Communication Channel

Generated by Doxygen 1.9.3

1 Namespace Index 1

1.1 Packages . . . 1

2 Class Index 3

2.1 Class List . . . 3

3 Namespace Documentation 5

3.1 CommunicationChannel Namespace Reference . . . 5

4 Class Documentation 7

4.1 CommunicationChannel.AntiDuplicate Class Reference . . . 7

4.1.1 Detailed Description . . . 7

4.2 CommunicationChannel.Channel Class Reference . . . 7

4.2.1 Detailed Description . . . 9

4.2.2 Constructor & Destructor Documentation . . . 9

4.2.2.1 Channel() . . . 9

4.2.3 Member Function Documentation . . . 9

4.2.3.1 IsConnected() . . . 10

4.2.4 Member Data Documentation . . . 10

4.2.4.1 LastPostParts . . . 10

4.3 CommunicationChannel.CommandsForServer Class Reference . . . 10

4.3.1 Detailed Description . . . 10

4.3.2 Member Function Documentation . . . 10

4.3.2.1 DataReceivedConfirmation() . . . 10

4.3.2.2 SendPostToServer() . . . 11

Index 13

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

CommunicationChannel 5

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

CommunicationChannel.AntiDuplicate
: In TCP connections, theoretically the connection could drop while the packet transmission is in progress, or the bidirectional connection could be interrupted by the system in one direction only: In iOS it happens that when the app is in the background the outgoing communication is interrupted while the incoming one remains active, and the device cannot confirm that it has received the packets, so that the server, having no acknowledgment of receipt, resends the packets that have actually already been received. 7

CommunicationChannel.Channel
: This class handle all the communication channel operation with server-side. 7

CommunicationChannel.CommandsForServer
: This class handle the commands to be executed at the server level. 10

Chapter 3

Namespace Documentation

3.1 CommunicationChannel Namespace Reference

Classes

- class AntiDuplicate

  In TCP connections, theoretically the connection could drop while the packet transmission is in progress, or the bidirectional connection could be interrupted by the system in one direction only: In iOS it happens that when the app is in the background the outgoing communication is interrupted while the incoming one remains active, and the device cannot confirm that it has received the packets, so that the server, having no acknowledgment of receipt, resends the packets that have actually already been received.
- class Channel

  This class handle all the communication channel operation with server-side.
- class CommandsForServer

  This class handle the commands to be executed at the server level.
- class **Converter**

  This class converts the data from one format to another as per the task requirement.
- class **Protocol**
- class **Spooler**

  This class is used for saving, updating the data in the queue list.
- class **Tcp**

  This class is used for establishing link connection and communicating with the server.
- class **Utility**

  This is an utility class which provides method to converting data, hashing data or get neccesity information.

Chapter 4

Class Documentation

4.1 CommunicationChannel.AntiDuplicate Class Reference

In TCP connections, theoretically the connection could drop while the packet transmission is in progress, or the bidirectional connection could be interrupted by the system in one direction only: In iOS it happens that when the app is in the background the outgoing communication is interrupted while the incoming one remains active, and the device cannot confirm that it has received the packets, so that the server, having no acknowledgment of receipt, resends the packets that have actually already been received.

Public Member Functions

- **AntiDuplicate** ()

  Constructor of class

4.1.1 Detailed Description

In TCP connections, theoretically the connection could drop while the packet transmission is in progress, or the bidirectional connection could be interrupted by the system in one direction only: In iOS it happens that when the app is in the background the outgoing communication is interrupted while the incoming one remains active, and the device cannot confirm that it has received the packets, so that the server, having no acknowledgment of receipt, resends the packets that have actually already been received.

The documentation for this class was generated from the following file:

- C:/documentation/CryptoMessenger/Crypto-Messenger/CommunicationChannel/AntiDuplicate.cs

4.2 CommunicationChannel.Channel Class Reference

This class handle all the communication channel operation with server-side.

Public Member Functions

- Channel (string serverAddress, int domain, Func< bool > contextIsReady, Action< ulong, byte[]> on↩ MessageArrives, Action< uint > onDataDeliveryConfirm, ulong myId, int connectionTimeout=Timeout.↩ Infinite)

  Initialize the library
- bool IsConnected ()

  checks the connection status.

Static Public Member Functions

- static void **ReEstablishConnection** ()

  Use this command to re-establish the connection if it is disabled by the timer set with the initialization

Public Attributes

- CommandsForServer **CommandsForServer**

  class object to use command at server-side.
- readonly Uri **ServerUri**

  server URL.
- readonly int **Domain**

  Server domain id.
- bool **LogError** = true

  Set this true if you want a ErrorLog
- int LastPostParts
- int **PostCounter**

  Number of posts
- int **DuplicatePost**

  Number of duplicate post
- string **ErrorLog**

  Display all the error logs

Properties

- bool **ClientExists** `[get]`

  TCP client exists
- bool **ClientConnected** `[get]`

  TCP client connection status
- bool **Logged** `[get]`

  Client log in status
- int **QueeCount** `[get]`

  Number of bytes in the queue
- ulong **KeepAliveFailures** `[get, set]`

  Number of failure connnection
- static bool **InternetAccess** `[get, set]`

  Check internet access.

Events

- Action< string > **RefreshLogError**

  Action to refresh error log.

4.2.1 Detailed Description

This class handle all the communication channel operation with server-side.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Channel()

```
CommunicationChannel.Channel.Channel (
            string serverAddress,
            int domain,
            Func< bool > contextIsReady,
            Action< ulong, byte[]> onMessageArrives,
            Action< uint > onDataDeliveryConfirm,
            ulong myId,
            int connectionTimeout = Timeout.Infinite )
```

Initialize the library

Parameters

serverAddress	Server Address
domain	A domain corresponds to a membership group. Using the domain it is possible to divide the traffic on a server into TestNet, MainNet group (in order to isolate the message circuit within a given domain).
contextIsReady	function that checks if the client is ready
onMessageArrives	Event that is raised when a message arrives.
onDataDeliveryConfirm	Event that is generated when the router (server) has received the outgoing message, This element returns the message in raw format
myId	The identifier of the current user. Since the server system is focused on anonymity and security, there is no user list, it is a cryptographic id generated with a hashing algorithm
connectionTimeout	Used to remove the connection when not in use. However, mobile systems remove the connection when the application is in the background so it makes no sense to try to keep the connection always open. This also lightens the number of simultaneous server-side connections.

4.2.3 Member Function Documentation

4.2.3.1 IsConnected()

```
bool CommunicationChannel.Channel.IsConnected ( )
```

checks the connection status.

Returns

True or False

4.2.4 Member Data Documentation

4.2.4.1 LastPostParts

```
int CommunicationChannel.Channel.LastPostParts
```

The documentation for this class was generated from the following file:

- C:/documentation/CryptoMessenger/Crypto-Messenger/CommunicationChannel/Channel.cs

4.3 CommunicationChannel.CommandsForServer Class Reference

This class handle the commands to be executed at the server level.

Public Member Functions

- void SendPostToServer (ulong chatId, byte[ ] dataToSend, bool directlyWithoutSpooler=false)
  Send data to the server.
- void DataReceivedConfirmation (byte[ ] dataReceived)
  Confirmation that data is recieved at the server side.

4.3.1 Detailed Description

This class handle the commands to be executed at the server level.

4.3.2 Member Function Documentation

4.3.2.1 DataReceivedConfirmation()

```
void CommunicationChannel.CommandsForServer.DataReceivedConfirmation (
            byte[] dataReceived )
```

Confirmation that data is recieved at the server side.

Parameters

dataReceived	data to recieve confirmation

4.3.2.2 SendPostToServer()

```
void CommunicationChannel.CommandsForServer.SendPostToServer (
            ulong chatId,
            byte[] dataToSend,
            bool directlyWithoutSpooler = false )
```

Send data to the server.

Parameters

chatId	chat to which data belong to
dataToSend	data
directlyWithoutSpooler	if you want to send directly without spooler make it true else false

The documentation for this class was generated from the following file:

- C:/documentation/CryptoMessenger/Crypto-Messenger/CommunicationChannel/CommandsForServer.cs

Index

Channel
    CommunicationChannel.Channel, 9
CommunicationChannel, 5
CommunicationChannel.AntiDuplicate, 7
CommunicationChannel.Channel, 7
    Channel, 9
    IsConnected, 9
    LastPostParts, 10
CommunicationChannel.CommandsForServer, 10
    DataReceivedConfirmation, 10
    SendPostToServer, 11

DataReceivedConfirmation
    CommunicationChannel.CommandsForServer, 10

IsConnected
    CommunicationChannel.Channel, 9

LastPostParts
    CommunicationChannel.Channel, 10

SendPostToServer
    CommunicationChannel.CommandsForServer, 11