

AS4 .NET

Configuration Manual



# Table of Contents

[Table of Contents 3](#_Toc465259003)

[Document history 4](#_Toc465259004)

[1. Purpose 5](#_Toc465259005)

[2. Introduction 6](#_Toc465259006)

[2.1 Package 6](#_Toc465259007)

[2.1.1 Config Folder 6](#_Toc465259008)

[2.1.2 Database Folder 6](#_Toc465259009)

[2.1.3 Messages Folder 6](#_Toc465259010)

[2.1.4 Documentation Folder 7](#_Toc465259011)

[2.1.5 Samples Folder 7](#_Toc465259012)

[2.2 Settings.xml 7](#_Toc465259013)

[2.2.1 GUID Format 7](#_Toc465259014)

[2.2.2 Database Provider 7](#_Toc465259015)

[2.2.3 Certificate 7](#_Toc465259016)

[2.2.4 Agents 8](#_Toc465259017)

[2.2.5 Custom Settings 8](#_Toc465259018)

[2.3 Sending Processing Mode 9](#_Toc465259019)

[2.4 Receiving Processing Mode 13](#_Toc465259020)

[3. Getting Started 16](#_Toc465259021)

[3.1 Sending Simple AS4 Message 16](#_Toc465259022)

[3.2 Sending Normal AS4 Message 17](#_Toc465259023)

[3.3 Sending Full-Blown AS4 Message 17](#_Toc465259024)

[4. Extra 18](#_Toc465259025)

[4.1 Configure Logging 18](#_Toc465259026)

[4.2 Install Certificates 19](#_Toc465259027)

# Document history

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Created by** | **Short Description of Changes** |
| v1 | 26/10/2016 | Stijn Moreels | Initial version |
| v2 | 26/10/2016 | Stijn Moreels | Updating Pmodes, adding screenshots in Extra paragraph and adding deliver/notify strategies in examples |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Purpose

The purpose of this document is to give a simple introduction to the AS4 .NET Component itself and how to configure the component.

# Introduction

Following items describe all the introduction folders and files that you find in the package.

## Package

The package itself is divided in several folders:

**config**

**database**

**messages**

**documentation (manual + xsd)**

**sample**

**component.exe**

In the root of the package you find the .exe file that runs the component. Next up all the folders are being explained.

### Config Folder

Inside the configuration folder, following structure is created:

**receive-pmodes**

**send-pmodes**

**settings.xml**

The folders send/receive-pmodes are the folders which configured the pmodes (respectively send/receive). Samples of this pmodes can be found in the “sample”-folder (see 2.1.5 Samples Folder).

The **settings.xml** file contains the global configuration of the component and will be explained in 2.2 Settings.xml.

### Database Folder

Default **SQLLite** is used as database. The .db file which contains the SQLLite database is stored in this folder.

### Messages Folder

Inside the messages folder, following structure is created:

**attachments**

**errors**

**exceptions**

**receipts**

**in**

**out**

The **attachmens** folder contains several files (pictures and .xml documents) that’s being used as reference for the send AS4 messages. The **receipts/errors/exceptions** folders are used to store **Notify Messages**. The **in** folder is used to store incoming messages and attachments and the **out** folder is being used to send messages to another MSH (the .xml file will be renamed to “.accepted” if it’s being retrieved by the component)

### Documentation Folder

Inside the documentation folder, following structure is created:

**schemas**

**configuration-manual.docx**

Inside the **schemas** folder the .xsd files are located of the **pmodes** and **messages**. In the root of this folder, this document (**configuration-manual.docx**) is located.

### Samples Folder

Inside the **samples** folder, following structure is created:

**certificates**

**messages**

**receive-pmodes**

**send-pmodes**

Each folder contains the respectively the samples of send/receive-pmodes and messages. Inside the **certificates** folder, you find sample certificates that can be used for sending (signing/encrypting) and receiving (verifying) messages.

## Settings.xml

The **settings.xml** located inside the **config** folder contains several global configuration settings used inside the component. Each kind of setting is explained in the following paragraphs.

### GUID Format

When creating AS4 Messages, Message Ids are being generated. To configure the format in which this must be done the **<IdFormat/>** tag is being used inside the **settings.xml**.

Default: {GUID}@{IPADDRESS}

### Database Provider

The component can be configured to store messages and exceptions in another datastore. Inside the **settings.xml** the **<Database/>** tag is responsible for this. In the two child’s of this tag you define the **<Provider/>**, which can be *SQLLite*, *SQLServer*,… any type which is supported in **Entity Framework Core**; and the **<ConnectionString/>** which defines the actual connection to the database.

So, let say you want to change the provider to store the messages in a SQL Server database; you must change the **<Provider/>** to **“SqlServer”** and the **<ConnectionString/>** to a valid SQL Server Connection String: **Server=myServerAddress;Database=myDataBase;Trusted\_Connection=True;**.

Default Provider: Sqllite

Default Connection String: Filename=database\messages.db

### Certificate

To support signing and encrypting messages, certificates are needed. The certificate store that’s needed to retrieve this certificates and be set in the **<CertificateStore/>** tag.

Default: My

The default implementation is used to retrieve certificates from a certificate store on a Windows environment; but you can write your own implementation. If you on a Windows environment (so the default implementation is OK for you) you can define here in which Store you want to search. Following values can be used:

|  |  |
| --- | --- |
| Member Name | Description |
| AddressBook | The X.509 certificate store for other users. |
| AuthRoot | The X.509 certificate store for third-party certificate authorities (CAs). |
| CertificateAuthority | The X.509 certificate store for intermediate certificate authorities (CAs). |
| Disallowed | The X.509 certificate store for revoked certificates. |
| My | The X.509 certificate store for personal certificates. |
| Root | The X.509 certificate store for trusted root certificate authorities (CAs). |
| TrustedPeople | The X.509 certificate store for directly trusted people and resources. |
| TrustedPublisher | The X.509 certificate store for directly trusted publishers. |

### Agents

The AS4 protocol has several operations: Submit, Send, Receive, Deliver and Notify. All of these operations are configured in the **settings.xml** as **Agents**. Each agent has three items which defines the agent: **Receiver**, **Transformer** and **Step(s)**.

A **Receiver** can be configured in the **<Receiver/>** tag in each agent. There are multiple kinds of receivers: FileReceiver, DatastoreReceiver, HttpReceiver… Each needed to be configured in order to work correctly. This can be done in the child’s of this tag as a **<Setting/>** (with attribute **key**; the inner text of the tag is the **value**).

A **Transformer** can be configured in the **<Transformer/>** tag in each agent. This transformer is needed in order to transform the received message (could be .xml, .json…) and transform it to a AS4 Message that can be used in the **Step(s)**.

A **Step** or **Steps** can be configured in the **<Steps/>** tag in each agent. These steps will be executed after the message is being transformed. Example of steps are: **CreateReceiptStep**, **CompressAttachmentsStep**, **DecryptAS4MessageStep**…

To support exception handling, **decorators** are created. Each step you define will be used to decorate except you define an **undecorated=true** attribute on the step. In that case, the step is not decorated and will be executed after the decorator is executed.

Each tag (**Receiver**, **Transformer** and **Step**/**Decorator**) has a **type** attribute which defines the type of which the instance must be created inside the component.

### Custom Settings

When creating custom implementations of types, settings can sometimes be useful. For example, an **EmailSender** by which you configure the **SMTP Server** in the **settings.xml**. This can be useful instead of hardcoded each configured value in the implementation itself.

## Sending Processing Mode

This contract describes all the properties available in the Sending PMode. The required data fields are marked as mandatory, default values are provided. Some values of the Sending PMode can be overridden by a [SubmitMessage](#_Submit_Message). This definition is available as XSD.

|  |  |  |
| --- | --- | --- |
| Sending PMode | \* | Description |
| **Id** | M | PMode Unique Id |
| **AllowOverride** | M | Boolean indication whether a [SubmitMessage](#_Submit_Message) may override already configured values within the sending PMode.  *Default:* false |
| **MEP** | M | Message Exchange Pattern  *Enumeration:*   * one-way |
| **MEPBinding** | M | Message Exchange Pattern Binding  *Enumeration:*   * push * pull |
| **PushConfiguration**  *Protocol*  URL  UseChunking  UseHTTPCompression  *TLSConfiguration*  IsEnabled  TLSVersion  ClientCertificateReference | O  M  M  O  O  O  O  O  O | URL of the receiving MSH  *Default:* false (true > not implemented)  *Default:* false (true > not implemented)  *Default:* false  *Default:* TLS 1.2  *Enumeration:*   * SSL 3.0 * TLS 1.0 * TLS 1.1 * TLS 1.2   Reference to certificate store |
| **PullConfiguration**  SubChannel | O  O | *Default:* NULL |
| **Reliability**  *ReceptionAwareness*  IsEnabled  RetryCount  RetryInterval | O  O  O  O  O | *Default:* false  *Default:* 5  *Default:* 00:01:00 (HH:mm:ss) |
| **ReceiptHandling**  NotifyMessageProducer  NotifyMethod  *Type* *Parameters*  *Parameter*  Name  Value | O  O  M  M  M  M  M  M | *Default:* false  Type of the Notify AgentRequired parameters for the specified agent  Name of the parameter  Value of the parameter |
| **ErrorHandling**  NotifyMessageProducer  NotifyMethod  *Type*  *Parameters*  *Parameter*  Name  Value | O  O  M  M  M  M  M  M | *Default:* false  Type of the Notify Agent  Required parameters for the specified agent  Name of the parameter  Value of the parameter |
| **ExceptionHandling**  NotifyMessageProducer  NotifyMethod  *Type*  *Parameters*  *Parameter*  Name  Value | O  O  M  M  M  M  M  M | *Default:* false  Type of the Notify Agent  Required parameters for the specified agent  Name of the parameter  Value of the parameter |
| **Security**  *Signing*  IsEnabled  PrivateKeyFindType      PrivateKeyFindValue  KeyReferenceMethod    Algorithm  HashFunction  *Encryption*  IsEnabled  PublicKeyFindType    PublicKeyFindValue  Algorithm  KeyTransport | O  O  O  M  M  M  M  M  O  O  M  M  M  O | *Default:* false  *Enumeration:*   * FindByThumbprint * FindBySubjectName * FindBySubjectDistinguishedName * FindByIssuerName * FindByIssuerDistinguishedName * FindBySerialNumber * FindByTimeValid * FindByTimeNotValid * FindByTimeNotYetValid * FindByTimeExpired * FindByTemplateName * FindByApplicationPolicy * FindByCertificatePolicy * FindByExtension * FindByKeyUsage * FindBySubjectKeyIdentifier   *Enumeration:*   * BTSReference * IssuerSerial * KeyIdentifier   *Default*: false  *Enumeration:*   * FindByThumbprint * FindBySubjectName * FindBySubjectDistinguishedName * FindByIssuerName * FindByIssuerDistinguishedName * FindBySerialNumber * FindByTimeValid * FindByTimeNotValid * FindByTimeNotYetValid * FindByTimeExpired * FindByTemplateName * FindByApplicationPolicy * FindByCertificatePolicy * FindByExtension * FindByKeyUsage * FindBySubjectKeyIdentifier |
| **Message Packaging**  Mpc  UseAS4Compression  IsMultiHop  IncludePModeId  PartyInfo  *FromParty*  *PartyIds*  *PartyId*  Id  Type  Role  *ToParty*  *PartyIds*  *PartyId*  Id  Type  Role  CollaborationInfo  *AgreementRef*  Value  Type  *Service*  Value  Type  Action  MessageProperties  *MessageProperty*  Name  Type  Value | O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O | *Default:* true  *Default:* false  *Default:* false  Id of the sending party  Type of Id of the sending party  Role of the sending party  Id of the receiving party  Type of Id of the receiving party  Role of the receiving party  Information about the partner agreement  Type of the agreement reference  The name of the service that is consumed  Type of the service  The service operation that is consumed  Name of the message property  Type of the message property  Value of the message property |

(\*): M = Mandatory | O = Optional | R = Recommended

## Receiving Processing Mode

This contract describes all the properties available in the Sending PMode. The required data fields are marked as mandatory; default values are provided. This definition is available as XSD.

|  |  |  |
| --- | --- | --- |
| Receive PMode | \* | Description |
| **Id** | M | PMode Unique Id |
| **MEP** | M | Message Exchange Pattern  *Enumeration:*   * one-way |
| **MEPBinding** | M | Message Exchange Pattern Binding  *Enumeration:*   * push * pull |
| **Reliability**  *DuplicateElimination*  IsEnabled | O  O  O | *Default:* false |
| **ReceiptHandling**  UseNNRFormat  ReplyPattern  SendingPMode | O  M  M  M | *Default:* false  *Enumeration:*   * Response: sync response (*default*) * Callback: async response   Reference to the Sending PMode (used for signing and the URL) |
| **ErrorHandling**  UseSOAPFault  ReplyPattern  SendingPMode  ResponseHTTPCode  NotifyMessageConsumer  NotifyMethod  *Type*  *Parameters*  *Parameter*  Name  Value | O  M  M  M  O  M  O  O  O  O  O  O | *Default:* false  *Enumeration:*   * Response: sync response (*default*) * Callback: async response   Reference to the Sending PMode (used for signing and the URL)  HTTP Status Code in case of reply = response  *Default*: false  Type of the Notify Agent  Required parameters for the specified agent  Name of the parameter  Value of the parameter |
| **ExceptionHandling**  NotifyMessageConsumer  NotifyMethod  *Type*  *Parameters*  *Parameter*  Name  Value | O  O  M  M  M  M  M  M | *Default:* false  Type of the Notify Agent  Required parameters for the specified agent  Name of the parameter  Value of the parameter |
| **Security**  *SigningVerification*  Signature    *Decryption*  Encryption    PrivateKeyFindType    PrivateKeyFindValue | O  O  M  O  O  M  M | *Enumeration*   * Allowed (*default)* * Not allowed * Required * Ignored   *Enumeration*   * Allowed (*default)* * Not allowed * Required * Ignored   *Enumeration:*   * FindByThumbprint * FindBySubjectName * FindBySubjectDistinguishedName * FindByIssuerName * FindByIssuerDistinguishedName * FindBySerialNumber * FindByTimeValid * FindByTimeNotValid * FindByTimeNotYetValid * FindByTimeExpired * FindByTemplateName * FindByApplicationPolicy * FindByCertificatePolicy * FindByExtension * FindByKeyUsage * FindBySubjectKeyIdentifier |
| **Message Packaging**  PartyInfo  *FromParty*  *PartyIds*  *PartyId*  Id  Type  Role  *ToParty*  *PartyIds*  *PartyId*  Id  Type  Role  CollaborationInfo  *AgreementRef*  Value  Type  *Service*  Value  Type  Action | O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O  O | Id of the sending party  Type of Id of the sending party  Role of the sending party  Id of the receiving party  Type of Id of the receiving party  Role of the receiving party  Information about the partner agreement  Type of the agreement reference  The name of the service that is consumed  Type of the service  The service operation that is consumed |
| **Deliver**  IsEnabled  PayloadReferenceMethod  *Type*  *Parameters*  *Parameter*  Name  Value  DeliverMethod  *Parameters*  *Parameter*  Name  Value | M  M  M  M  M  M  M  M  M  M  M  M | Default: *true*  Type of Payload Reference  Type of the Deliver Agent  Required parameters for the specified reference  Name of the parameter  Value of the parameter  Type of the Deliver Agent  Required parameters for the specified agent  Name of the parameter  Value of the parameter |

(\*): M = Mandatory | O = Optional | R = Recommended

# Getting Started

Now the basics are explained, we can now start with some example setups to sending messages.

To send/receive messages from two different MSH’s you can use your own MSH configuration or just copy the root folder to another name:

**AS4 .NET Sender**

**AS4 .NET Receiver**

When we have two instances, we can configure the one for sending and another for receiving. When you’re done copying the **Sender/Receiver** folders for the Component Instances; we can continue with the configuration of the two.

The two instances having an own **HttpReceiver** in place which can be hosted on a port. Default is this **8080**. It’s logical that we can’t have to HTTP endpoints on the same port, that’s why you must configure one of the two to another port. (Note: the samples are made so that the **Receiver** instance is on port 9090 and the **Sender** on port 8080).

This can be configured in the **AS4 .NET Receiver\config\settings.xml** in the **Receive Agent’s** **HTTP Receiver**. In the settings of this receiver, change the port to 9090.

## Sending Simple AS4 Message

The first example consists of a AS4 message with a single payload that’s being send from one **AS4 .NET Instance** to another. This example is the simplest form of sending a message, it contains no compression, signing or encryption.

In the **samples\pmodes** folder (doesn’t matter what configured instance), you can find several pmode examples. For this test, we use the **01-sample-receive-pmode.xml** and **01-sample-send-pmode.xml** pmodes.

Copy the **01-sample-receive-pmode.xml** to the configured **AS4 .NET Receiver** instance inside the **config\send-pmodes** folder; and copy the **01-sample-send-pmode.xml** to the configured **AS4 .NET Sender** instance inside the **config\receive-pmodes**.

Now that the pmodes are configured, we can now start with sending the message. Go the configure **AS4 .NET Sender** to the **samples\messages** folder and copy the **01-sample-message.xml** to the configured **AS4 .NET Sender** in the **messages\out** folder.

When the two instances are running, the message will now be send to the configured **AS4 .NET Receiver** and will be stored in the **messages\in** folder of this instance together with the attachment that was being send with it.

The sending/receiving pmodes in this example are configured to notify/deliver the “Business Application” through the file system. Any **Receipts** received are notified in the **messages\receipts** folder, any **Errors** received are notified in the **messages\errors** folder and any **Exceptions** generated are notified in the **messages\exceptions** folder.

## Sending Normal AS4 Message

The second example consists of a AS4 Message with multiple payloads that’s being send from one **AS4 .NET Instance** to another. This example uses compression and signing of the message and will return a normal receipt to the configured **AS4 .NET Sender**.

Because this example contains signing, the sample certificates must be installed to the target machine. These certificates can be found in the **samples\certificates** folder. To install these certificates (see 4.2 Install Certificates).

In the **samples\pmodes** folder (doesn’t matter what configured instance), you can find several pmode examples. For this test, we use the **02-sample-receive-pmode.xml** and **02-sample-send-pmode.xml** pmodes.

Copy the **02-sample-receive-pmode.xml** to the configured **AS4 .NET Receiver** instance inside the **config\send-pmodes** folder; and copy the **02-sample-send-pmode.xml** to the configured **AS4 .NET Sender** instance inside the **config\receive-pmodes**.

Now that the pmodes are configured, we can now start with sending the message. Go the configure **AS4 .NET Sender** to the **samples\messages** folder and copy the **02-sample-message.xml** to the configured **AS4 .NET Sender** in the **messages\out** folder.

When the two instances are running, the message will now be send to the configured **AS4 .NET Receiver** and will be stored in the **messages\in** folder of this instance together with the attachment that was being send with it.

The sending/receiving pmodes in this example are configured to notify/deliver the “Business Application” through the file system. Any **Receipts** received are notified in the **messages\receipts** folder, any **Errors** received are notified in the **messages\errors** folder and any **Exceptions** generated are notified in the **messages\exceptions** folder.

## Sending Full-Blown AS4 Message

The third example consists of a AS4 Message with multiple payloads that’s being send from one **AS4 .NET Instance** to another. This example uses compression, signing and encryption of the message and will return a NRR (Non-Repudiation Receipt) to the configured **AS4 .NET Sender**.

Because this example contains signing, the sample certificates must be installed to the target machine. These certificates can be found in the **samples\certificates** folder. To install these certificates (see 4.2 Install Certificates).

In the **samples\pmodes** folder (doesn’t matter what configured instance), you can find several pmode examples. For this test, we use the **03-sample-receive-pmode.xml** and **03-sample-send-pmode.xml** pmodes.

Copy the **03-sample-receive-pmode.xml** to the configured **AS4 .NET Receiver** instance inside the **config\send-pmodes** folder; and copy the **03-sample-send-pmode.xml** to the configured **AS4 .NET Sender** instance inside the **config\receive-pmodes**.

Now that the pmodes are configured, we can now start with sending the message. Go the configure **AS4 .NET Sender** to the **samples\messages** folder and copy the **03-sample-message.xml** to the configured **AS4 .NET Sender** in the **messages\out** folder.

When the two instances are running, the message will now be send to the configured **AS4 .NET Receiver** and will be stored in the **messages\in** folder of this instance together with the attachment that was being send with it.

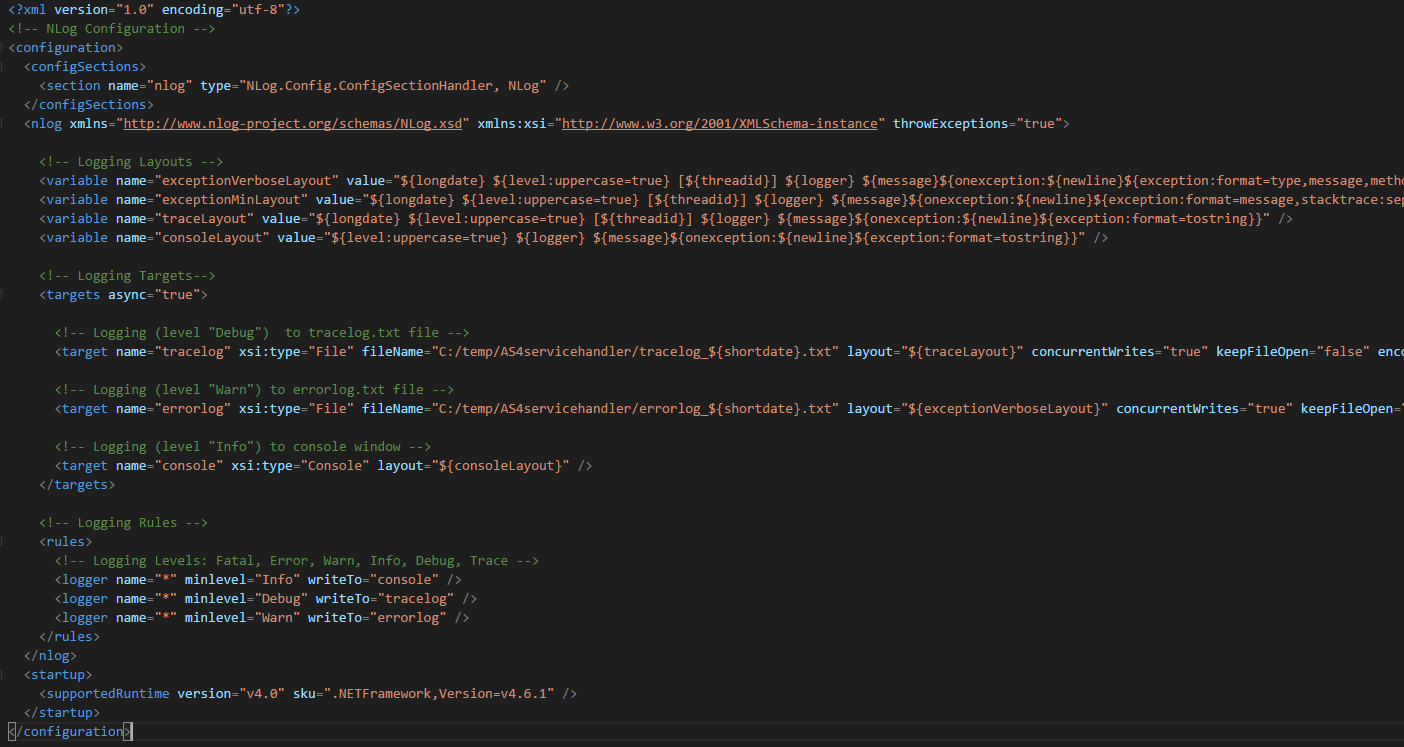
The sending/receiving pmodes in this example are configured to notify/deliver the “Business Application” through the file system. Any **Receipts** received are notified in the **messages\receipts** folder, any **Errors** received are notified in the **messages\errors** folder and any **Exceptions** generated are notified in the **messages\exceptions** folder.

# Extra

## Configure Logging

The AS4 .NET Component uses **NLog** for logging. The logging levels can be configured to minimum: **Info**, and maximum: **Debug**. Further configuration can be found on the **NLog** support itself (<https://github.com/nlog/nlog/wiki/Configuration-file> )

The sample NLog configuration file is named: **App.config**, below you find the default configuration:



## Install Certificates

The AS4 .NET Component contains several sample certificates (but your own certificates can be used as well). The certificate must be installed on the target machine in order to used it. **This must be installed inside the “Trusted Certificates” inside the “Microsoft Management Console” in a Windows environment.**

