IoT Datahub C# SDK

by Dasudian

Contents

1 IoT Datahub C# SDK

SDK提供了一个简单的基于MQTT协议与服务器交互的方法
SDK基于MQTT协议,传输实时的消息到大数点IoT云服务器
你可以收集设备上的数据发送到云上;也可以订阅某个topic,来接收云服务器推送的 消息

如何使用SDK:
1: 创建一个客户端实例
2: 如果想接收消息,那么就订阅某个topic
3: 或者发送消息到服务器
4: 退出时,销毁该客户端

2 更新日志

Author	Date	Version	Note
Kevin Liang	10/26/2017	03.00.00	删除了异步发送接口,发送数据的时候需要指定数据的格式（文本,json,二进制）;在Builder()的第3个参数改成客户端的设备类型;大版本升级,不向下兼容
Kevin Liang	6/29/2017	2.2.0	修改API：添加了接收消息的代理回调函数原型和连接状态改变的回调函数原型，即将废弃异步发布消息选项
Jack Liu	4/27/2017	2.1.0	修复bug: 无法发送qos0消息; 暴露选项cleansession; 订阅时可设置qos
Jack Liu	3/27/2017	2.0.0	客户端全面升级，版本更新为2.0.0

3 命名空间索引

3.1 包

这里列出所有的包，附带简要说明(如果有的话):

4 继承关系索引

4.1 类继承关系

此继承关系列表按字典顺序粗略的排序:

5 类索引

5.1 类列表

这里列出了所有类、结构、联合以及接口定义等，并附带简要说明:

6 文件索引

6.1 文件列表

这里列出了所有文件，并附带简要说明:

7 命名空间文档

7.1 com 命名空间参考

命名空间

- namespace dasudian

7.2 com.dasudian 命名空间参考

命名空间

- namespace iot

7.3 com.dasudian.iot 命名空间参考

命名空间

- namespace sdk

7.4 com.dasudian.iot.sdk 命名空间参考

类

- class ConnectionStatusChangedEventArgs
- class Constants
- class DataHubClient
- class Message
- class MessageEventArgs
- class ServiceException

枚举

- enum DataHubClientErrorCode { EXCEPTION_ILLEGAL_PARAMETERS = -1 }

7.4.1 枚举类型说明

7.4.1.1 DataHubClientErrorCode

```
enum DataHubClientErrorCode  [strong]
```

枚举值

EXCEPTION_ILLEGAL_PARAMETERS	非法参数

8 类说明

8.1 DataHubClient.Builder类 参考

使用该Builder获取DataHubClient实例

Public 成员函数

- Builder (string instanceId, string instanceKey, string clientType, string clientId)
  通过该*Builer*获取*DataHubClient*实例
- Builder (string instanceId, string instanceKey, string clientId)
  通过该*Builer*获取*DataHubClient*实例
- Builder SetServerURL (string serverURL)
  设置私有云服务器地址。如果不设置，则默认使用大数点公有云测试服务器。
- Builder SetDebug (bool debug)
  是否打开调试模式
- Builder SetCleanSession (bool cleanSession)
  是否清除会话
- DataHubClient Build ()
  构建*DataHubClient*实例

属性

- string InstanceId [get, set]
- string InstanceKey [get, set]
- string ClientType [get, set]
- string ClientId [get, set]
- string ServerURL [get, set]
- bool Debug [get, set]
- bool CleanSession [get, set]

8.1.1 详细描述

使用该Builder获取DataHubClient实例

8.1.2 构造及析构函数说明

8.1.2.1 Builder() [1/2]

```
Builder (
            string instanceId,
            string instanceKey,
            string clientType,
            string clientId )
```

通过该Builer获取DataHubClient实例

参数

instanceId	用于大数点验证用户，保证客户端与服务器间的安全通信。 demo中的instanceId仅可以用于测试大数点IoT DataHub功能使用， 如果您想正式使用大数点IoT服务，请联系大数点客服获取私有的instanceId
instanceKey	用于大数点验证用户，保证客户端与服务器间的安全通信。 demo中的instanceKey仅可以用于测试大数点IoT DataHub功能使用， 如果您想正式使用大数点IoT服务，请联系大数点客服获取私有的instanceKey
clientType	客户端设备类型，可以填写任意的utf-8字符（不能以"_"开头且不能包含“|”）。 如 sensor,electricArm
clientId	客户端id，用于服务器唯一标记一个客户端，服务器通过该id向客户端推送消息; 注意：不同的客户端的id不能相同，如果有两个相同的客户端id， 服务器会关闭掉其中的一个客户端的连接。 你可以使用设备的mac地址，或者第三方账号系统的id（比如qq号，微信号）。 如果没有自己的账号系统，则可以随机生成一个不会重复的客户端id。

8.1.2.2 Builder() [2/2]

```
Builder (
            string instanceId,
            string instanceKey,
            string clientId )
```

通过该Builer获取DataHubClient实例

参数

instanceId	用于大数点验证用户，保证客户端与服务器间的安全通信。 demo中的instanceId仅可以用于测试大数点IoT DataHub功能使用， 如果您想正式使用大数点IoT服务，请联系大数点客服获取私有的instanceId
instanceKey	用于大数点验证用户，保证客户端与服务器间的安全通信。 demo中的instanceKey仅可以用于测试大数点IoT DataHub功能使用， 如果您想正式使用大数点IoT服务，请联系大数点客服获取私有的instanceKey
clientId	客户端id，用于服务器唯一标记一个客户端，服务器通过该id向客户端推送消息; 注意：不同的客户端的id不能相同，如果有两个相同的客户端id， 服务器会关闭掉其中的一个客户端的连接。 你可以使用设备的mac地址，或者第三方账号系统的id（比如qq号，微信号）。 如果没有自己的账号系统，则可以随机生成一个不会重复的客户端id。

8.1.3 成员函数说明

8.1.3.1 Build()

```
DataHubClient Build ( )
```

构建DataHubClient实例

返回

返回DataHubClient实例

8.1.3.2 SetCleanSession()

```
Builder SetCleanSession (
            bool cleanSession )
```

是否清除会话

参数

cleanSession	false: 当客户端断线或下线后, 保存客户端订阅的topic和发送给客户端的所有消息. true: 当客户端断线或下线后, 不保留客户端订阅的topic和发送给客户端的任何消息. 默认为false

返回

Builder实例

8.1.3.3 SetDebug()

```
Builder SetDebug (
            bool debug )
```

是否打开调试模式

参数

debug	true:打开调试模式；false:关闭调试。默认为false

返回

Builder实例

8.1.3.4 SetServerURL()

```
Builder SetServerURL (
            string serverURL )
```

设置私有云服务器地址。如果不设置，则默认使用大数点公有云测试服务器。

参数

serverURL	服务器的地址，加密连接方式tcp://host:port，非加密连接方式ssl://host:port； 其中ssl和tcp都必须为小写，host可以使用域名或ip地址，port表示对应的端口。

返回

Builder实例

8.1.4 属性说明

8.1.4.1 CleanSession

```
bool CleanSession  [get], [set]
```

8.1.4.2 ClientId

```
string ClientId [get], [set]
```

8.1.4.3 ClientType

```
string ClientType [get], [set]
```

8.1.4.4 Debug

```
bool Debug [get], [set]
```

8.1.4.5 InstanceId

```
string InstanceId [get], [set]
```

8.1.4.6 InstanceKey

```
string InstanceKey [get], [set]
```

8.1.4.7 ServerURL

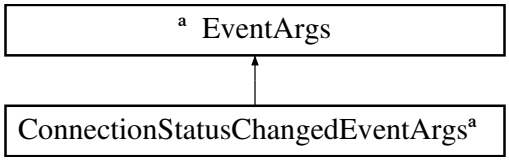
```
string ServerURL [get], [set]
```

该类的文档由以下文件生成:

- DataHubClient.cs

8.2 ConnectionStatusChangedEventArgs 类参考

类 ConnectionStatusChangedEventArgs 继承关系图:



Public 成员函数

- ConnectionStatusChangedEventArgs (bool isConnected)

属性

- bool IsConnected [get, set]
  当前连接状态。*true:*连接正常；*false:*与服务器连接断开。

8.2.1 成员函数说明

8.2.1.1 ConnectionStatusChangedEventArgs()

```
ConnectionStatusChangedEventArgs (
            bool isConnected )
```

8.2.2 属性说明

8.2.2.1 IsConnected

```
bool IsConnected  [get], [set]
```

当前连接状态。true:连接正常；false:与服务器连接断开。

该类的文档由以下文件生成:

- ConnectionStatusChangedEventArgs.cs

8.3 Constants类 参考

Public 属性

- const int ERROR_NONE = 0
  没有错误
- const int ERROR_ILLEGAL_PARAMETERS = -1
  非法参数
- const int ERROR_DISCONNECTED = -2
  没有与服务器连接
- const int ERROR_UNACCEPT_PROTOCOL_VERSION = -3
  协议版本不支持
- const int ERROR_IDENTIFIER_REJECTED = -4
  标识符已拒绝
- const int ERROR_SERVER_UNAVAILABLE = -5
  服务器不可用
- const int ERROR_BAD_USERNAME_OR_PASSWD = -6
  错误的用户名和密码
- const int ERROR_UNAUTHORIZED = -7
  未认证
- const int ERROR_AUTHORIZED_SERVER_UNAVAILABLE = -8
  认证服务器不可用
- const int ERROR_OPERATION_FAILURE = -9
  操作失败
- const int ERROR_MESSAGE_TOO_BIG = -10
  消息太大
- const int ERROR_NETWORK_UNREACHABLE = -11
  网络不可达
- const int ERROR_TIMEOUT = -12
  超时
- const int ERROR_CLIENT_TYPE = -13
  clientType 有误
- const int ERROR_NOT_JSON = -14
  数据类型不是*json*

静态 Public 属性

- static string MQTT_URL = "tcp://try.iotdatahub.net:1883"

  默认的服务器地址
- static ushort KEEP_ALIVE_PERIOD = 20

  *ping*包发送间隔时间*(*秒*)*
- static double INITIAL_RECONNECT_RATE = 1000

  最小重连间隔*(*单位毫秒*)*
- static double MAX_RECONNECT_RATE = 16000

  最大重连间隔*(*单位毫秒*)*
- static String [ ] ARR_TYPE = {"json","text","bnry"}

  发送数组的类型
- static string JSON = "json"

  数据类型 *JSON*
- static string TEXT = "text"

  数据类型 *text*
- static string BINARY = "bnry"

  数据类型 *Binary*
- static string UNKNOWN = "UNKNOWN"

8.3.1 类成员变量说明

8.3.1.1 ARR_TYPE

```
String [] ARR_TYPE = {"json","text","bnry"}  [static]
```

发送数组的类型

8.3.1.2 BINARY

```
string BINARY = "bnry"  [static]
```

数据类型 Binary

8.3.1.3 ERROR_AUTHORIZED_SERVER_UNAVAILABLE

```
const int ERROR_AUTHORIZED_SERVER_UNAVAILABLE = -8
```

认证服务器不可用

8.3.1.4 ERROR_BAD_USERNAME_OR_PASSWD

```
const int ERROR_BAD_USERNAME_OR_PASSWD = -6
```

错误的用户名和密码

8.3.1.5 ERROR_CLIENT_TYPE

```
const int ERROR_CLIENT_TYPE = -13
```

clientType 有误

8.3.1.6 ERROR_DISCONNECTED

```
const int ERROR_DISCONNECTED = -2
```

没有与服务器连接

8.3.1.7 ERROR_IDENTIFIER_REJECTED

```
const int ERROR_IDENTIFIER_REJECTED = -4
```

标识符已拒绝

8.3.1.8 ERROR_ILLEGAL_PARAMETERS

```
const int ERROR_ILLEGAL_PARAMETERS = -1
```

非法参数

8.3.1.9 ERROR_MESSAGE_TOO_BIG

```
const int ERROR_MESSAGE_TOO_BIG = -10
```

消息太大

8.3.1.10 ERROR_NETWORK_UNREACHABLE

```
const int ERROR_NETWORK_UNREACHABLE = -11
```

网络不可达

8.3.1.11 ERROR_NONE

```
const int ERROR_NONE = 0
```

没有错误

8.3.1.12 ERROR_NOT_JSON

```
const int ERROR_NOT_JSON = -14
```

数据类型不是json

8.3.1.13 ERROR_OPERATION_FAILURE

```
const int ERROR_OPERATION_FAILURE = -9
```

操作失败

8.3.1.14 ERROR_SERVER_UNAVAILABLE

```
const int ERROR_SERVER_UNAVAILABLE = -5
```

服务器不可用

8.3.1.15 ERROR_TIMEOUT

```
const int ERROR_TIMEOUT = -12
```

超时

8.3.1.16 ERROR_UNACCEPT_PROTOCOL_VERSION

```
const int ERROR_UNACCEPT_PROTOCOL_VERSION = -3
```

协议版本不支持

8.3.1.17 ERROR_UNAUTHORIZED

```
const int ERROR_UNAUTHORIZED = -7
```

未认证

8.3.1.18 INITIAL_RECONNECT_RATE

```
double INITIAL_RECONNECT_RATE = 1000  [static]
```

最小重连间隔(单位毫秒)

8.3.1.19 JSON

```
string JSON = "json"  [static]
```

数据类型 JSON

8.3.1.20 KEEP_ALIVE_PERIOD

```
ushort KEEP_ALIVE_PERIOD = 20  [static]
```

ping包发送间隔时间(秒)

8.3.1.21 MAX_RECONNECT_RATE

```
double MAX_RECONNECT_RATE = 16000  [static]
```

最大重连间隔(单位毫秒)

8.3.1.22 MQTT_URL

```
string MQTT_URL = "tcp://try.iotdatahub.net:1883"  [static]
```

默认的服务器地址

8.3.1.23 TEXT

```
string TEXT = "text"  [static]
```

数据类型 text

8.3.1.24 UNKNOWN

```
string UNKNOWN = "UNKNOWN"  [static]
```

该类的文档由以下文件生成:

- Constants.cs

8.4 DataHubClient类 参考

类

- class Builder

  使用该*Builder*获取*DataHubClient*实例

Public 成员函数

- delegate void MessageEventHandler (object sender, MessageEventArgs e)

  接收消息的代理回调函数
- delegate void ConnectionStatusChangedEventHandler (object sender, ConnectionStatusChangedEventArgs e)

  连接状态改变的回调函数
- int Subscribe (string topic, byte qos, long timeout)

  订阅
- int Unsubscribe (string topic, long timeout)

  取消订阅
- int SendRequest (string topic, Message message, byte qos, long timeout, string type)

  同步发布消息
- void Destroy ()

  销毁当前客户端，并断开与服务器的连接

Public 属性

- const byte QOS0 = 0x00
- const byte QOS1 = 0x01
- const byte QOS2 = 0x02
- const byte QOS_LEVEL_AT_MOST_ONCE = 0x00
- const byte QOS_LEVEL_AT_LEAST_ONCE = 0x01
- const byte QOS_LEVEL_EXACTLY_ONCE = 0x02

事件

- MessageEventHandler MessageReceived
- ConnectionStatusChangedEventHandler ConnectionStatusChanged

8.4.1 成员函数说明

8.4.1.1 ConnectionStatusChangedEventHandler()

```
delegate void ConnectionStatusChangedEventHandler (
            object sender,
            ConnectionStatusChangedEventArgs e )
```

连接状态改变的回调函数

参数

sender	触发该事件的对象本身
e	含与服务器连接状态的 ConnectionStatusChangedEventArgs

8.4.1.2 Destroy()

```
void Destroy ( )
```

销毁当前客户端，并断开与服务器的连接

8.4.1.3 MessageEventHandler()

```
delegate void MessageEventHandler (
            object sender,
            MessageEventArgs e )
```

接收消息的代理回调函数

参数

sender	触发该事件的对象本身
e	含主题和内容的MessageEventArgs

8.4.1.4 SendRequest()

```
int SendRequest (
            string topic,
            Message message,
            byte qos,
            long timeout,
            string type )
```

同步发布消息

参数

topic	主题名
message	消息内容;长度必须小于512k。
qos	服务质量 QOS0: 消息可能到达，也可能不到达 QOS1: 消息一定会到达，但可能会重复，当然，前提是返回 Constants.ERROR_NONE QOS2: 消息一定会到达，且只到达一次，当然，前提是返回 Constants.ERROR_NONE
timeout	超时时间，单位s，必须大于0。表示该函数最多 阻塞多长时间。 对于普通的文本消息，建议超时时间为10s。 注意：该函数超时返回不代表消息发送失败，仅表示在指定时间内没有接 收到服务器的应答。
type	发送数据的类型。请在json,text,bnry,中选择一个。推荐使用json

返回

成功：返回Constants.ERROR_NONE；失败：返回错误码

8.4.1.5 Subscribe()

```
int Subscribe (
```

```
            string topic,
            byte qos,
            long timeout )
```

订阅

参数

topic	主题名
qos	订阅消息的服务质量(发送消息的qos和订阅消息的qos 共同决定服务器下发消息的qos) QOS0: 客户端最大以QOS0的服务质量接收服务器推送的消息 QOS1: 客户端最大以QOS1的服务质量接收服务器推送的消息 QOS2: 客户端最大以QOS2的服务质量接收服务器推送的消息
timeout	超时时间，单位s，必须大于0。表示该函数最多阻 塞多长时间。 对于普通的文本消息，建议超时时间为10s。 注意：该函数超时返回不代表消息发送失败，仅表示在指定时间内没有接收 到服务器的应答。

返回

成功：返回Constants.ERROR_NONE；失败：返回错误码

8.4.1.6 Unsubscribe()

```
int Unsubscribe (
            string topic,
            long timeout )
```

取消订阅

参数

topic	主题名
timeout	超时时间，单位s，必须大于0。表示该函数最多阻 塞多长时间。 对于普通的文本消息，建议超时时间为10s。 注意：该函数超时返回不代表消息发送失败， 仅表示在指定时间内没有接收到服务器的应答。

返回

成功：返回Constants.ERROR_NONE；失败：返回错误码

8.4.2 类成员变量说明

8.4.2.1 QOS0

```
const byte QOS0 = 0x00
```

最多发送一次

8.4.2.2 QOS1

```
const byte QOS1 = 0x01
```

至少发送一次，服务器可能接收到多次

8.4.2.3 QOS2

```
const byte QOS2 = 0x02
```

至少发送一次，但确保服务器只收到一次

8.4.2.4 QOS_LEVEL_AT_LEAST_ONCE

```
const byte QOS_LEVEL_AT_LEAST_ONCE = 0x01
```

至少发送一次，服务器可能接收到多次(推荐使用QOS1，以后会废弃)

8.4.2.5 QOS_LEVEL_AT_MOST_ONCE

```
const byte QOS_LEVEL_AT_MOST_ONCE = 0x00
```

最多发送一次(推荐使用QOS0，以后会废弃)

8.4.2.6 QOS_LEVEL_EXACTLY_ONCE

```
const byte QOS_LEVEL_EXACTLY_ONCE = 0x02
```

至少发送一次，但确保服务器只收到一次(推荐使用QOS2，以后会废弃)

8.4.3 事件说明

8.4.3.1 ConnectionStatusChanged

```
ConnectionStatusChangedEventHandler ConnectionStatusChanged
```

SDK与服务器连接状态改变回调函数

8.4.3.2 MessageReceived

```
MessageEventHandler MessageReceived
```

接收到消息的回调函数

该类的文档由以下文件生成:

- DataHubClient.cs

8.5 Message类 参考

属性

- byte [] payload `[get, set]`

  消息内容

8.5.1 属性说明

8.5.1.1 payload

```
byte [] payload [get], [set]
```

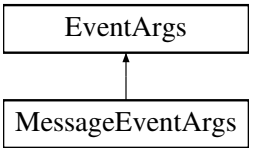
消息内容

该类的文档由以下文件生成:

- Message.cs

8.6 MessageEventArgs类 参考

类 MessageEventArgs 继承关系图:



Public 成员函数

- MessageEventArgs (string topic, byte[ ] message)

  Constructor

属性

- string Topic [get, set]

  消息的主题
- byte [ ] Message [get, set]

  消息的内容

8.6.1 构造及析构函数说明

8.6.1.1 MessageEventArgs()

```
MessageEventArgs (
            string topic,
            byte [ ] message )
```

Constructor

参数

topic	消息的topic
message	消息的内容

8.6.2 属性说明

8.6.2.1 Message

```
byte [ ] Message [get], [set]
```

消息的内容

8.6.2.2 Topic

```
string Topic [get], [set]
```

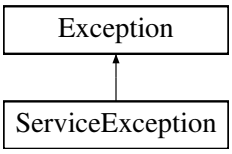
消息的主题

该类的文档由以下文件生成:

- MessageEventArgs.cs

8.7 ServiceException类 参考

类 ServiceException 继承关系图:



Public 成员函数

- ServiceException (DataHubClientErrorCode errorCode)

属性

- DataHubClientErrorCode ErrorCode [get, set]
  错误码

8.7.1 构造及析构函数说明

8.7.1.1 ServiceException()

```
ServiceException (
            DataHubClientErrorCode errorCode )
```

8.7.2 属性说明

8.7.2.1 ErrorCode

```
DataHubClientErrorCode ErrorCode [get], [set]
```

错误码

该类的文档由以下文件生成:

- ServiceException.cs

9 文件说明

9.1 ConnectionStatusChangedEventArgs.cs 文件参考

类

- class ConnectionStatusChangedEventArgs

命名空间

- namespace com.dasudian.iot.sdk

9.2 Constants.cs 文件参考

类

- class Constants

命名空间

- namespace com.dasudian.iot.sdk

9.3 DataHubClient.cs 文件参考

类

- class DataHubClient
- class DataHubClient.Builder

  使用该*Builder*获取*DataHubClient*实例

命名空间

- namespace com.dasudian.iot.sdk

9.4 Message.cs 文件参考

类

- class Message

命名空间

- namespace com.dasudian.iot.sdk

9.5 MessageEventArgs.cs 文件参考

类

- class MessageEventArgs

命名空间

- namespace com.dasudian.iot.sdk

9.6 release_note.txt 文件参考

9.7 ServiceException.cs 文件参考

类

- class ServiceException

命名空间

- namespace com.dasudian.iot.sdk

枚举

- enum DataHubClientErrorCode { EXCEPTION_ILLEGAL_PARAMETERS = -1 }

Index