Tangle Net Documentation

Release docs

Feb 18, 2019

Contents

This is a inoffical port of the IOTA Client library.

It implements all standard API calls, as described in the API documentation (https://iota.readme.io/v1.3.0/reference) and the extended methods for signing, sending and receiving bundles.

CHAPTER 1

Getting Started

1.1 Installation

Tangle.Net is compatible with .NET Standard 2.0 and .NET Framework 4.6.1.

You can install the packages via nuget

```
https://www.nuget.org/packages/Tangle.Net/
https://www.nuget.org/packages/Tangle.Net.Standard/
```

CHAPTER 2

Working with the Core

Coming soon™

CHAPTER 3

MAM

Masked Authenticated Messaging (MAM) is a second layer data communication protocol which adds functionality to emit and access an encrypted data stream. You can read more about it here: https://blog.iota.org/introducing-masked-authenticated-messaging-e55c1822d50e

3.1 Compatibility

The current C# MAM implementation is compatible to https://www.npmjs.com/package/mam.ts. Compatibility with the iota.mam.js has not been tested and may therefore not be compatible.

3.2 Channels

In the context of MAM, channels represent the sender. A channel manages the its seed, tracks its state, creates and signs messages. More about the statefullness of channels can be read below.

Creating a message through a channel and publishing it, can be done with a few lines of code. Channels should not be instantiated directly, rather they are the product of a channel factory.

```
var factory = new MamChannelFactory(CurlMamFactory.Default, CurlMerkleTreeFactory.
↪Default, iotaRepository);
var channel = factory.Create(Mode.Restricted, seed, SecurityLevel.Medium,
↪"yourchannelkey");

var message = channel.CreateMessage(TryteString.FromAsciiString("This is my first␣
↪message with MAM from CSharp!"));
await channel.PublishAsync(message);
```

3.3 Subscriptions

Subscriptions are used to listen to certain channels and retrieve messages from it. Listening do a channel can be done from any point (root) on, but not backwards. For a subscription it is not needed to know the channels seed.

```
var factory = new MamChannelSubscriptionFactory(iotaRepository, CurlMamParser.Default,
↪ CurlMask.Default);

var channelSubscription = factory.Create(new Hash("CHANNELROOT"), Mode.Restricted,
↪"yourchannelkey");
var publishedMessages = await channelSubscription.FetchAsync();
```

3.4 Serialization and State

Given the statefullness of channels and subscriptions, any application should persist the state of them. This is especially true for channels, where each message has its own index. No second message should be published to that index (similar to the address reuse issie).

The state of a channel/subscription can be retrieved by simply calling the .ToJson method. This generates a JSON representation of the channel/subscription. When recreating the channel/subscription, simply use the factories CreateFromJson method.

```
var subscrptionJson = subscription.ToJson();
var factory = new MamChannelSubscriptionFactory(iotaRepository, CurlMamParser.Default,
↪ CurlMask.Default);

var channelSubscription = factory.CreateFromJson(subscrptionJson)
```

```
var channelJson = channel.ToJson();
var factory = new MamChannelFactory(CurlMamFactory.Default, CurlMerkleTreeFactory.
↪Default, iotaRepository);

var channel = factory.CreateFromJson(channelJson)
```

3.5 Code examples

To deepen your unterstanding on how channels, subscriptions and compatibility with the TS version (see above) work, take a look at https://github.com/Felandil/tangle-.net/tree/master/Tangle.Net/Tangle.Net.Examples/Examples/Mam