# AKM .NET library

## Required configuration

AKM .NET library requires specific configuration items present in application’s configuration file (like appsettings.json)

"AkmAppConfigs": [

{

"SelfAddressValue": 1,

"DefaultKeySize": 32,

"NumberOfKeys": 4,

"RelationshipId": 1,

"FrameSchema": {

"RelationshipId\_Index": 0,

"RelationshipId\_Length": 2,

"SourceAddress\_Index": 2,

"SourceAddress\_Length": 2,

"TargetAddress\_Index": 4,

"TargetAddress\_Length": 2,

"AkmEvent\_Index": 6,

"AkmEvent\_Length": 1,

"AkmDataStart\_Index": 7

},

"InitialKeys": [

{

"InitialKey": "6v9y$B&E)H+MbQeThWmZq4t7w!z%C\*F-"

},

{

"InitialKey": "z$C&F)H@McQfTjWnZr4u7x!A%D\*G-KaN"

},

{

"InitialKey": "6v9y$B&E)H+MbQeMhWmZq4t7w!z%C\*Fo"

},

{

"InitialKey": "z$C&F)H@McQfTjWaZr4u7x!A%D\*G-Kat"

}

],

"NodesAddresses": [ 1, 5 ]

}

]

This information is used to configure AKM library and state machine for specific Relationship group based on Relationship Id value.

## Basic interaction objects:

AKM.NET provides two classes used for network communication over NetworkStream object in .NET environment: Sender and Receiver.

Instance of Receiver class can be created directly by providing NetworkStream, CancellationToken and ILogger objects.

\_client = await server.AcceptTcpClientAsync();

\_stream = \_client.GetStream();

\_receiver = new Receiver(\_stream,stoppingToken,\_logger);

As an optional parameter own implementation of ICryptography interface can be provided to customize security hash calculation method as well as encryption and decryption algorithms.

public interface ICryptography

{

int HashLength { get; }

byte[] CalculateHash(byte[] data);

byte[] Decrypt(byte[] dataToDecrypt, byte[] key);

byte[] Encrypt(byte[] dataToEncrypt, byte[] key);

}

Receiver type exposes DataReceived event which is fired after AKM frame data is received and processed. This event’s argument contains byte array with content send by sender node and Relationship Id value.

public class AkmDataReceivedEventArgs : EventArgs

{

/// <summary>

/// Byte array with transmission content

/// </summary>

public byte[] FrameData { get; set; }

/// <summary>

/// Relationship Id value

/// </summary>

public short RelationshipId { get; set; }

}

Handling this event is client’s application responsibility – AKM does not process data sent in its frame in any way. Sample usage is presented here:

private void OnDataReceived(object sender, AkmDataReceivedEventArgs e)

{

if ((e?.FrameData.Length ?? 0) > 0)

{

//Own processing of received data

}

}

(...)

\_receiver.DataReceived += OnDataReceived;

To activate Receiver you need to invoke the StartReceiving method

\_receiver.StartReceiving(cancellationToken);

Sender object is used to wrap data in AKM Frame structure and send it over provided NetworkStream. Sender object should not be created directly, but taken from AkmSenderManager class based on Relationship Id value, or by using GetDefaultSender method when Relationship Id is irrelevant.

If you want to send data as a reaction for DataReceived event then you should use RelationshipId from that event to get proper Sender from AKmSenderManager class

var sender = AkmSenderManager.GetSender(relationshipId);

If you want to start communication and have only single Relationship group defined in your application configuration file you can use the GetDefaultSender method

var sender = AkmSenderManager.GetDefaultSender();

In both cases you will need to provide at least NetworkStream object and ILogger implementation object for the Sender using its Set… methods.

sender.SetNetworkStream(stream);

sender.SetLogger(\_logger);

If you want to use your own cryptography provider you can set it as well using the SetCryptography method.

ICryptography cryptographyProvider;

sender.SetCryptography(cryptographyProvider);

To see if Sender object is configured properly you can use IsInitialized property

if (sender.IsInitialized) {...}

To send data simply use SendData method which accepts byte array as the data that should be sent and target node numeric address value

sender.SendData(messageBytes, 1);

This will result in creating AKM frame structure and transmitting it over provided NetworkStream.