EC2x&AG35-QuecOpen JSON-XML Application Guide

LTE Standard/Automotive Module Series

Rev. EC2x&AG35-QuecOpen _Application_Guide_V1.0

Date: 2017-12-21

Status: Preliminary

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
7th Floor, Hongye Building, No.1801 Hongmei Road, Xuhui District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:
http://www.quectel.com/support/sales.htm

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm
Or email to: support@quectel.com

GENERAL NOTES

QUECTEL OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

COPYRIGHT

About the Document

History

Revision	Date	Author	Description
1.0	2017-12-21	Gale GAO	Initial

Contents

About the Document **2**

Contents **3**

1 JSON **4**
1.1. JSON-C Introduction 4
1.2. JSON-C API Introduction 4
1.2.1. Creation of Json Object 4
1.2.2. Parsing Json-format String into Json Object 5
1.2.3. Parsing Json Object into Json-Format String 5
1.2.4. Adding, Querying and Deleting Operations of Json Object 5
1.2.5. Parsing Other Basic Data Types into Json Basic Type 5
1.2.6. Basic Data Type Object Conversion 5
1.2.7. Destroying A Json Object 6
1.2.8. Making A Judgement for Whether the Type of Json_Object Is Basic Data Type 6
1.2.9. Getting Json Object Basic Type 6
1.2.10. Json Array 6
1.2.11. Reference Information 6
1.3. Quectel Json Demonstrate Example 6

2 XML **8**
2.1. LIBXML2 Introduction 8
2.2. LIBXML2 API Introduction 8
2.2.1. Removing Blank Characters 8
2.2.2. XML File Loading and Function Saving 8
2.2.3. XML Memory Loading and Function Output 9
2.2.4. Creation and Releasing for XML Document Function 9
2.2.5. XML Node Operation Function 9
2.2.6. Reference 10
2.3. Quectel XML Demonstrate Example 11

1 JSON

1.1. JSON-C Introduction

JSON-C implements a reference counting object model that allows you to easily construct JSON objects in C, output them as JSON formatted strings and parse JSON formatted strings back into the Crepresentation of JSON objects. It aims to conform to RFC 7159.

JSON-C provides a set of processing API that is json-formatted. The Quectel module integrates with open source JSON-C, which can be used directly by users when developing application programs. Related head file needs to be included and also it needs to link libjson-c library. For more details, please refers to chapter 1.3

1.2. JSON-C API Introduction

Enumeration:
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. Creation of Json Object

Creating a Json Object:
struct json_object * json_object_new_object (void)

Creating a Json Array Object:
struct json_object * json_object_new_array (void)

1.2.2. Parsing Json-format String into Json Object

struct json_object * json_tokener_parse(const char *str)

1.2.3. Parsing Json Object into Json-Format String

const char * json_object_to_json_string (struct json_object *obj)

1.2.4. Adding, Querying and Deleting Operations of Json Object

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

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

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

1.2.5. Parsing Other Basic Data Types into Json Basic Type

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. Basic Data Type Object Conversion

Converting a json object to another one
struct json_object * json_object_get (struct json_object *obj)

Converting json_object to lh_table
struct lh_table * json_object_get_object (struct json_object *obj)

Converting json_object to arraylist
struct array_list * json_object_get_array (struct json_object *obj)

Converting json_object to boolean
boolean json_object_get_boolean (struct json_object *obj)

Converting json_object to int
int json_object_get_int (struct json_object *obj)

Converting json_object to double

double json_object_get_double (struct json_object *obj)

Converting json_object to char *
const char * json_object_get_string (struct json_object *obj)

1.2.7. Destroying A Json Object

void json_object_put (struct json_object *obj)

1.2.8. Making A Judgement for Whether the Type of Json_Object Is Basic Data Type

Return Value:Yes: 0; No: 1
int json_object_is_type (struct json_object *obj, enum json_type type)

1.2.9. Getting Json Object Basic Type

enum json_type json_object_get_type (struct json_object *obj)

1.2.10. Json Array

Creating a json array:
struct json_object * json_object_new_array (void)

Converting json array to arraylist:
struct array_list * json_object_get_array (struct json_object *obj)

Getting json array length:
int json_object_array_length (struct json_object *obj)

Adding a json_object:val to json array obj, in which obj means json array, val means obj that needs to be added
int json_object_array_add (struct json_object *obj, struct json_object *val)

Updating or inserting object val to the idx position in obj:
int json_object_array_put_idx (struct json_object *obj, int idx, struct json_object *val)

Getting a specific object from json array:
struct json_object * json_object_array_get_idx (struct json_object *obj, int idx)

1.2.11. Reference Information

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

1.3. Quectel Json Demonstrate Example

Reference: example/json
Compiling example json.c and uploading example json to /example json directory in the module.

For more detailed examples, please refer to:
https://gist.github.com/jasoner/6e39c168650206ce18dcf954846cd971

2 XML

2.1. LIBXML2 Introduction

Libxml2 is an XML C parser and toolkit used for the Gnome project development. It's a free software available under the MIT license agreement. XML itself is a meta-language used for designing markup language, namely, a textual language, in which both semantics and structure are added by the information content within the angle brackets between the extra "tags". Although the library is programmed by C language, many binding languages are still able to be used in other environment. As is known to us, Libxml2 has an excellent portability. Its library can be constructed and operated on various systems. Libxml2 provides a set of processing API which is xml format. Quectel module integrates with open source libxml2, which can be used directly by users when developing application programs. Related head files need to be included and also need to link to libxml2 library. Please refer to chapter 2.3

2.2. LIBXML2 API Introduction

2.2.1. Removing Blank Characters

int xmlKeepBlanksDefault(int val)
Function: analyzing XML data, removing blank characters. If not to do so, these characters will be processed as a node.
Parameter Description: var : 0 or 1. 0 means to remove blank characters, 1 means not to remove;
Return Value: 0 means setting failure, 1 means setting success.

2.2.2. XML File Loading and Function Saving

xmlDocPtr xmlParseFile(const char * filename)
Function: Loading XML file from hard disk to memory and generate a DOM tree. After usage completion, it needs to release its resource by xmlFreeDoc().
Parameter Description: filename: XML filename:
Return Value: If file loading is successful, it will return to root node of this document, or it will return Null;

int xmlSaveFormatFileEnc(const char * filename, xmlDocPtr cur, const char * encoding, int format)
Function: Saving DOM tree in the memory to hard disk and then generate a XML- format file.
Parameter Description: filename: filename that needs to be saved
Cur: XML document that needs to be saved
Encoding: The encoding type of the exported file, or the type is NULL

Format: Whether it needs formatting. 0: Yes, 1: No. Note: Only when xmlIndentTreeOutput is set to 1 or xmlKeepBlanksDefault(0), format is set to 1 will take effect.

Return Value: Bytes were written to the file

2.2.3. XML Memory Loading and Function Output

xmlDocPtr xmlParseMemory(const char * buffer, int size)
Function: Generating XML data from the memory to a DOM tree. After usage completion, it needs to release its resource by xmlFreeDoc().

Parameter Description: buffer: Memory zone used for storing XML format data
Size: XML-format data length in the memory
Return value: if the load is successful, it will return to root node of this document; otherwise return NULL

void xmlDocDumpFormatMemoryEnc(xmlDocPtr out_doc, xmlChar ** doc_txt_ptr,
int * doc_txt_len, const char * txt_encoding, int format)
Function: Exporting DOM tree to memory to form an XML- format data
Parameter Description: Out_doc: the output is an XML document that is buffer type
doc_txt_ptr：Memory zone for outputting document. It will be applied internally by this function. After usage completion, it must release memory block by calling xmlFree().
doc_txt_len: Memory zone length for outputting document
txt_encoding：Encoding type for outputting document
Format: Whether needs to format. 0: Yes, 1: No. Note: Only when xmlIndentTreeOutput is set to 1 or xmlKeepBlanksDefault(0), format is set to 1 will take effect.

2.2.4. Creation and Releasing for XML Document Function

xmlDocPtr xmlNewDoc (const xmlChar * version)
Function: Creating a new XML document in the memory. The document created needs to use xmlFreeDoc() to release resource.
Parameter Description: version: The XML standard version, it can currently only be designated as "1.0"
void xmlFreeDoc(xmlDocPtr cur)
Function:Releasing XML document in the memory
Parameter description: cur: XML document that needs to be released

2.2.5. XML Node Operation Function

xmlNodePtr xmlDocGetRootElement(xmlDocPtr doc)
Function: Getting root nodes
Parameter Description: doc: XML document handle.
Return Value: Return to XML document root node or return NULL.

xmlNodePtr xmlDocSetRootElement(xmlDocPtr doc, xmlNodePtr root)
Function: Setting root node
Parameter Description: doc: Document handle
Root: New root node for XML document
Return value: Return value: If the document originally has a root node, it will return root node, otherwise it will return NULL;

xmlChar * xmlNodeGetContent (xmlNodePtr cur)
Function: Getting node content
Parameter Description: cur: Node pointer
Return value: The text content of the node. If the node has no text content, it will return NULL.When the return value is not NULL, it needs to release the returned resource by xmlFree()

void xmlNodeSetContentLen(xmlNodePtr cur, const xmlChar * content, int len)
Function: setting node content length
Parameter Description: cur: node pointer
Content: New text content for node
Len: New text content length

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

Function: Add new content at the end of the node content
Parameter Description: cur: node pointer
Content: New added text content for node
Len: New added text content length for node

xmlChar * xmlGetProp(xmlNodePtr node, const xmlChar * name)
Function: Getting node property
Parameter Description: node: XML node pointer
Name: Property name for this node
Return Value: The value of this property may be NULL. If it's not NULL, then it needs to release its resource by xmlFree().

xmlAttrPtr xmlSetProp(xmlNodePtr node, const xmlChar * name, const xmlChar * value)
Function: Setting node property (if the property has already existed, then replace its value)
Parameter Description：node：node that needs to set its property
Name: property name
Value: property value
Return value: node pointer of this property

2.2.6. Reference

http://www.xmlsoft.org/

2.3. Quectel XML Demonstrate Example

Demonstrate example: example/xml
Compiling example xml.c, uploading.xml and test.xml to module, /example.xml test.xml get designated value
For more details, Please browse website: http://www.xmlsoft.org/examples/