseMemory(const char * buffer, int size)

函数功能：将一块内存中的 XML 数据生成一个 DOM 树。使用完毕之后，需要用 xmlFreeDoc()来释放资源

参数说明: buffer：存放 XML 格式数据的内存区

size：内存中 XML 格式数据的长度

返回值：如果载入成功，则返回这个文档的根节点；否则返回 NULL

void xmlDocDumpFormatMemoryEnc(xmlDocPtr out_doc, xmlChar ** doc_txt_ptr,
int * doc_txt_len, const char * txt_encoding, int format)

函数功能：将 DOM 树导出到内存中，形成一个 XML 格式的数据

参数说明: out_doc：需要输出成为一个 buffer 的 XML 文档

doc_txt_ptr：输出文档的内存区。由该函数在内部申请。使用完成之后，必须调用 xmlFree()函数来释放该内存块

doc_txt_len：输出文档内存区的长度

txt_encoding：输出文档的编码类型

format：是否格式化。0 表示不格式化，1 表示需要格式化。注意只有当 xmlIndentTreeOutput 设置为 1，或者 xmlKeepBlanksDefault(0)时，format 设置为 1 才能生效

2.2.4. 创建和释放 XML 文档函数

xmlDocPtr xmlNewDoc (const xmlChar * version)

函数功能：在内存中创建一个新的 XML 文档。所创建的文档需要使用 xmlFreeDoc()来释放资源

参数说明：version：XML 标准的版本，目前只能指定为“1.0”

void xmlFreeDoc(xmlDocPtr cur)

函数功能：释放内存中的 XML 文档

参数说明：cur：需要释放的 XML 文档

2.2.5. XML 节点操作函数

xmlNodePtr xmlDocGetRootElement(xmlDocPtr doc)

函数功能：获得根节点

参数说明：doc：XML 文档句柄。

返回值：XML 文档的根节点，或者 NULL。

xmlNodePtr xmlDocSetRootElement(xmlDocPtr doc, xmlNodePtr root)

函数功能：设置根节点
参数说明：doc：XML 文档句柄
root：XML 文档的新的根节点
返回值：如果该文档原来有根节点，则返回根节点，否则返回 NULL;
xmlChar * xmlNodeGetContent (xmlNodePtr cur)

函数功能：获得节点的内容
参数说明：cur：节点的指针
返回值：节点的文本内容。如果该节点没有文本内容，则返回 NULL。当返回值不为 NULL 时，需要用 xmlFree()函数来释放返回的资源

void xmlNodeSetContentLen(xmlNodePtr cur, const xmlChar * content, int len)

函数功能：设置节点的内容长度
参数说明：cur：节点的指针
content：节点的新文本内容
len：节点新文本内容的长度

void xmlNodeAddContentLen(xmlNodePtr cur, const xmlChar * content, int len)
函数功能：在节点的内容后面添加新的内容
参数说明：cur：节点的指针
content：节点的新加的文本内容
len：节点新加的文本内容的长度

xmlChar * xmlGetProp(xmlNodePtr node, const xmlChar * name)
函数功能：获得节点的属性
参数说明：node：XML 节点的指针
name：该节点的属性的名称
返回值：该属性的值或者为 NULL。如果不为 NULL，则需要用 xmlFree()来释放资源

xmlAttrPtr xmlSetProp(xmlNodePtr node, const xmlChar * name, const xmlChar * value)

函数功能：设置节点的属性（如果该属性已经存在，则替换其值）
参数说明：node：需要设置属性的节点
name：属性的名称
value：属性的值
返回值：该属性节点的指针

2.2.6. 参考资料

http://www.xmlsoft.org/

2.3. Quectel xml 示例

示例：example/xml
编译 example_xml.c，上传 example_xml 和 test.xml 到模块，./example_xml test.xml 获取指定 value
更详细示例浏览：http://www.xmlsoft.org/examples/