SigningHub
Installation Guide

SigningHub 8.1.0

ASCERTIA LTD

FEBRUARY 2022

DOCUMENT VERSION – 1.0.0.1

This document contains commercial-in-confidence material. It must not be disclosed to any third party without the written authority of Ascertia Limited.

CONTENTS

1 Introduction 4

1.1 Scope 4
1.2 Intended Readership 4
1.3 Conventions 4
1.4 Technical Support 5
1.5 Glossary 5
1.6 References to PKI Standards 5

2 System Requirements 6
3 Installation Modules 10

3.1 Deployment Scenario 11

4 Prerequisites 11

4.1 SMTP Server 11
4.2 Windows Roles and Features (2016, 2012 R2, 2012) 12
4.3 URL Rewrite Module 17
4.4 Additional Windows Configurations (2016, 2012 R2, 2012) 17
4.5 ADSS Signing Server Dependencies 17
4.6 Database 18

5 SigningHub Enterprise Installation 18

5.1 Fresh Installation of SigningHub Enterprise 18
5.2 Installing SigningHub Enterprise with a Load-Balanced Configuration 46
5.3 Installing SigningHub Enterprise with an Existing Database 53
5.4 Post Installation Steps 65
5.5 Upgrading SigningHub Enterprise 66
5.6 Changing Database Credentials for an Existing Installation 79
5.7 Troubleshooting 82

6 SigningHub Enterprise Uninstallation 83
Appendix A - Configuring AJP Connector for Local Signing 86

A.1 Prepare the Packages 86
A.2 Add ISAPI Filter for SigningHub Enterprise 87
A.3 Add Virtual Directory 88
A.4 Register ISAPI Extension 91
A.5 Update ADSS Signing Server & SigningHub Enterprise Configuration 94

Appendix B - Securing SigningHub Desktop Web 95

B.1 Securing cookies 95
B.2 'X-XSS-Protection' header 95
B.3 HTTP Strict Transport Security (HSTS) header 95
B.4 HTTP Public Key Pinning header 96
B.5 TLS Fallback SCSV 96
B.6 SSL Medium Strength Cipher Suites 96
B.7 Hiding Application Errors and Server Information 96
B.8 Content Security Policy Header 97
B.9 Referrer Policy Header 98
B.10 'X-FRAME-OPTIONS' Response Header 98
B.11 Cacheable HTTPS Response 98
B.12 CAPTCHA Configurations 98

Appendix C - Securing SigningHub API 99
C.1 'X-XSS-Protection' header 99
C.2 Content Security Policy Header 99
C.3 'X-FRAME-OPTIONS' Response Header 99
C.4 Cacheable HTTPS Response 100
Appendix D - Application Settings 100
D.1 Desktop Web 100
D.2 Mobile Web 101
D.3 API 102
D.4 Admin 104
Appendix E - Securing SigningHub Mobile Web 104
E.1 Securing cookies 104
E.2 'X-XSS-Protection' header 104
E.3 Content Security Policy Header 104
E.4 'X-FRAME-OPTIONS' Response Header 105
E.5 Cacheable HTTPS Response 105
Appendix F - Configuring SigningHub Demo Site 105
Appendix G - Proxy Settings in Internet Explorer 106
Appendix H - Installing Redis Server 107
H.1 Issues in Existing Redis Installation 107
H.2 Uninstall Redis Service 107
H.3 Installation of Latest Redis Server 108
Appendix I - Enable Transparent Data Encryption (TDE) 117
I.1 1 Introduction 117
I.2 How It Works? 117
I.3 Create Master Key 118
I.4 Create Certificate Protected by Master Key 118
I.5 Create Database Encryption Key 118
I.6 Enable Encryption 118
I.7 Backup Certificate 118
I.8 Restoring Certificate 119
I.9 Restoring Certificate 119
I.10 Limitations and Restrictions on TDE 120
Appendix J - Enable Transport Security Layer (TLS) 121
J.1 Introduction 121
J.2 How It Works? 121
J.3 Configuration for Certificate Enrollment 122
J.4 Installation of SQL Server Certificate Using Microsoft Management Console 125
J.5 Configuration of SQL Server to Use Encrypted Connections 130
J.6 Post Configurations Steps 132
J.7 Configurations Required for SigningHub Connection String 134
J.8 Verification of TLS Configurations 135
J.9 Update Existing SigningHub Instance Connection String Over TLS 135

1 Introduction

SigningHub Enterprise is a complete solution for document approval workflows, advanced digital signatures and document status tracking. It is designed to quickly optimise the way businesses deliver, review, approve and sign their business documents.

The following online resources supplement the contents herein, and provide in depth user and administration guides:

- http://manuals.ascertia.com/SigningHubv7
- http://manuals.ascertia.com/SigningHub-admin

SigningHub Enterprise is powered by ADSS Signing Server. This is a comprehensive solution for creating and verifying advanced digital signatures on any type of document, web form or transaction.

1.1 Scope

This manual describes the installation process for SigningHub Enterprise and ADSS Signing Server that underpins SigningHub Enterprise.

SigningHub Enterprise comprises seven components and the installation procedure for all are covered herein:

- **Desktop Web** interface that provides user services on desktop browsers.
- **Admin** console that provides system administration and configuration.
- **API** that utilises the ASP.NET Web API framework to provide a REST architecture.
- **Mobile Web** interface that provides user services on mobile browsers.
- **Website Integration Demo** to illustrate tight integration using the REST architecture.
- **Core** sends email notifications as a background process against few particular events and also performs the housekeeping operations.

1.2 Intended Readership

This manual is intended for SigningHub Enterprise administrators who are responsible for the installation, configuration, and maintenance of the system. It is assumed that the reader has a basic knowledge of Microsoft Internet Information Services (IIS) based web applications, digital signatures, digital certificates and general security.

1.3 Conventions

These typographical conventions are used in this guide:

- **Bold** text identifies menu names, menu options, items you can click on the screen, file names, folder names, and keyboard keys.
- `Courier New` font identifies code and text that appears on the command line.
- `Courier New` or `Courier New` identifies single line commands or code that you are required to type in.

1.4 Technical Support

If technical support is required, Ascertia has a dedicated support team providing debugging and integration assistance as well as general customer support. Ascertia Support can be accessed through Ascertia Ticketing System or email address: support@ascertia.com

Ascertia provides formal support agreements with all product sales. Contact sales@ascertia.com for further details.

A Product Support Questionnaire should be completed in order to provide Ascertia Support with information about your system environment, along with details of any issues encountered. When requesting help, it is always important to confirm these details:

- System Platform.
- SigningHub Enterprise version number.
- Details of the specific issue and relevant steps taken to reproduce it if possible.
- Database vendor, version and patch level.
- Product log files.

1.5 Glossary

The following table illustrates glossary terms and their definitions.

Term	Definition
ADSS Signing Server	ADSS Signing Server is an Ascertia product that provides a wide range of PKI cryptographic services and digital signature creation and verification and supplementary trust services. SigningHub Enterprise uses this product to power its underlying digital signature functionality.
IIS	Internet Information Services

1.6 References to PKI Standards

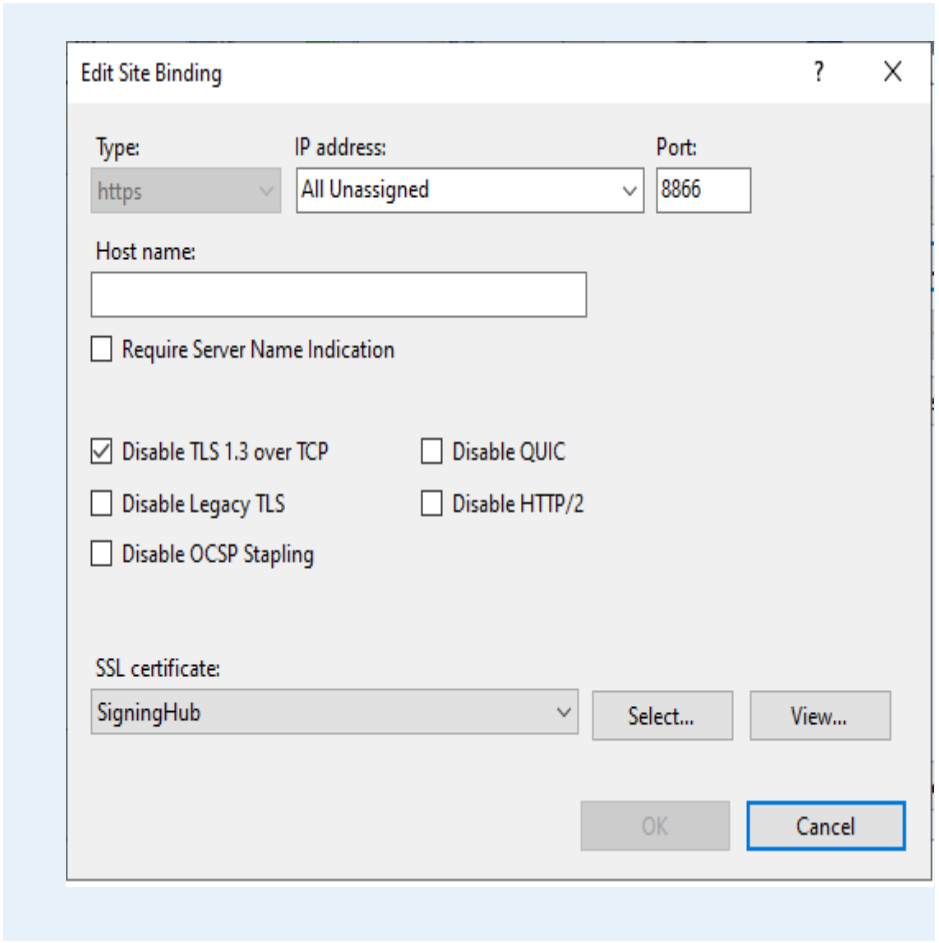
The following table illustrates the PKI Standards and their references.

PKI Standards	Reference
PDF Signatures	PDF Public Key Digital Signature and Encryption Specification v3.2 http://www.adobe.com/devnet/pdf/pdf_reference.html
PKCS#7	http://www.faqs.org/rfcs/rfc2315.html
PKCS#11	ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20.pdf

2 System Requirements

The following table defines the system requirements for SigningHub Enterprise:

Components	Requirements
Server System	SigningHub Enterprise is a Microsoft .NET Framework application and is supported on these operating systems: • Windows Server 2022 • Windows Server 2019 • Windows Server 2016 • Windows Server 2012 R2 • Windows Server 2012 IIS is a mandatory component. The versions stated below are supported. For more information refer to 4.Prerequisites: • IIS 10 • IIS 8.5 • IIS 8.0 • Microsoft .NET Framework v4.8 or above • (https://dotnet.microsoft.com/download/dotnet-framework/net48) Hardware: A modern multi-core CPU such as the Xeon E56xx or the E3 or E5 range is recommended, with a minimum of 16GB RAM and 200 GB disk space required for the application and its logs. 32GB RAM and higher is needed to support a larger population of concurrent users. In case you are Using Microsoft Windows 2022 Server and Installing SigningHub. **TLS 1.3** is by default ON on Windows Server 2022 (integrated applications should support TLS 1.3). Client Certificate Authentication is not supported with TLS 1.3 and as a workaround disable TLS 1.3 by Enabling the checkbox to **Disable TLS 1.3 over TCP** in your IIS Bindings.



Database Server	SigningHub Enterprise supports these databases to store the configurations data: Oracle 12c Enterprise Edition, Microsoft Azure SQL Server, Microsoft SQL Server 2019, 2017, 2016 (Express, Web Edition, Standard or Enterprise) SigningHub Enterprise uses Dapper framework so support for other databases can be considered for specific projects – ask for further information. SigningHub Enterprise database can be installed on the same machine as the SigningHub Enterprise application. However, for optimal performance it is recommended to use a dedicated host, either virtual or physical. Hardware: A modern multi-core CPU such as Xeon E3-xxxx or E5-xxxx series is recommended, with a minimum 16GB RAM (24GB RAM recommended). In addition, 30GB of disk space as a minimum for document storage of File System, and 50GB for Database Storage, with the upper level determined by the online storage requirement.

	In case you are installing SigningHub using Microsoft SQL Server 2016, upgrade the database instance to latest service pack, which is SP2. To download the SP2 service pack, visit Microsoft's Download Center.
Client Machines (systems using the service)	Windows 10, 8, 7 — Browser support for central and local signing Linux (any) — Browser support for central and local signing Mac OS X 10+ — Browser support for central and local signing iPad (iOS 9+) — SigningHub App supports central signing only iPhone (iOS 9+) — SigningHub App supports central signing only Android 5+ — SigningHub App supports central signing only
Web Brower (for end-users and administrators)	The following browsers are supported: • Internet Explorer IE 11 • Edge 14+ • Firefox 35+ • Chrome 40+ • Safari 8+ (Local signing is not available for Safari installed on Windows platform.) • Opera 26+

The following table defines the system requirements for ADSS Signing Server:

Components	Requirements
ADSS Signing Server	ADSS Signing Server 6.0.x or above is a mandatory component of the solution. It provides the underlying security services required by SigningHub Enterprise. Note that ADSS Signing Server may be installed on the same server platform as SigningHub Enterprise and is silently installed as part of the installation process. However, a dedicated, separate installation of ADSS Signing Server can also be used. Database requirements for ADSS Signing Server differ to those from SigningHub Enterprise. Please refer to the ADSS Signing Server documentation suite for more information: http://manuals.ascertia.com/ADSS-Admin-Guide. Regardless of the deployment model chosen, SigningHub Enterprise and ADSS Signing Server must have their own respective databases. ADSS Signing Server v6.5 or above is required when upgrading to SigningHub v7.7.7 or above. This is required to

	have a backward compatibility with SigningHub application, due to a change in existing implementation of CSP services.
Admin/Operator web browsers	For ADSS Signing Server Admin interface these browsers are supported: • Internet Explorer 9+ • Edge 14+ • Chrome 40+ • Firefox 35+
Optional HSMs	The following Hardware Security Modules (HSMs) are supported: • Thales SafeNet Luna SA, Luna PCI, Luna G5 • Thales SafeNet Protect Server (PCI or External) • nCipher nShield Solo or Connect HSMs • Utimaco HSMs • Azure Key Vault • Amazon AWS Cloud HSM
DMZ Proxy Systems	A DMZ proxy server is recommended to provide enhanced security for SigningHub Enterprise. Supported web servers are: • Windows Server + IIS or Apache or IBM HTTP Server • Linux + Apache or IBM HTTP Server Ascertia recommends a reasonable CPU, 4GB RAM, 1000 MB disk space for the web server host. SigningHub Enterprise and ADSS Signing Server support network proxies to allow authenticated access to external services. Signing with local smartcards or USB Tokens requires ADSS Signing Server Go>Sign Service, which requires Apache Jakarta Protocol (AJP) connector on the proxy server (if used). See Appendix A for details.

SigningHub Enterprise, ADSS Signing Server, and the database can all be installed on the same system for testing purposes. However, for optimal performance and production systems Ascertia recommends that the database should be installed on a dedicated host. If required ADSS Signing Server can also be installed on a separate server.

The details given above are the minimum system requirements; these may need to be revised to meet specific performance requirements. SigningHub Enterprise and ADSS Signing Server can also be configured in load-balanced mode to provide higher resilience and throughput.

3 Installation Modules

SigningHub Enterprise consists of the following modules. Note the API and Demo components are the only non-mandatory ones for a working solution:

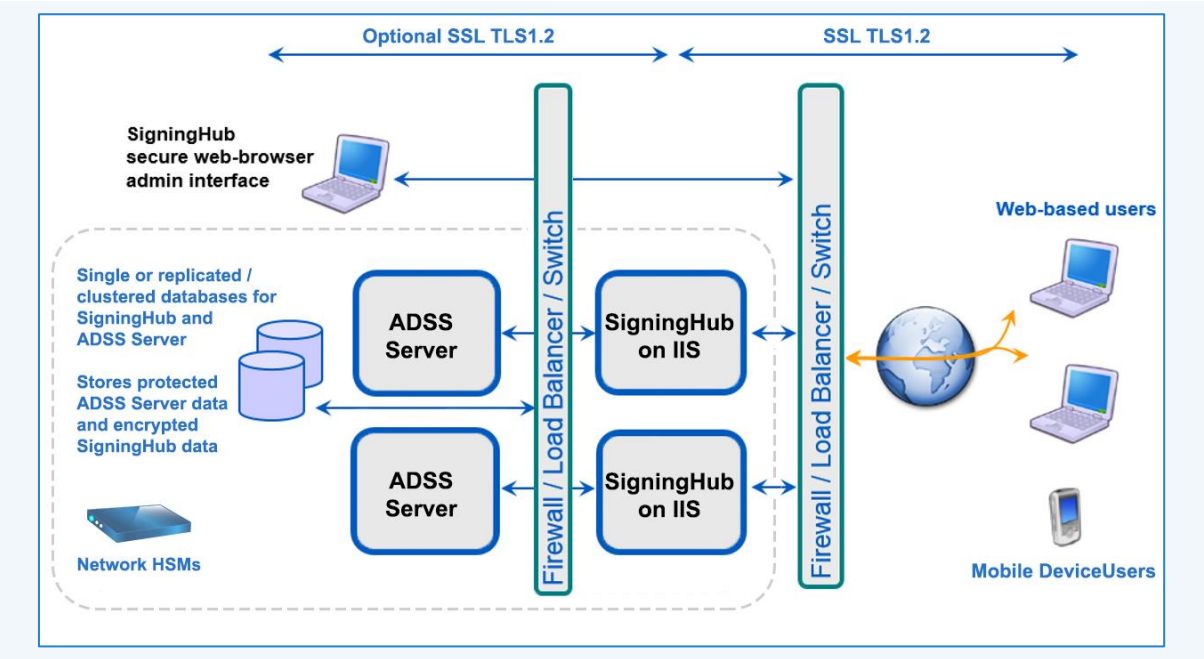
- SigningHub Enterprise Desktop Web application
- Main web application, which provides the client-facing functionality for document workflow approval/sign-off and user account management.
- SigningHub Enterprise Admin Console
- Administration application that allows central secure management of the global settings and registered users. This application is only accessible to registered administrators and operators authenticated using client-server mutual TLS/SSL.
- SigningHub Enterprise API
- REST architecture API support that is used to integrate SigningHub Enterprise functionality within your own portal. The API uses OAuth 2.0 to implement authentication and authorisation. There is a separate API Guide that provides full details of the REST architecture implementation, see details.
- SigningHub Enterprise Core
- The SigningHub enterprise Core is used for back ground processes e.g. email sending, reminders, summary emails, document deletion etc.
- SigningHub Enterprise Mobile Web
- Web application for mobile browsers, it provides the client-facing functionality for document workflow approval/sign-off and user account management.
- SigningHub Enterprise Website Integration Demo
- A demonstration application to illustrate REST architecture API.
- ADSS Signing Server
- This provides the underlying PKI security services for SigningHub Enterprise application. These include cryptographic keys and trust anchors management, server-side signatures creation and their verification, and other auxiliary services.

For local signing ADSS Go>Sign Desktop is required.

ADSS Signing Server is only licensed for use with SigningHub Enterprise unless specifically agreed otherwise in writing.

3.1 Deployment Scenario

A typical SigningHub Enterprise deployment looks like this:



Database clustering is supported for both ADSS Signing Server and SigningHub Enterprise.

4 Prerequisites

4.1 SMTP Server

SigningHub uses email as the primary notification medium. User registration, and all notifications are sent via SMTP. Hence it is a critical part of the architecture and deployment. Details required are:

- Hostname/IP address of SMTP server
- Listening Port of SMTP server
- TLS/SSL authentication to communicate with SMTP server (if required)
- User name and password to authenticate to SMTP server (if required)
- Email from Address for notifications sent from SigningHub
- Email to Address for alerts and warnings sent by SigningHub
- Email Subject for alerts and warnings sent by SigningHub

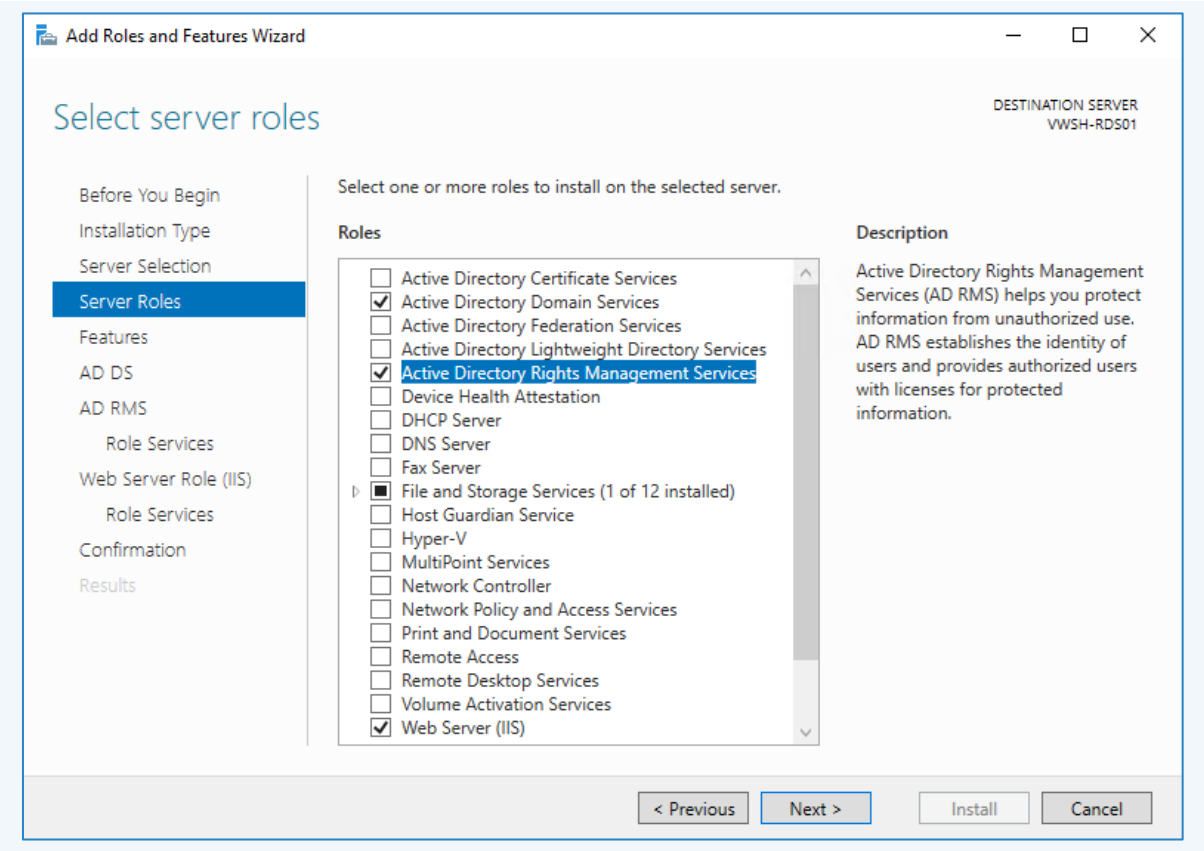
If there is no alternative it is possible to still use SigningHub. However, this involves copying the notification emails directly from the database and manually running the links therein. This usage is strongly discouraged in favour of a standard deployment though.

4.2 Windows Roles and Features (2016, 2012 R2, 2012)

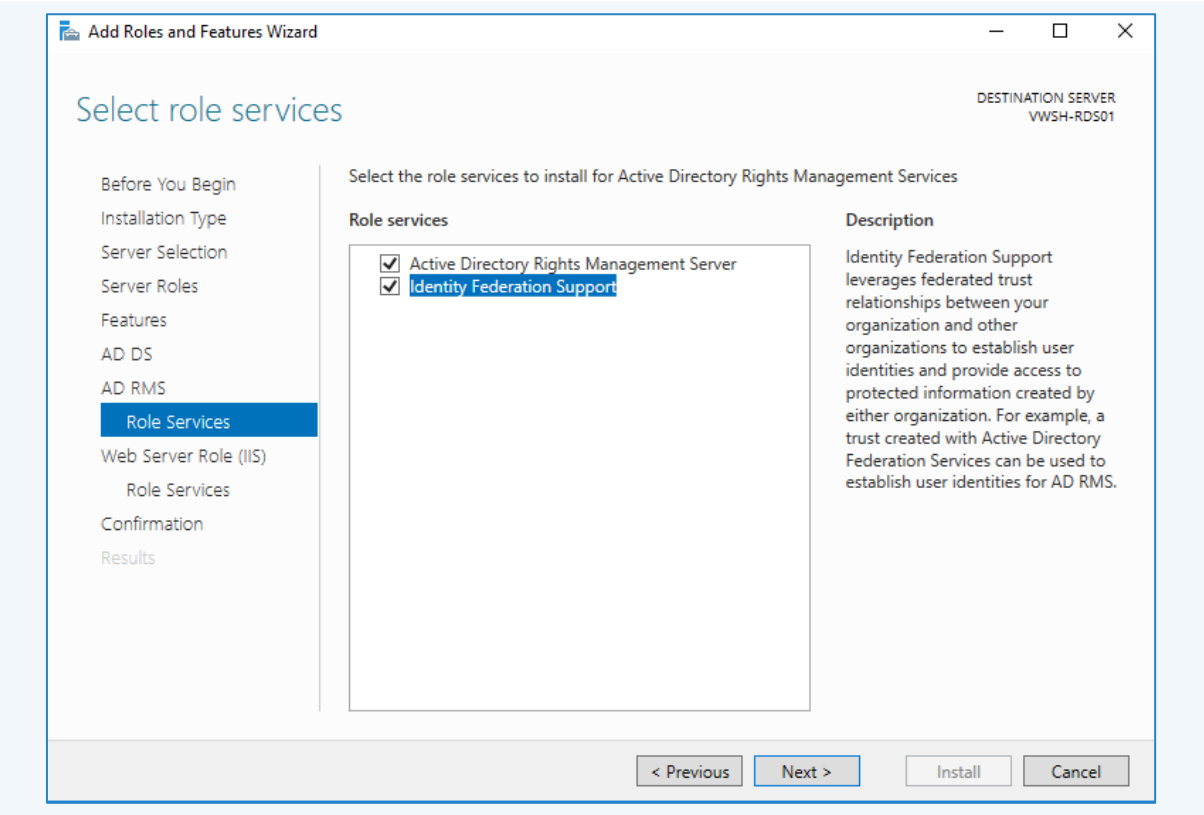
SigningHub Enterprise has dependencies on many Windows roles and features etc. These must be installed before deploying SigningHub Enterprise. Required dependencies are added via the Windows Server Manager and are detailed here.

Windows dependencies are added via Server Manager.

1. Open the **Server Manager** application and select the **Manage** menu.
2. Under this menu choose **Add Roles and Features** option.
3. On the **Add Roles and Feature Wizard**, click the **Next** button thrice so you reach Server Roles dialog as show in the screen:

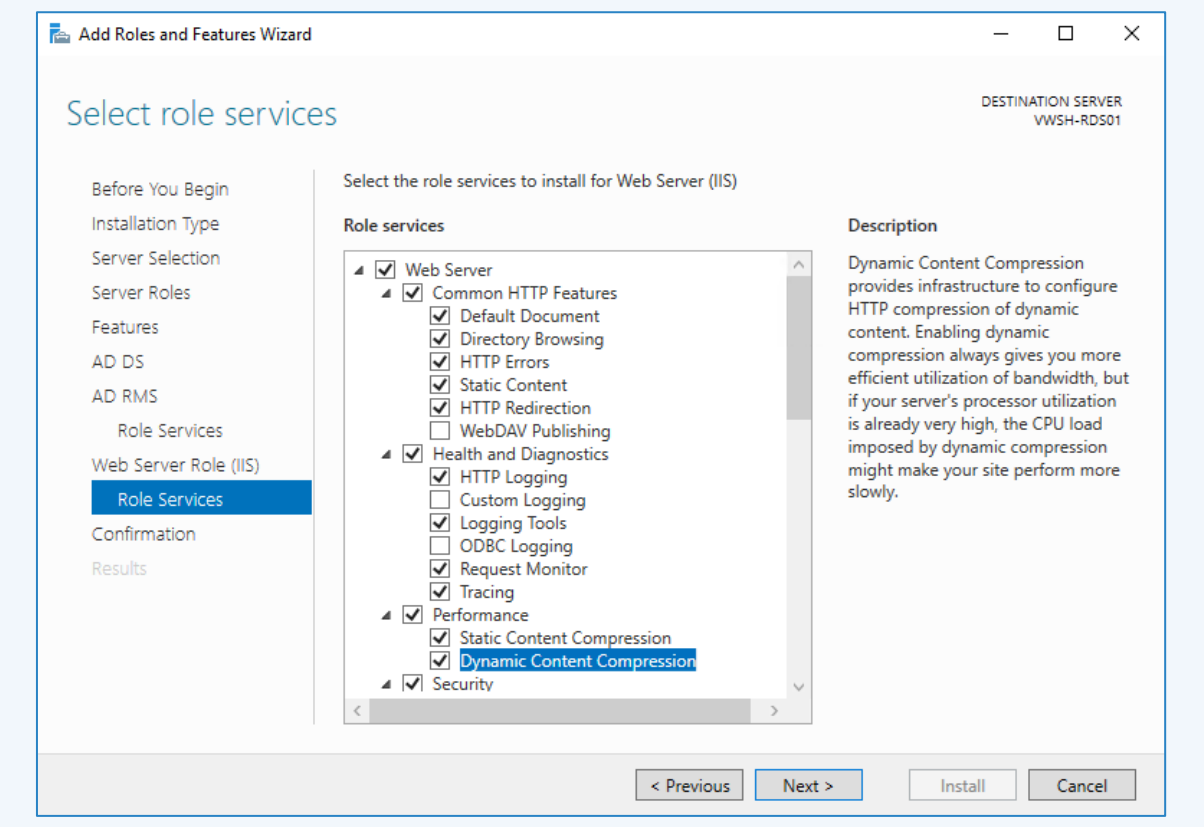


4. Click the **Next** button until you reach the **Roles Services** screen as below:

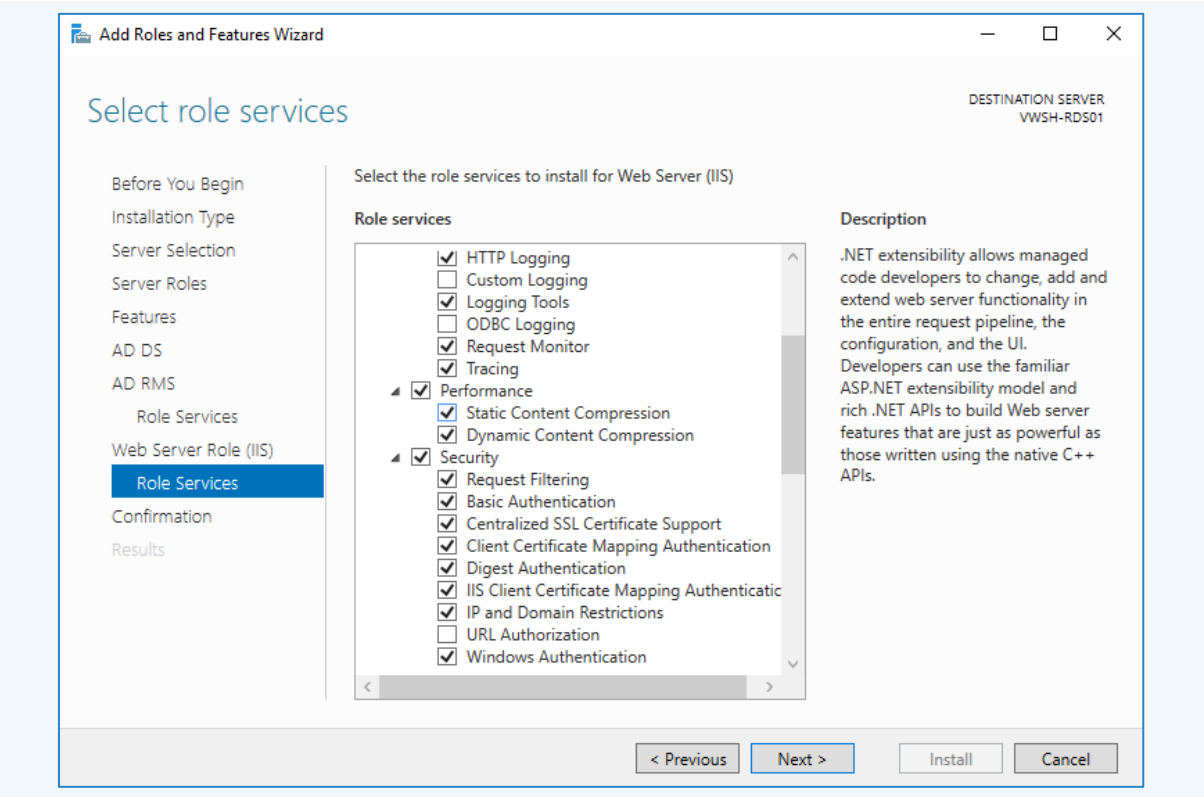


5. Select **Identity Federation Support** and click the **Next** button to select the IIS features that are shown in the next 4 screenshots.

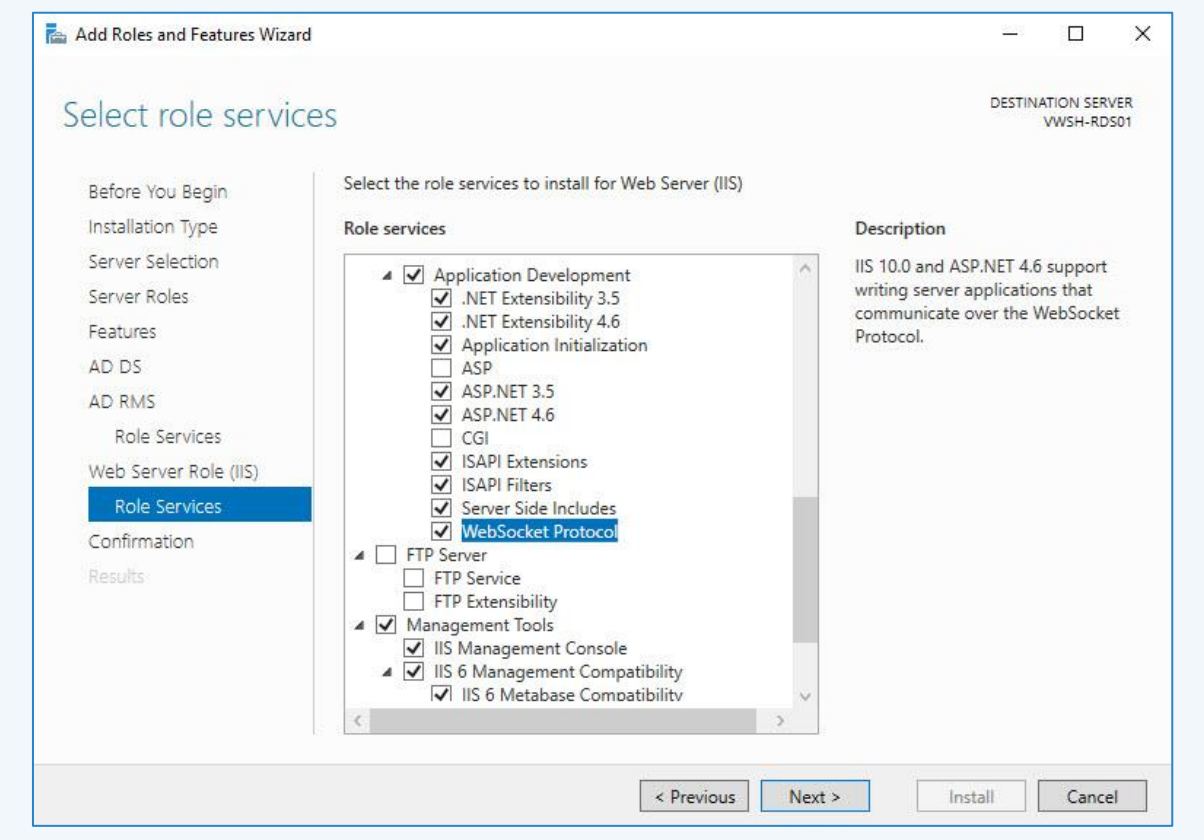
6. Select the items ticked, these are mandatory items for SigningHub Enterprise if Active Directory is used to authenticate the SigningHub Enterprise users.



7. Scroll down to select next options.

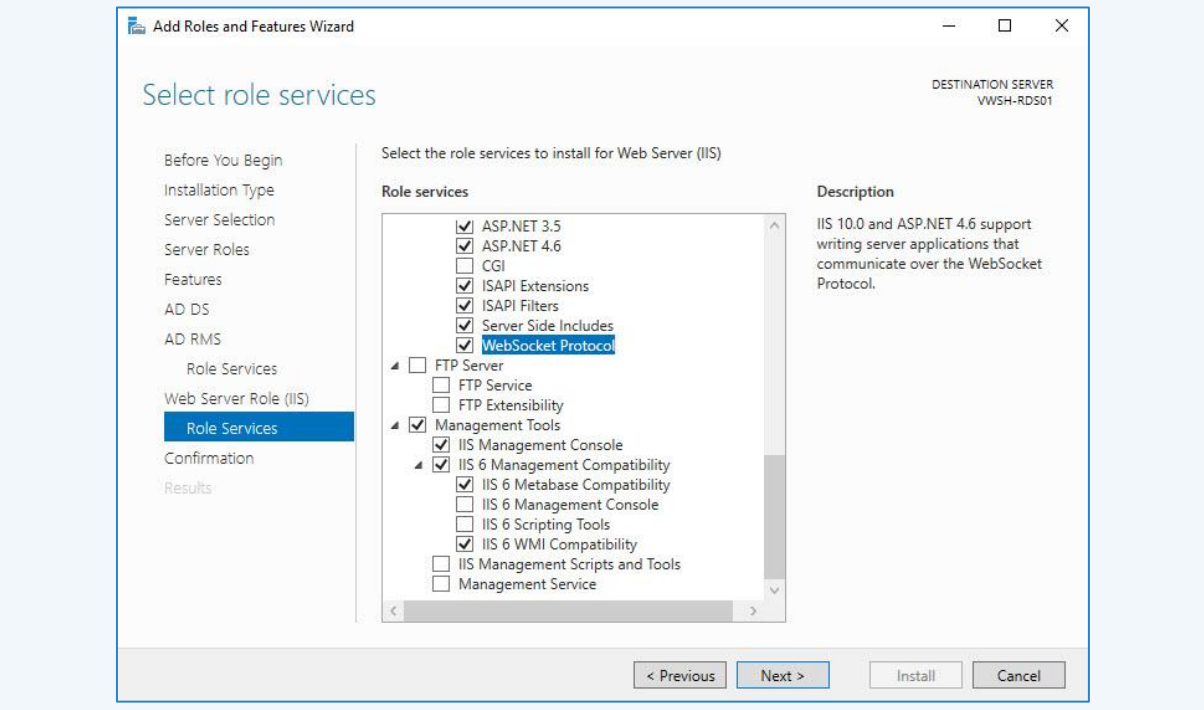


8. Scroll down to select next options.

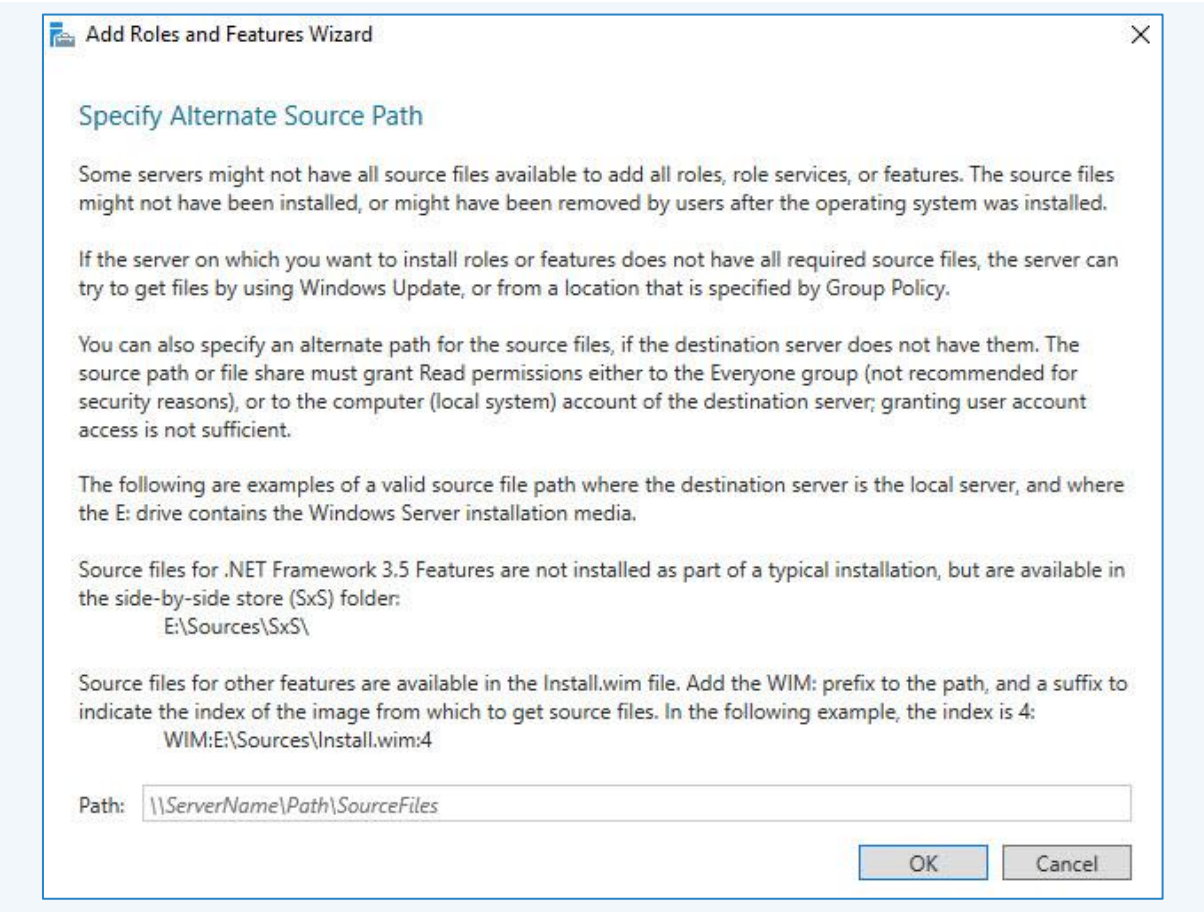


9. Scroll down to select next options.

10.



If the Windows operating system media is not available directly on the server, you may be prompted to provide the source path on the Confirm Installation Selections window:



11. If necessary, provide the correct path to the installation media and click **OK.**

12. Once the roles and features have been successfully deployed IIS must be restarted.

13. Restart IIS Server from IIS Management Console for the changes to take effect. Ensure that the IIS server is restarted and not individual application pools or web sites.

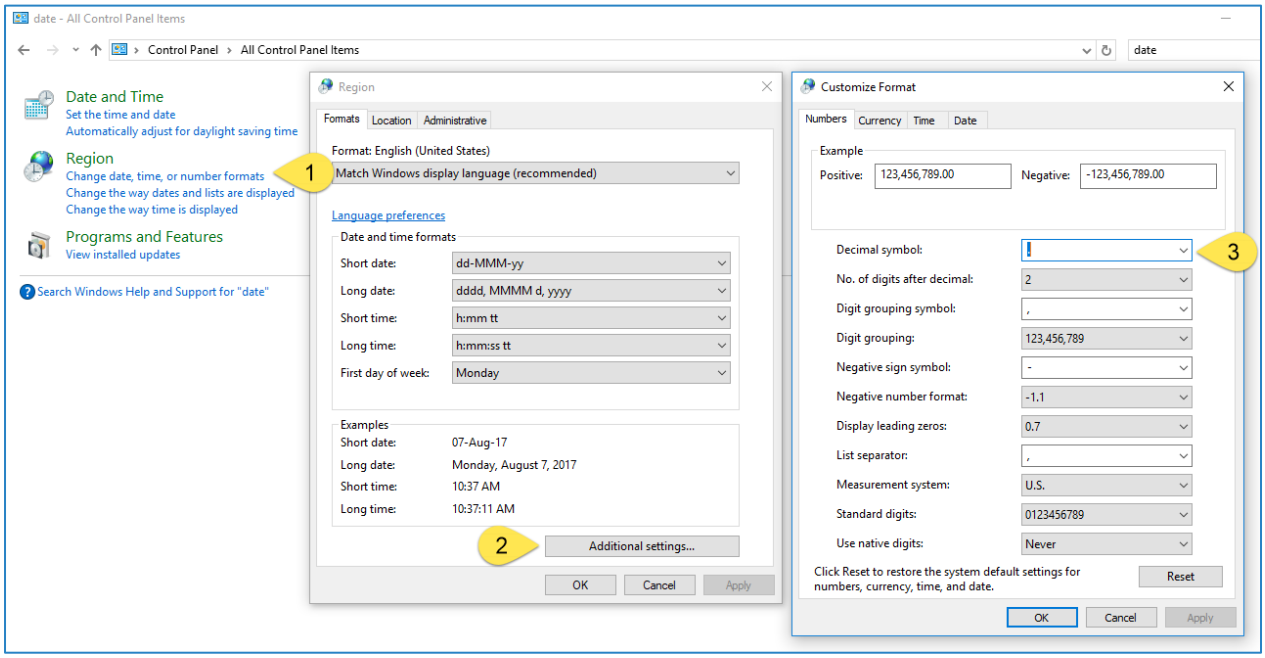
4.3 URL Rewrite Module

This module is required to open SigningHub in responsive design mode, when opened on a mobile device. Click here to download the URL Rewrite module from Microsoft.com

This is sample text for adding special notes in the document To install URL Rewrite Module, a live connection is required to get URL Re-write. Click here to download or use an offline version.

4.4 Additional Windows Configurations (2016, 2012 R2, 2012)

Make sure that the Decimal Symbol is dot (.) instead of comma (,) in **Control Panel > Region> Additional Settings** as shown in the following screenshot:



4.5 ADSS Signing Server Dependencies

ADSS Signing Server has no Windows dependencies like those required for SigningHub Enterprise. This is because it is a Java EE application that runs off the bundled Tomcat server. However, a database is still required and optionally an HSM to secure private key material.

If you are using an external CA, then ADSS Signing Server must be configured appropriately. Otherwise SigningHub Enterprise will not function as expected. Click here for complete details of ADSS Server.

4.6 Database

Both SigningHub Enterprise and ADSS Signing Server require their own respective databases. It is not needed to create the schema or configure any other feature prior to the installation.

Permissions are required to allow the creation of database tables, and entry, modification, and removal of data within those tables.

4.6.1 SQL Server

For details about SQL Server installation, user creation and permissions, refer to

Microsoft-SQL-Server-Installation-Guide-for-ADSS-Server.pdf `in [SigningHub-Home]/tool/adss-server/docs` directory.

4.6.2 Oracle

The following privileges are required to install the application with Oracle DBMS:

- connect
- dba
- resource

Note that the same level of database permissions are required for SigningHub Enterprise and ADSS Signing Server databases and users.

5 SigningHub Enterprise Installation

5.1 Fresh Installation of SigningHub Enterprise

The installation script must be run from a user account with the Windows Administrator privileges.

The SigningHub Enterprise package must be unzipped on to a disk that has sufficient space – a minimum of **100GB** is recommended. This is because the product is installed and runs from where the installation package is extracted to. Hence please choose a suitable location and naming structure.

Note do not include any spaces in the installation folder name and path - use hyphen or underscore characters instead if required. Spaces will cause functional problems with SigningHub Enterprise.

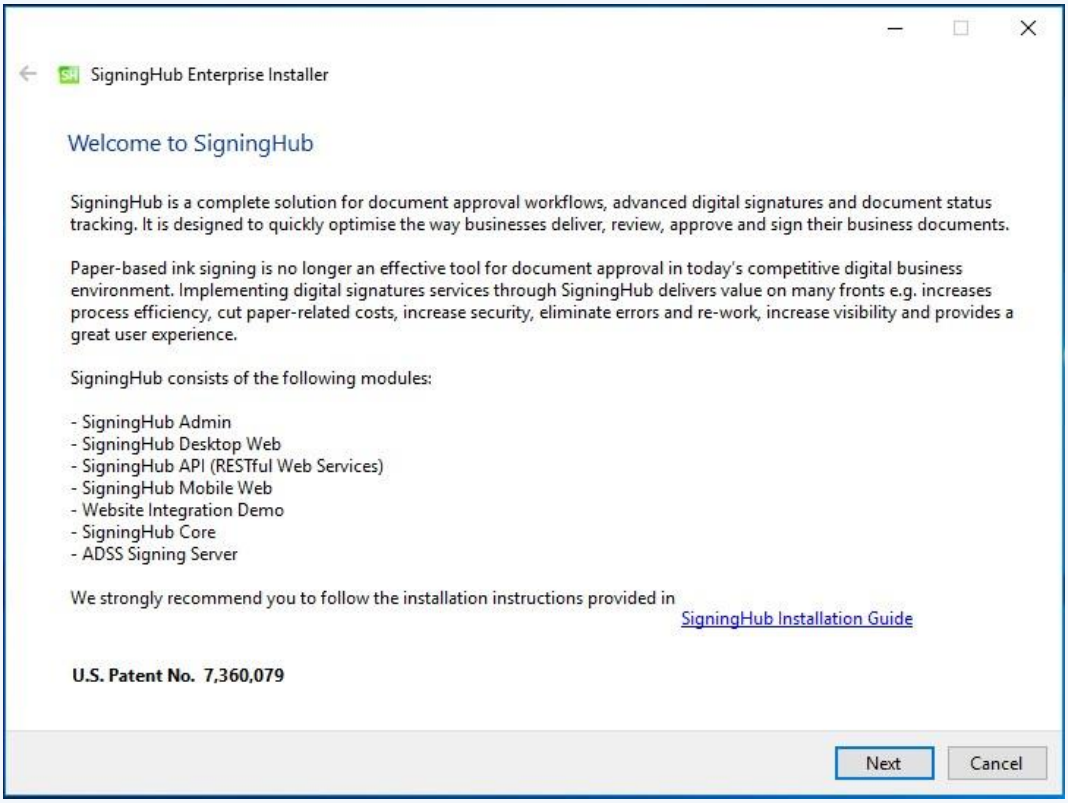
SigningHub Enterprise installer generates all the required database tables and

populates the default data required to run the system. Therefore, there is no requirement for separate SQL scripts or equivalent for non-SQL databases.

Note there is a 'back' arrow towards the top left of the installer dialogue window. Use this button if you wish to return to previous screens and modify your input values/ configuration choices.

Once the database is created, launch the installer by right-clicking the file **"[SigningHub installation directory] /setup/install.bat**" and selecting **Run as administrator** from the menu.

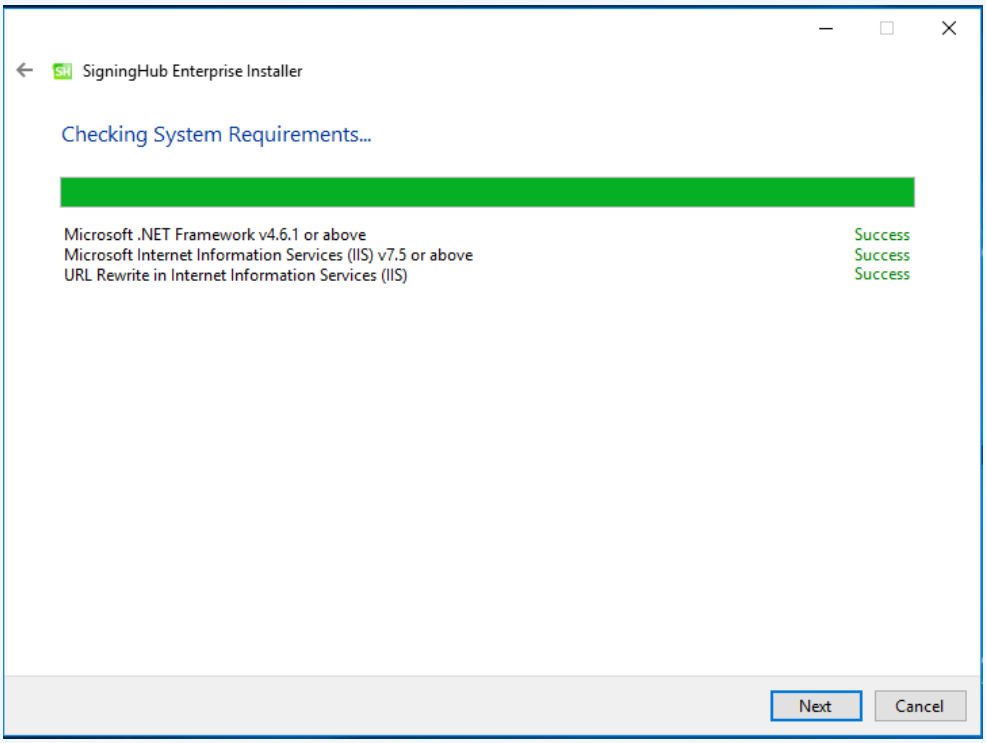
The following welcome screen is shown:



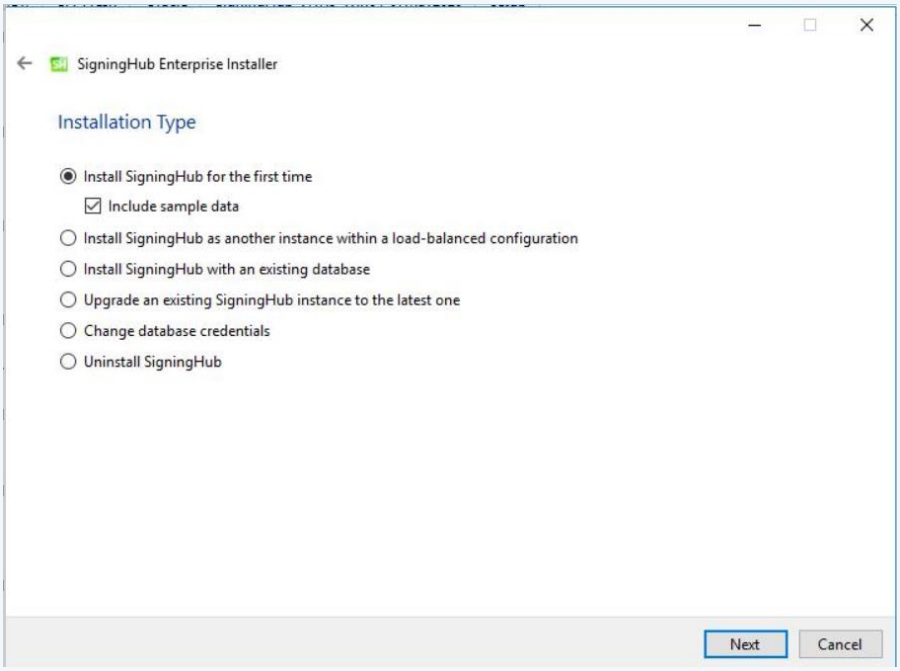
Click the **Next** button to continue.

A check of various operating system requirements is performed, i.e. for those feature dependencies detailed previously. If any of the SigningHub Enterprise system dependencies is not found, or not functioning, then this will be reported on the next screen.

Note only proceed with the installation once all issues related to system dependencies are resolved as shown here:



Click the **Next** button to select an installation type:



If you are installing SigningHub Enterprise for the first time or you wish to deploy a fresh installation with a new database, then select **"Install SigningHub for the first time"**.

If you do not select **Include sample data** while installing a fresh instance, then it will not create profiles with sample data (i.e. authentication, certification, and verification

profiles etc). However, Default ADSS and SMTP connectors will always be configured with sample data. On selection of this option, all the profiles will be created with default sample data.

The **"Install SigningHub as another instance within a load-balanced configuration"** option will install the SigningHub Enterprise instance in a load-balanced mode.

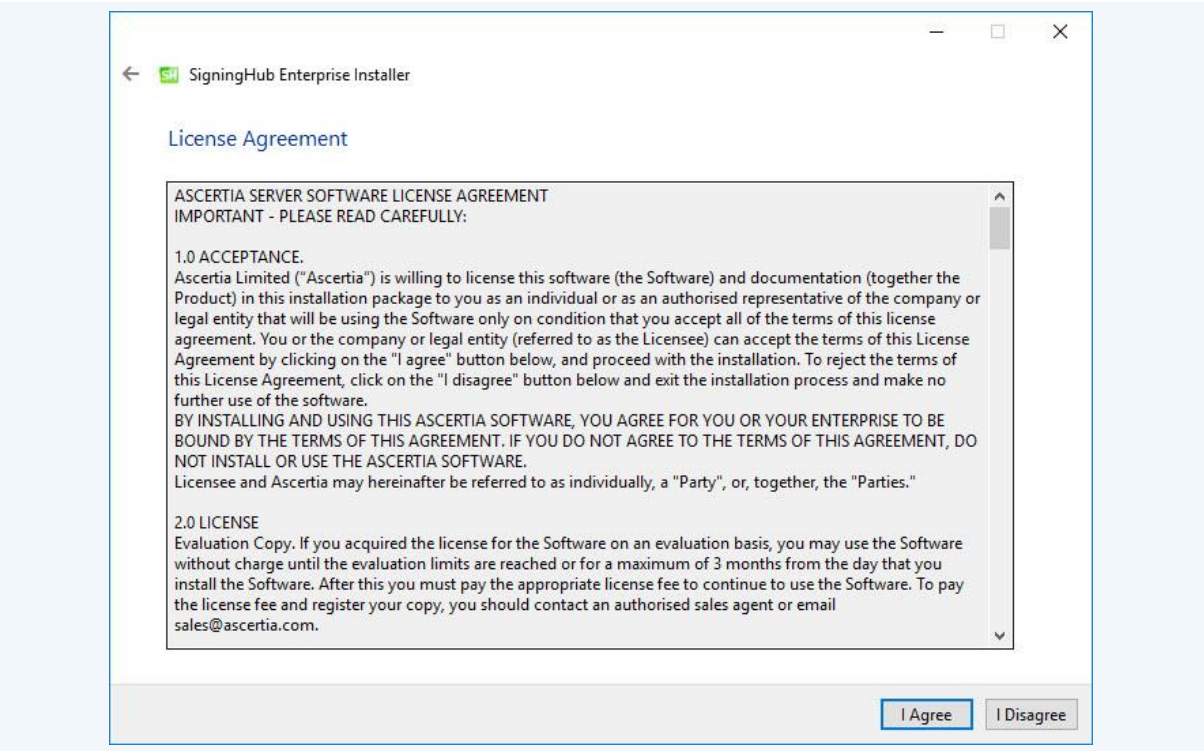
If you wish to upgrade an older system to the latest version, then select **"Upgrade an existing SigningHub instance to the latest one"**. Installer supports the upgrade when the base (current) installation is v6.2 or higher.

To upgrade to version 6.2 from version 5.x contact support@ascertia.com. The **"Install SigningHub with an existing database"** option will install SigningHub Enterprise against an existing SigningHub Enterprise database. For example, this option can be used to recover a system from a database back-up.

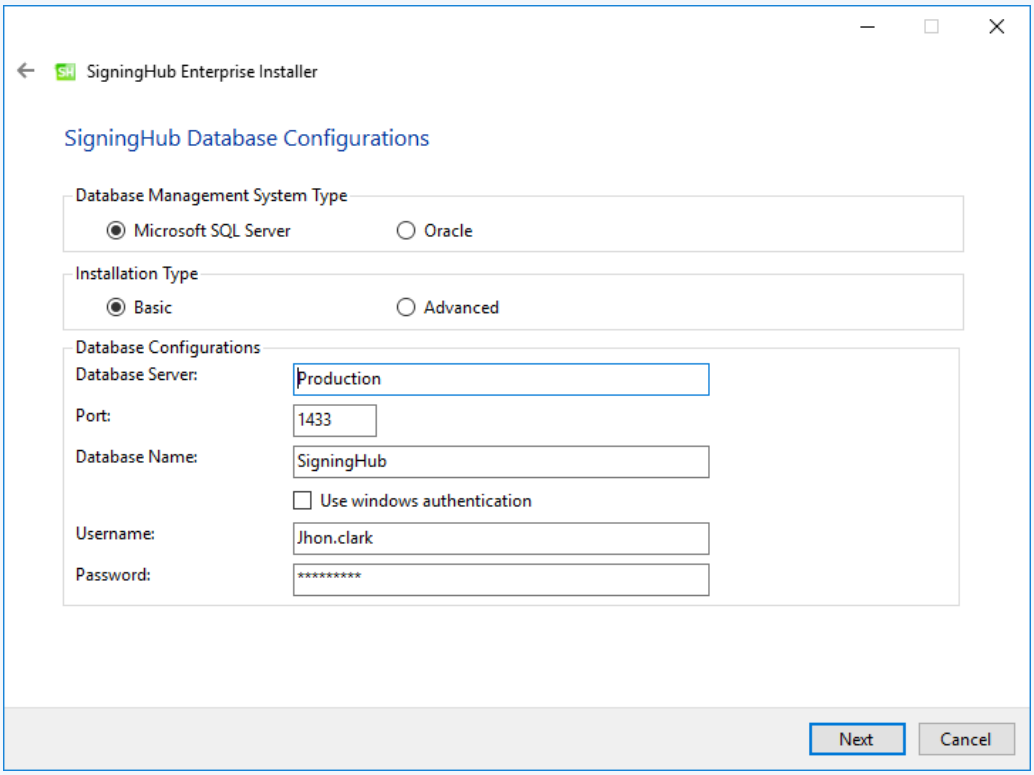
The "**Change database credentials**" option is used if the database password, user, database name and/or server is changed, and it needs to be updated in SigningHub installation.

Select the last option "**Uninstall SigningHub**" if you wish to uninstall SigningHub Enterprise from the system.

Click the **Next** button to show the License Agreement:



Click the **I Agree** button to proceed. The following screen for database details is displayed:



Username and Password must not contain the following characters: [] () , ; ? * ! @ =. These characters are used to either initialize a connection or separate connection values.

On the SigningHub Database Configurations screen you can either choose a **"Microsoft SQL Server"** or **"Oracle"** Database Management System.

Further you can either choose to do a basic installation or use an advanced one. If this is a basic installation, then use the first option "**Basic**" and provide the appropriate SigningHub database credentials.

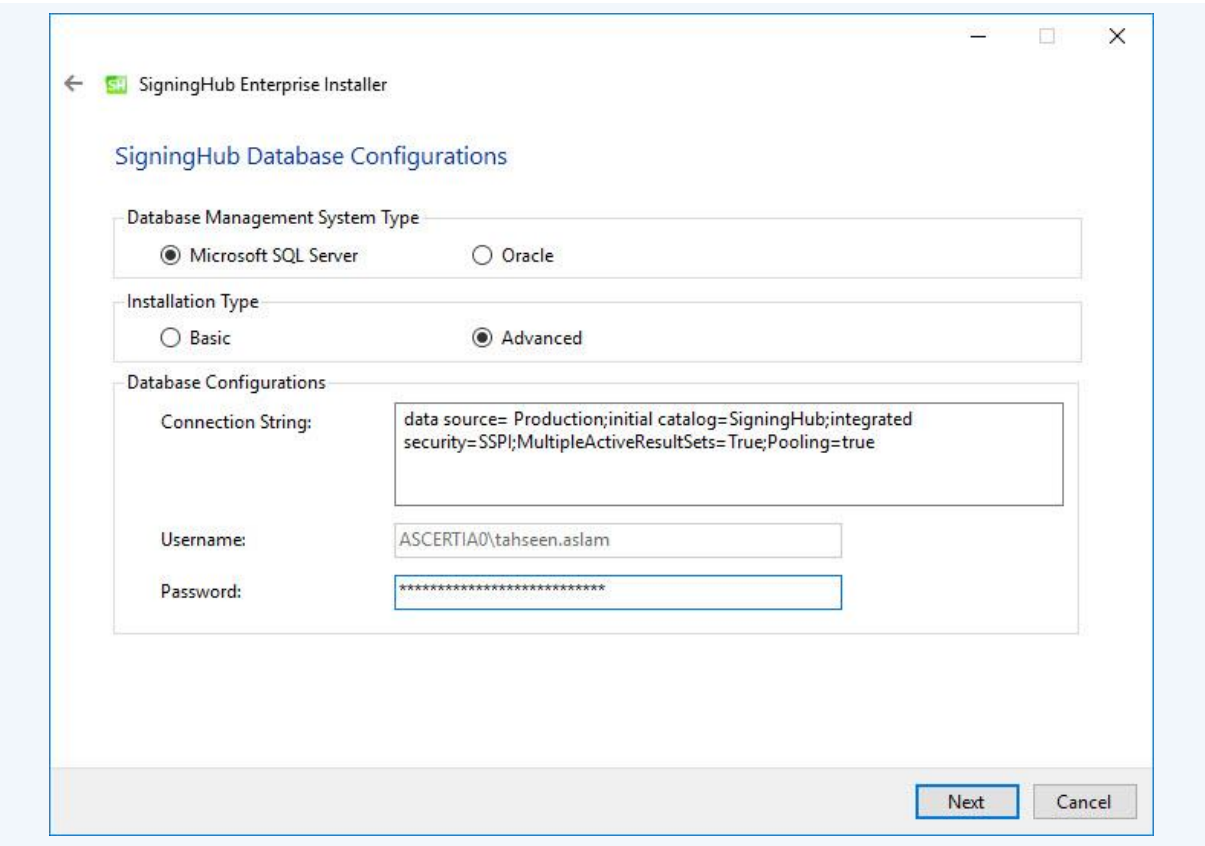
The information displayed above is an example and you should configure the relevant settings for your own environment.

Note that once you complete the options and select **Next,** the installer uses the information provided to test the connectivity to the database. If the installer can establish the connection with the database, then it will proceed with the installation.

The following table details the configuration options:

Item	Description
Database Server / Host Name	Database server IP or DNS name.
Port	Database listening port. For SQL Server the default port is 1433 and for Oracle the default port is 1521.
Database Name / Service Name/SID	Name of the database instance. Note this must exist prior to the installation. Provide Service Name/SID for Oracle database management system.
Use Windows Authentication	If enabled, installer will use the Windows logged in user to communicate with database. You are required to enter password because it will be used in Application Pool to set the Identity against this user for all websites. By default, the current logged in user will be configured in the Application Pool Identity. If you wish to run SigningHub Enterprise under a different Windows user, then you need to change it manually. If your requirement is to use SQL Server authentication, then type SQL Server user name and password in the underneath fields without enabling this option. Windows authentication is not supported for Oracle Database Management System.
Username	Name of the database user. Note this must exist prior to the installation. It is not required in the case of Windows Authentication.
Password	Password credential of the database user. Note this must exist prior to the installation. In case of Windows Authentication, type the password of domain user shown in the Username field to configure the Application Pool Identity in IIS Server for successful communication with SQL Server.

If this is not a basic installation and you choose the second option to "**Advanced**" then the following screen is shown:



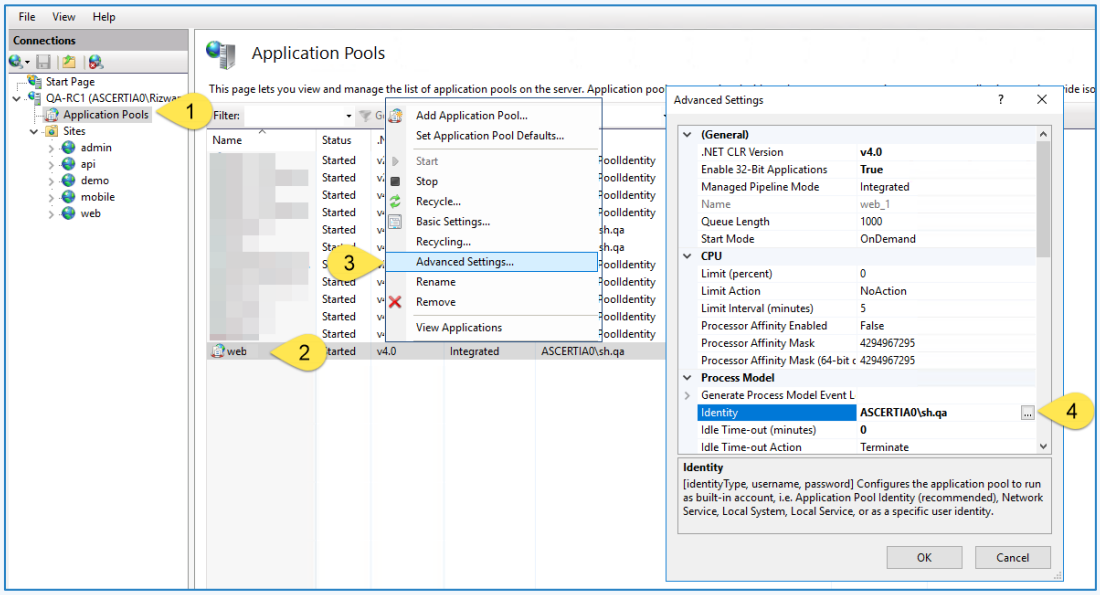
Username and Password must not contain the following characters: [] () , ; ? * ! @ =. These characters are used to either initialize a connection or separate connection values.

The information displayed above is an example and you should configure the relevant settings for your own environment.

Once you complete the options and select **Next,** the installer uses the information provided to test the connectivity to the database. If the installer can establish the connection with the database, then it will proceed with the installation.

If windows authentication is enabled in connection string, installer will use the Windows logged in user to communicate with database upon clicking the Next button. You are required to enter password because it will be used in Application Pool to set the Identity against this user for all websites.

By default, the current logged in user will be configured in the Application Pool Identity. If you wish to run SigningHub Enterprise under a different Windows user, then you need to change it manually. As shown in the following Screen:



The following table details the configuration options:

Item	Description

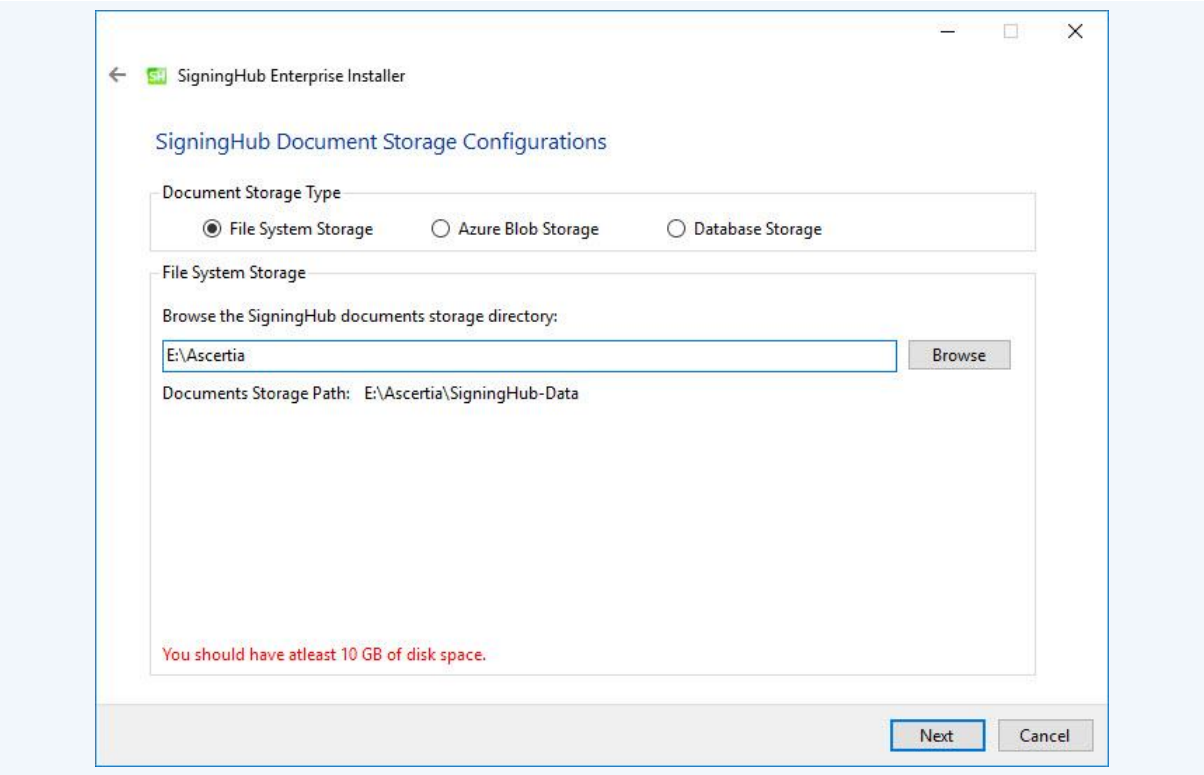
SigningHub Connection String	The following are sample connection strings for SQL Server:
	Simple One
	`"data source= [Database Server Address];initial catalog= [Database Name];user id=[Database User Name];password=[Database User Password];MultipleActiveResultSets=True;Pooling=true"`
	For Named instance
	`"data source= [Database Server Address] \SQL2012std;initial catalog=[Database Name];user id=[Database User Name];password[Database User Password];MultipleActiveResultSets=True;Pooling=true"`
	For Windows Authentication
	`"data source= [Database Server Address];initial catalog=[Database Name];integrated security=SSPI;MultipleActiveResultSets=True;Pooling=true"`
	The following are sample connection strings for Oracle Server:

	Simple One "Data Source=(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = **[Database Server Address]**)(PORT = 1521))(CONNECT_DATA =(SERVER = DEDICATED)(SERVICE_NAME = **[Service Name/SID]**)));User ID**==[Database User Name]**;Password==**[Database User Password]**;Pooling=true;Min Pool Size=0;Max Pool Size=100;Connection Lifetime=0"
Username	Field will only be shown in case of Windows Authentication while for SQL Server Authentication/Oracle, username will be provided in the connection string.
Password	In case of Windows Authentication, type the password of domain user shown in the Username field to configure the Application Pool Identity in IIS Server for successful communication with SQL Server. In case of SQL Server authentication/Oracle, password will be provided in the connection string.

Click the **Next** button to select the SigningHub data storage directory:

On the SigningHub Document Storage Configurations screen you can either choose a "**File System Storage"** or **"Azure Blob Storage"** or "**Database Storage"**.

If the Document Storage is either on local file system or on the local network path, then select the option **"File System Storage"**.



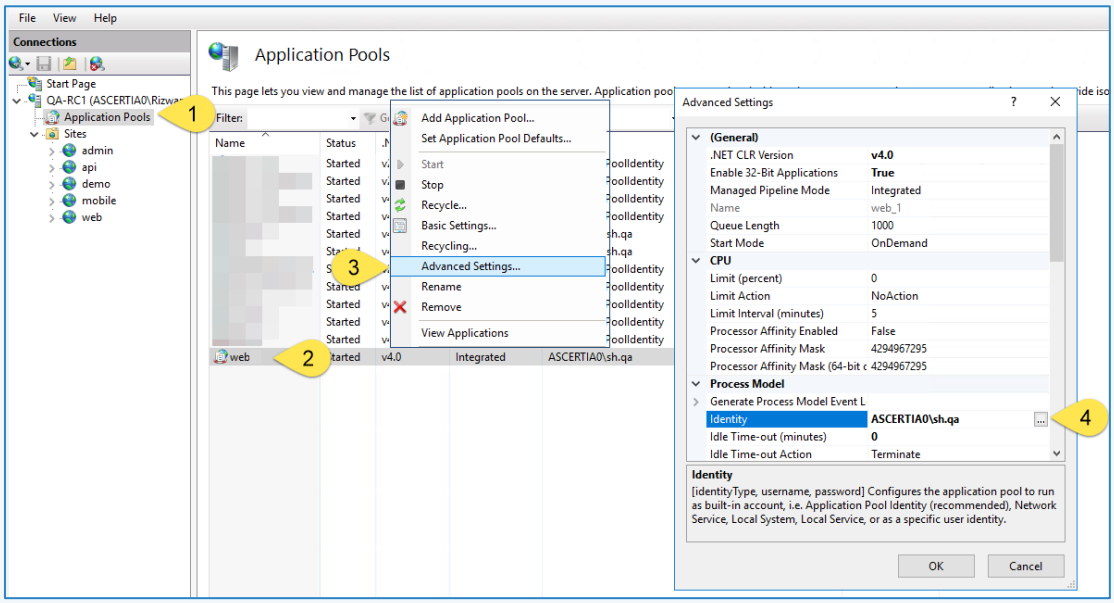
The information displayed above is an example and you should configure the relevant settings for your own environment.

Click **Browse** and specify a storage path to store the SigningHub data.

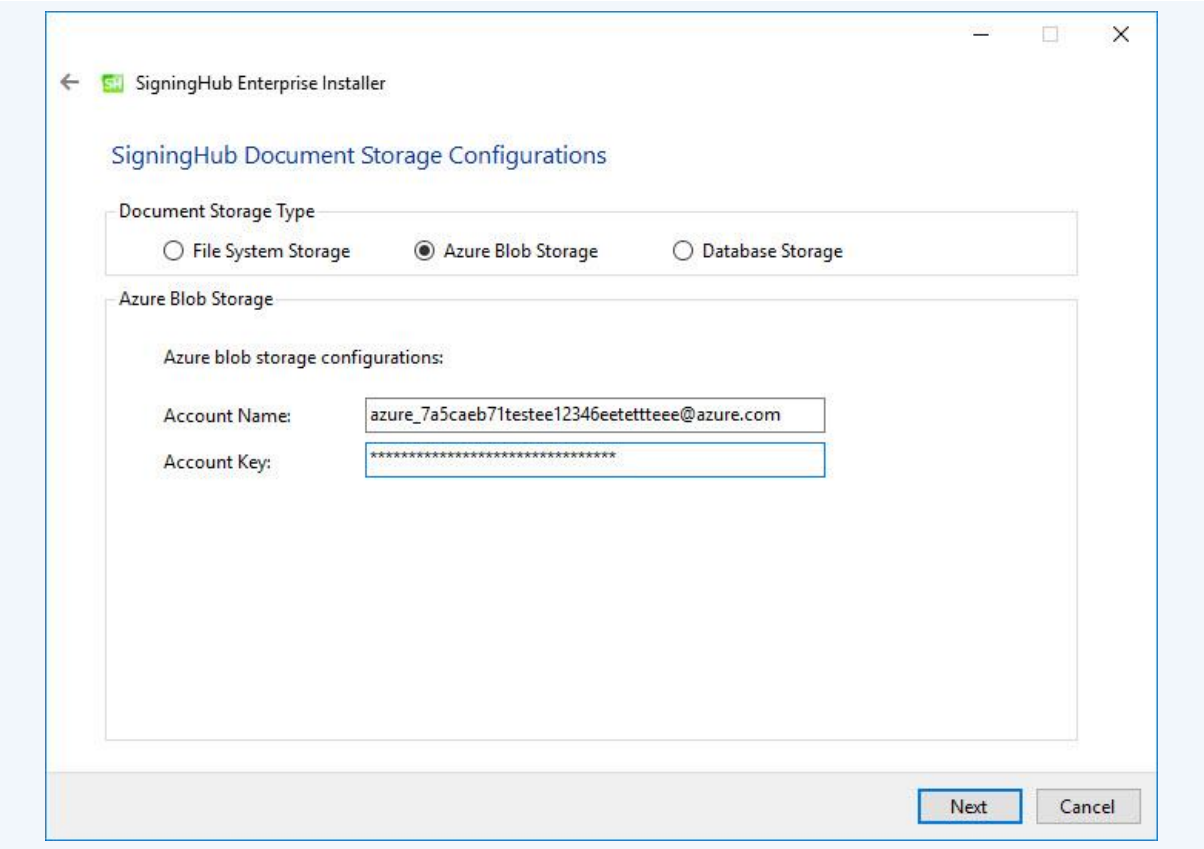
Document Storage path can be a local drive, a network drive or an Azure blob. If the path is on a local drive, then the installer will automatically assign the read/write permissions to the "IIS_IUSRS" user group.

If the path is on a network/Azure drive, then the permissions should be assigned manually to a user before continuing the installation process. To add the permissions on a network drive, follow these instructions:

1) Create a domain/Azure user with read/write permissions.
2) Add the read/write permissions on the directory [Document Storage Path] and complete the installation process.
3) Now go to IIS Manager and add the user created in step 1 in Application Pool against all SigningHub websites one by one as shown below (Skip this step if SigningHub Enterprise is installed by using Windows Authentication):



If this is not a File System Storage and you choose the second option to **"Azure Blob Storage"** then the following screen is shown:

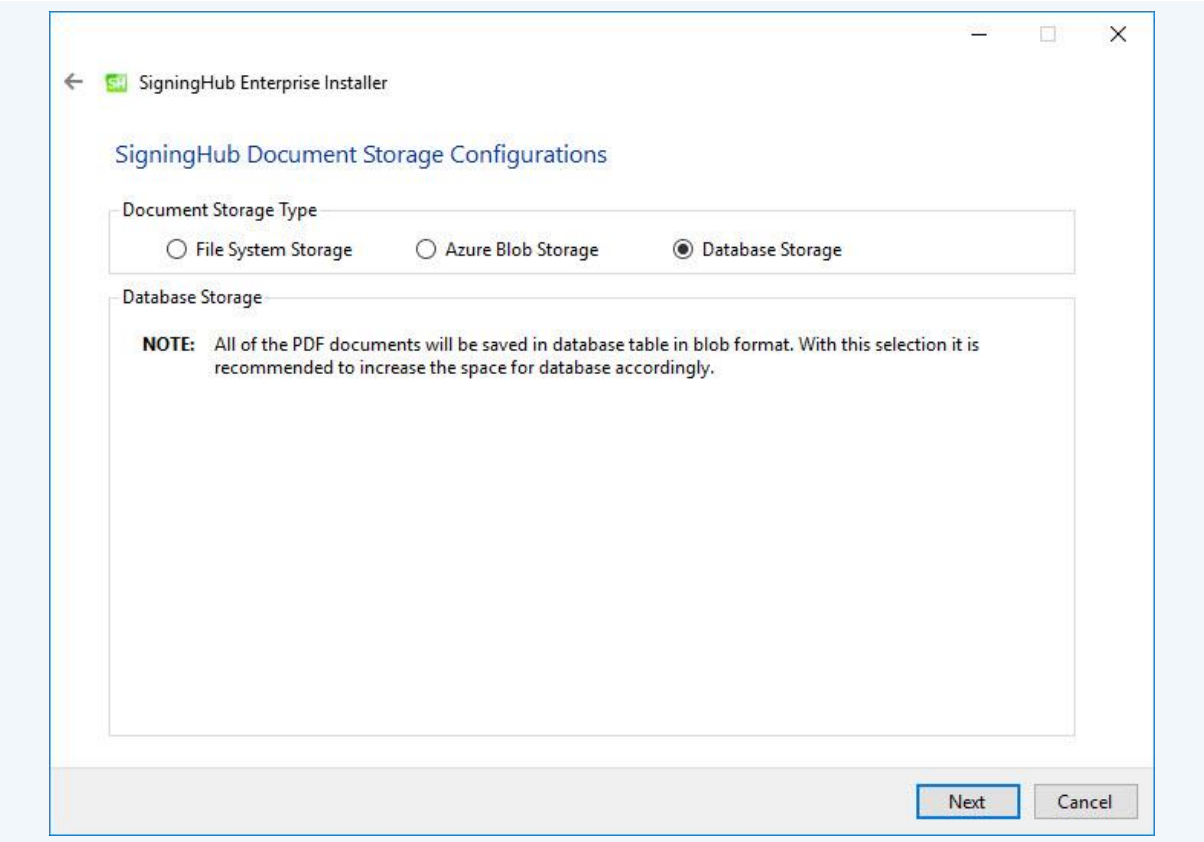


The information displayed above is an example and you should configure the relevant settings for your own environment.

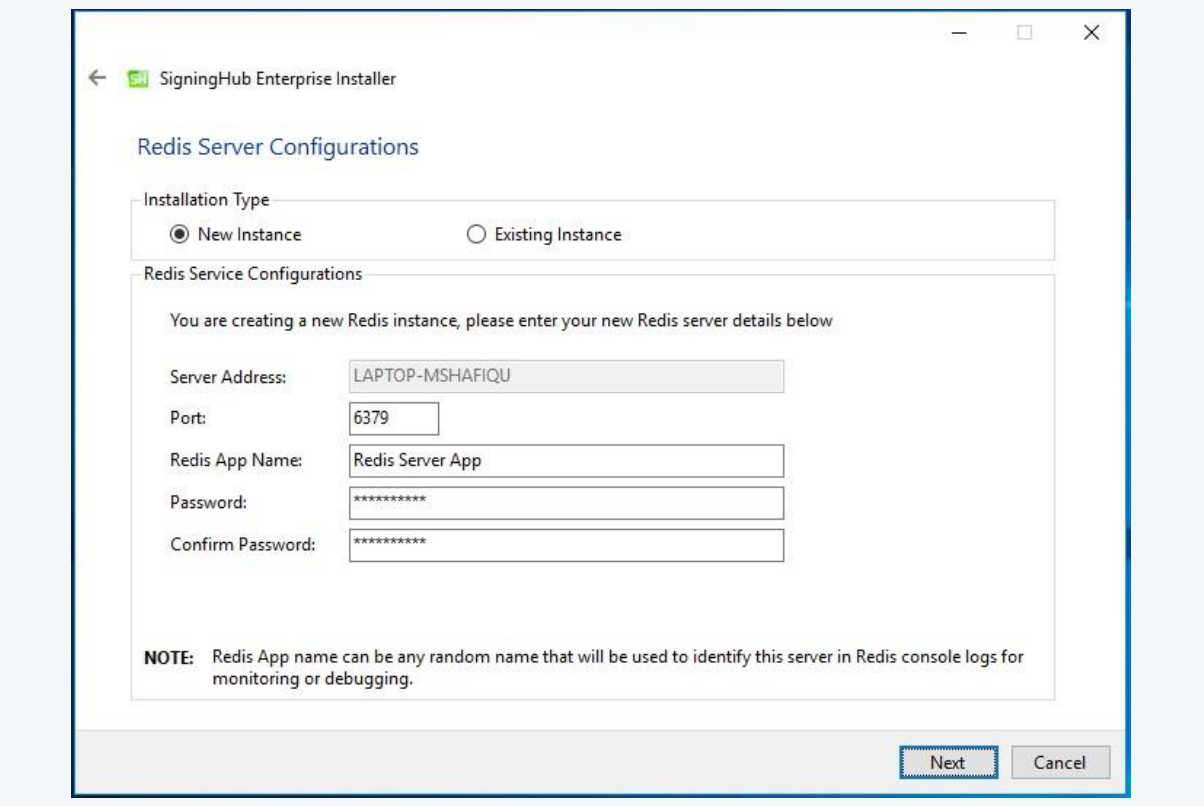
The following table details the configuration options:

Item	Description
Account Name	Account Name of the Azure. Note this must exist prior to the installation.
Account Key	Account Key of the Azure. Note this must exist prior to the installation.

If this is not a File System Storage and you choose the third option to **"Database Storage"** then the following screen is shown:



Click the **Next** button to proceed. The following screen for Redis Server will appear:



Redis is a light weight server, which works as backplane and message broker for SigningHub application over an HTTP/s port. It is bundled within the SigningHub package, and is used for message popup notifications in SigningHub when a user is already logged in to SigningHub.

1) Redis is a mandatory module of SigningHub. If you do not already have a Redis instance, please choose “New Instance” and configure an App Name of your choice along with a password. SigningHub will define the port automatically when using the “New Instance” option.

2) If by any chance you have Redis server installed or you want to use Redis server from Azure or Amazon, you need to know the app name, password and port to connect to that instance. In that case, select Existing Instance in the above screen..

By default, SigningHub Installer installs Redis 3.0. You must upgrade Redis to the latest version, which is more secure. For detailed steps on how to install latest Redis server, see Appendix H – Installing Redis Server.

On the Redis Server Configurations screen you can either choose a **“New Instance”** or **“Existing Instance”** option.

If this is a new instance, then use the first option i.e. “**New Instance**” and provide the appropriate Redis server configurations.

The information displayed above is an example and you should configure the relevant settings for your own environment.

The following table details the configuration options:

Item	Description
Server Address	Specify the Redis server address. This server is used to send real-time on-screen notifications for document sharing.
Port	Specify the service port for the Redis server.
Redis App Name	Specify the name of Redis App. This can be any random name that will be used to identify this server in Redis console logs for monitoring or debugging.
Password	Specify the password to authenticate the Redis server.
Confirm Password	Specify the same password again as provided in the above password field to confirm it.

Redis can enforce password-based security to save or read the key value pairs from the Redis server. To enable password-based security, follow these instructions:

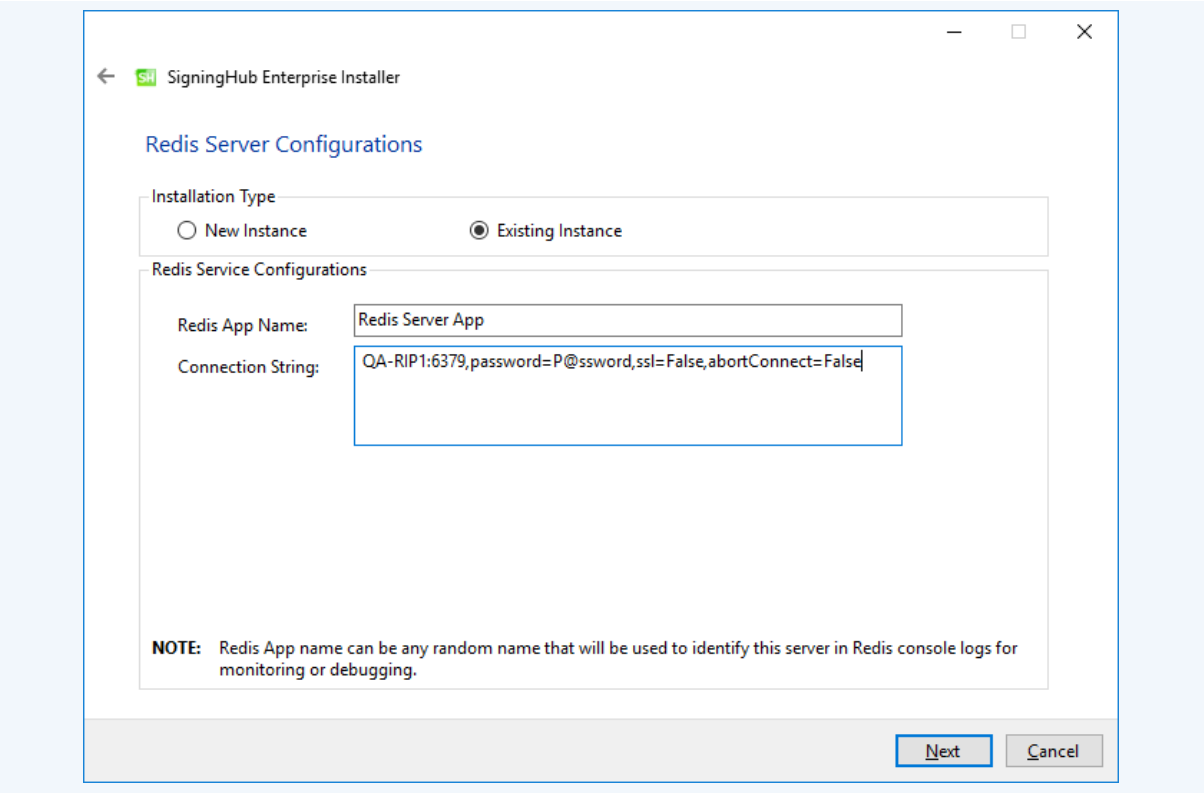
1) Go to **[SigningHub Installation Directory]/Redis**
3) Run the Redis command line interface by click on "**redis-cli**" application in administrator mode
4) Run the command "**CONFIG SET requirepass "[password]"**"
5) Sign into **SigningHub Administrator** account
6) Go to **Configurations>Redis** and change the password in Redis Server Connection String
7) Update the settings and **Restart IIS**

Redis can disable asking for password for saving and reading the key value pairs from the Redis server. To turn off the password, follow these instructions:

1) Go to **[SigningHub Installation Director]/Redis**
8) Run the Redis command line interface by click on "**redis-cli**" application in administrator mode
9) Run the command **"CONFIG SET requirepass "" "**
10) Sign into SigningHub Administartor account
11) Go to Configurations>Redis and change the password as empty in Redis Server Connection String
12) Update the settings and **Restart IIS**

For Load balanced deployments, only one instance of Redis is needed for SigningHub to work with. Rest of the instances of SignignHub will communicate with Redis using HTTP/s address and Port configured in SigningHub Admin.

If this is not a new instance, and you are choosing the second option i.e. "**Existing Instance"** then the following screen will appear:

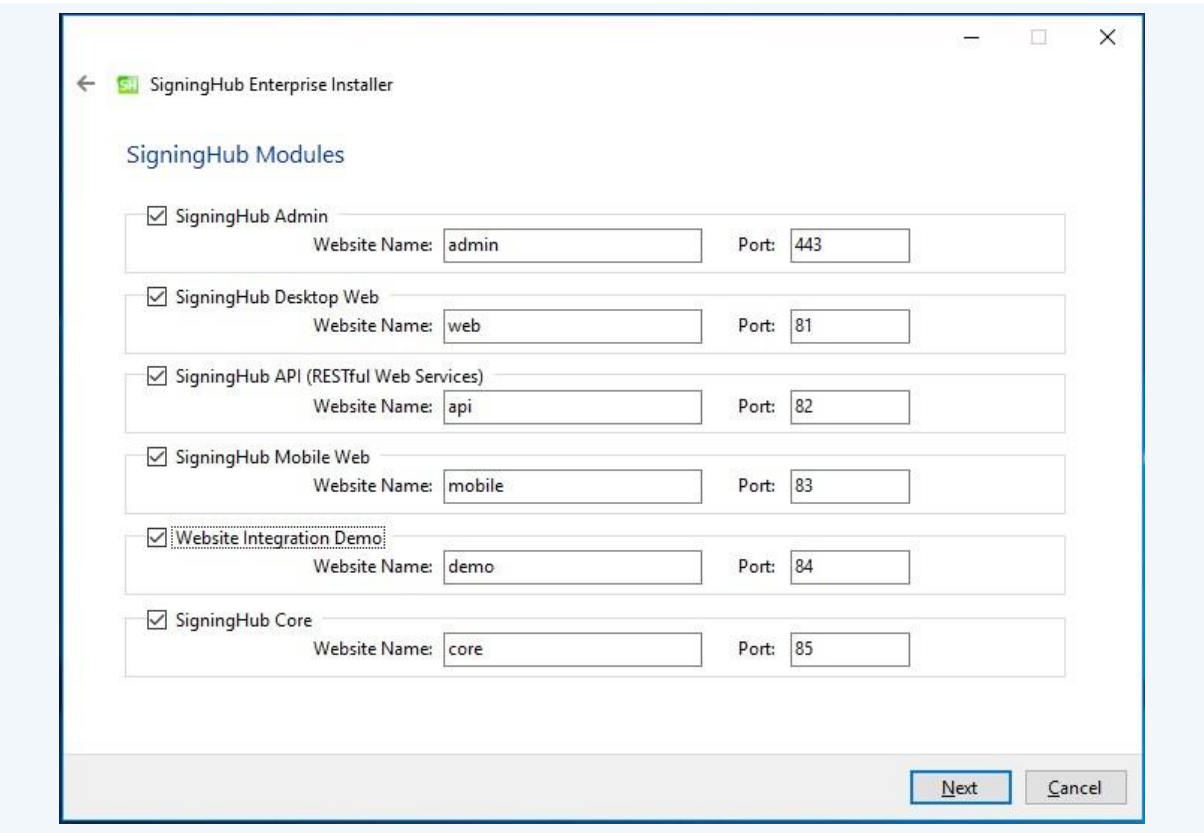


The information displayed above is an example and you should configure the relevant settings for your own environment.

The following table details the configuration options:

Item	Description
Redis App Name	Specify the name of Redis App. This can be any random name that will be used to identify this server in Redis console logs for monitoring or debugging.
Connection String	The following is a sample connection string for a Redis Server: `"[Redis Server Address]: [port], password=[Redis Server Password],ssl=False,abortConnect=False"`

Click the **Next** button to select specific modules: -



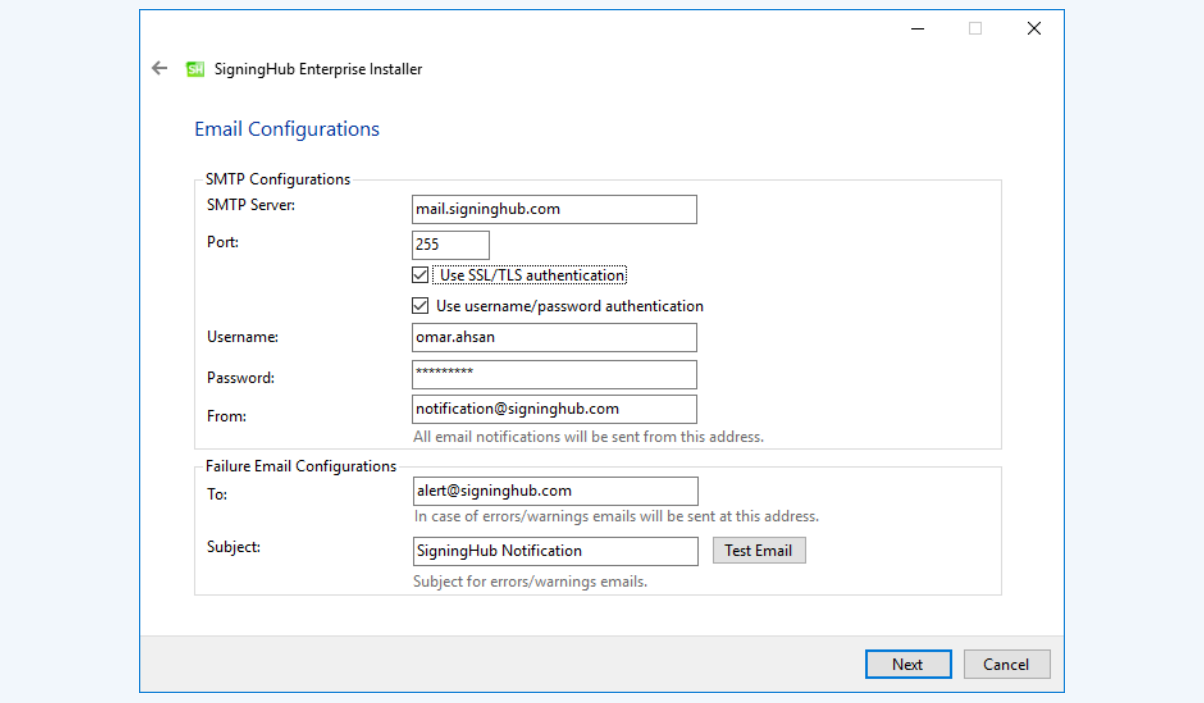
Select the appropriate modules to install the required features. For each selected application, provide the web application name and port. A typical in-house installation of SigningHub Enterprise should only include Admin, Desktop Web and Mobile Web. Where tight integration through Web Services API is required, then the API Web Services option must also be installed.

The information displayed above is an example, which you may change to suit your environment and organisation preferences. However, the example shown is sufficient. The names will appear as websites under IIS.

The following table details the modules options:

Item	Description
SigningHub Enterprise Admin	SigningHub Enterprise Admin console is used by the administrators to manage the system wide configurations, service plans, user accounts, billing, access control etc.
SigningHub Enterprise Desktop Web	SigningHub Enterprise Desktop Web is used to create workflows, share documents, create digital/electronic signatures etc.
SigningHub Enterprise API	REST API provides the functionality to communicate with SigningHub Enterprise server to create workflows, upload documents, apply templates, share documents etc. from business applications.
SigningHub Mobile Web	Web application for mobile browsers; it provides the client-facing functionality for document workflow approval/sign-off and user account management.
SigningHub Enterprise Website Integration Demo	A demo application to illustrate SigningHub Enterprise and business application integration.
SigningHub Enterprise Core	SigningHub Enterprise Core is used to manage backend processes, e.g. send emails, delete documents, auto reminders for pending documents and many others. This is a mandatory module for SigningHub Enterprise.

Click the **Next** button to configure the SMTP server and email settings:



Configure the SMTP Server and email settings for your environment. SigningHub Enterprise must have access to a suitable SMTP server. Without which users will not be able to receive registration emails that are required to complete the sign-up process. In addition, administration notification and alert emails will also not be sent. Although the latter will not prevent functionality, but it is not a recommended approach.

The information displayed above is an example and you should configure the relevant settings for your own environment.

The configuration items are explained in the following table:

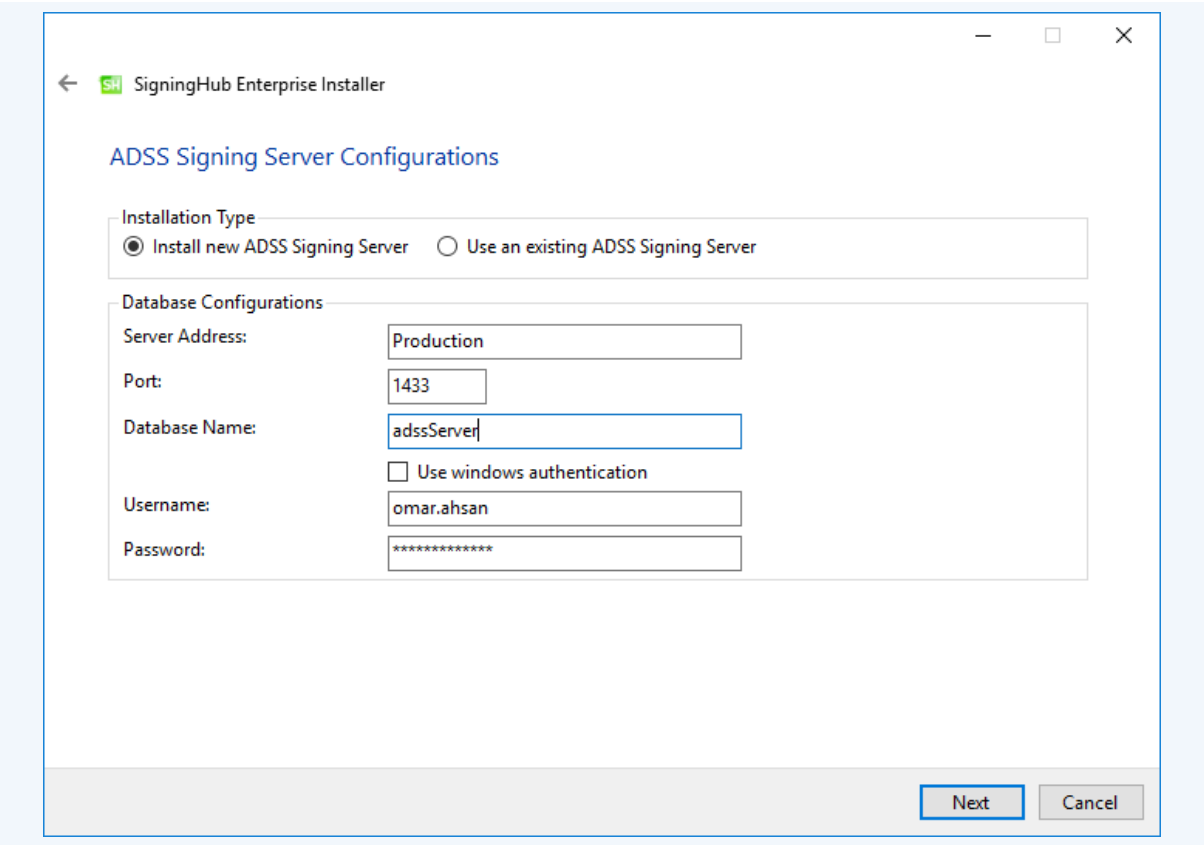
Item	Description
SMTP Server	Defines the email server address. This email server is used to send email notifications to users as required, such as for account registration, document sharing, and workflow completion. It is also used for sending notification emails to SigningHub Enterprise administrators.
Port	Define the service port for the SMTP mail server.
Use SSL/TLS authentication	Select this option if the SMTP mail server requires SSL/TLS.
Username	Configure the SMTP mail server username that is used to send SigningHub Enterprise generated emails.
Password	Define the password to authenticate the SMTP server.
From	Configure the “From” email address that should be used to send notification emails to users and administrators.
To	Configure the email address where error notifications should be sent. This is usually the IT support team address.
Email Subject	Define a subject line for the notification emails that are sent to the administrator, e.g. SigningHub Enterprise Issue Alert.

After configuring these SMTP settings, click the Test Email button to verify that SMTP configurations are valid.

Click the **Next** button to proceed.

Now its time to install the ADSS Signing Server engine that powers SigningHub Enterprise.

The following screen is shown:



On the ADSS Signing Server screen you can either choose to do a fresh installation of the ADSS Signing Server along with SigningHub Enterprise installation or use an existing ADSS Signing Server installation.

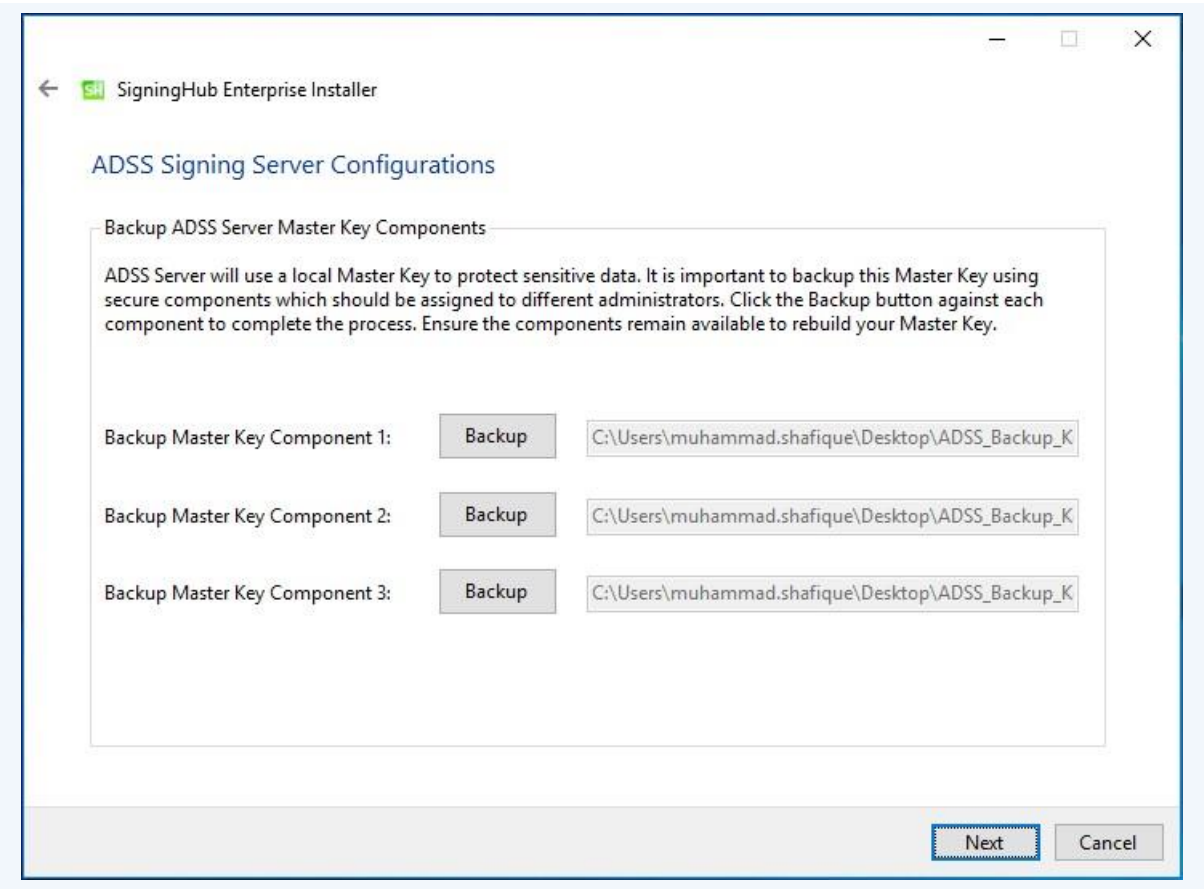
If this is a fresh installation of SigningHub Enterprise, then use the first option "**Install New ADSS Signing Server**" and provide the appropriate ADSS Signing Server database credentials.

The information displayed above is an example and you should configure the relevant settings for your own environment.

This table details the options:

Item	Description
Database Server	Database server IP address or DNS name.
Port	Database listening port. For SQL Server the default port is 1433 and for Oracle the default port is 1521.
Database Name/ Service Name/SID	Name of the database instance. Note this must exist prior to the installation. Provide Service Name/SID for Oracle database management system.
Use windows authentication	If enabled, the installer will use the Windows logged in user to communicate with database. If your requirement is to use SQL Server authentication, then type SQL Server user name and password in the underneath fields without enabling this option. Windows authentication is not supported for Oracle Database Management System.
Username	Name of the database user. Note this must exist prior to the installation.
Password	Password credential of the database user. Note this must exist prior to the installation.

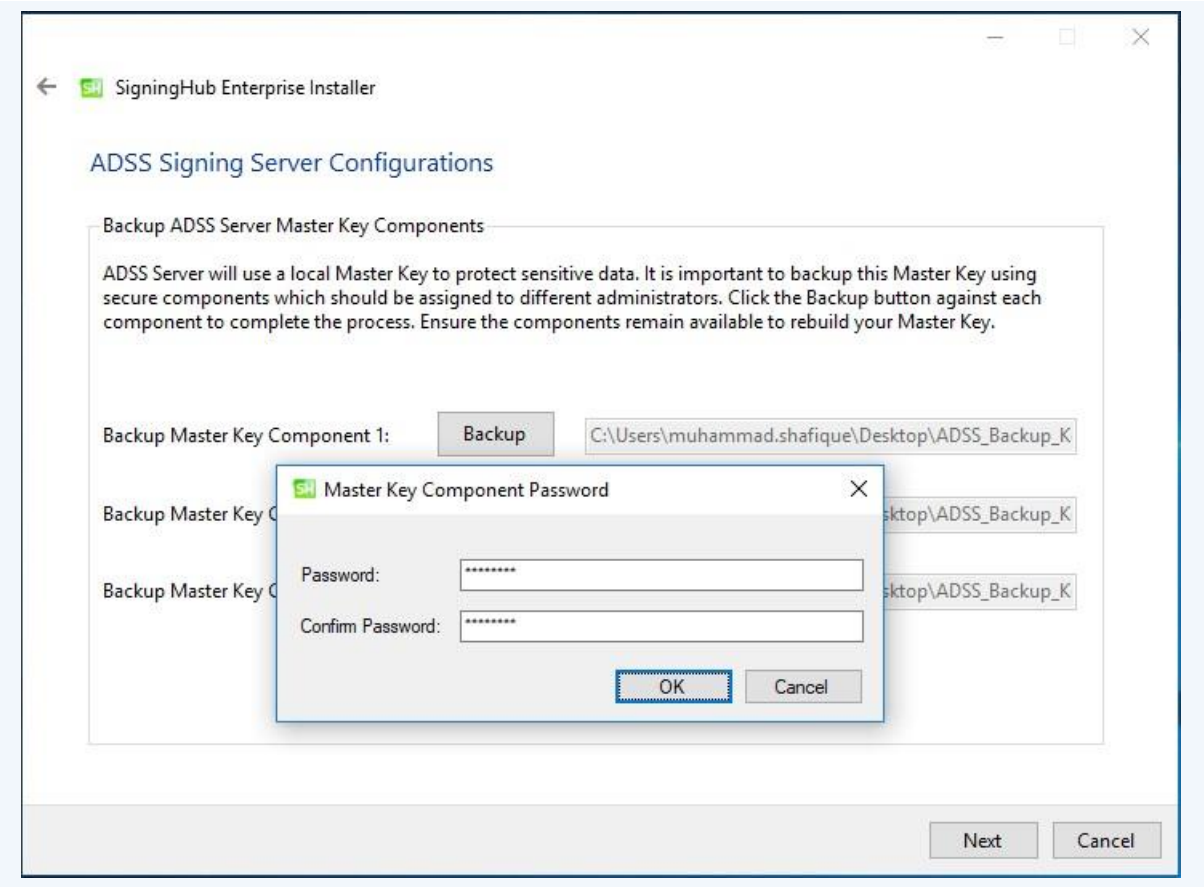
When you select the "**Install New ADSS Signing Server**" option, then from the next screen you need to generate a Master Key to encrypt the database data and take a backup of the Master Key in the form of three components. Use the **Backup** buttons one by one to take the backup of each Master key component. The installer will prompt to provide a password for each Master Key component and encrypt it with the provided password before saving on the disk:



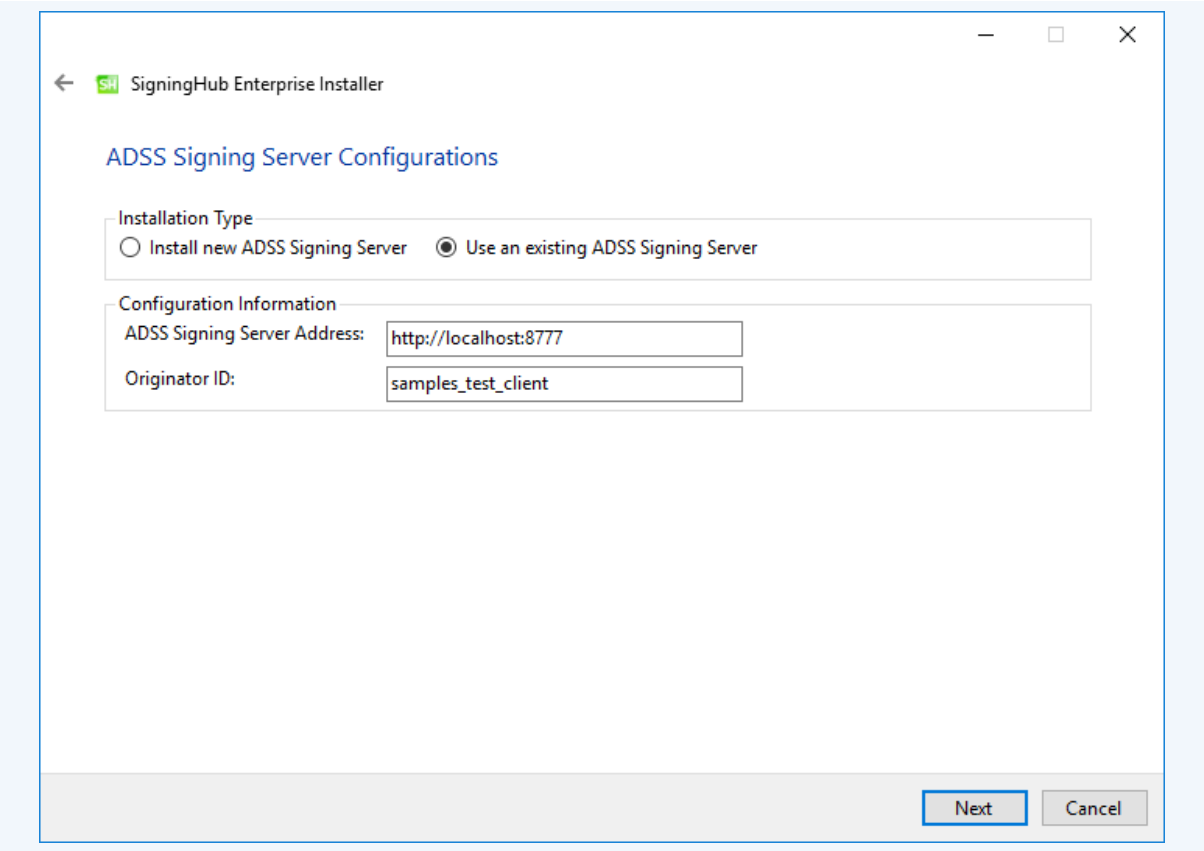
The following dialog will appear to input password and confirm password for backup keys:

Keep all the Master key components secure and remember their passwords.

For any reason, if you lose these keys/ passwords then you cannot upgrade this ADSS Server to the next versions and even Ascertia cannot help you to recover these keys.



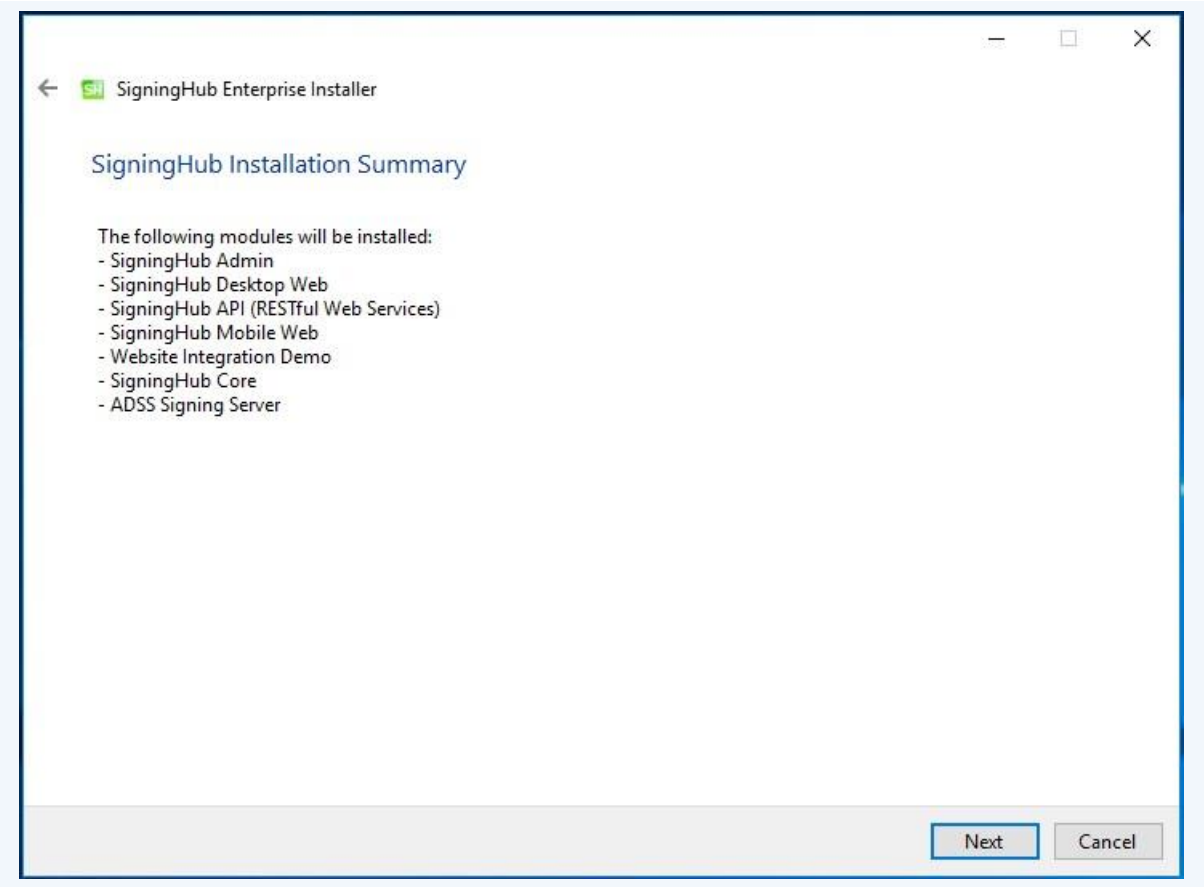
If this is not a fresh installation and you choose the second option to "**Use an existing ADSS Signing Server**" then the following screen is shown:



Configure the **ADSS Signing Server Address** and **Originator ID**. The ADSS Signing Server administrator will be able to provide this information. The Originator ID shown above is the default one when installing ADSS Signing Server with sample data. Ascertia recommends configuring a dedicated Originator ID for the use of SigningHub Enterprise.

Every client of ADSS Signing Server requires an Originator ID to identify itself, click here to get details as how to obtain it from ADSS Signing Server. This is like well-known APIs keys of Google, etc. Note this Originator ID will be created automatically if you chose to install a fresh ADSS Signing Server via the installer.

Click the **Next** button to see the summary and complete the installation:

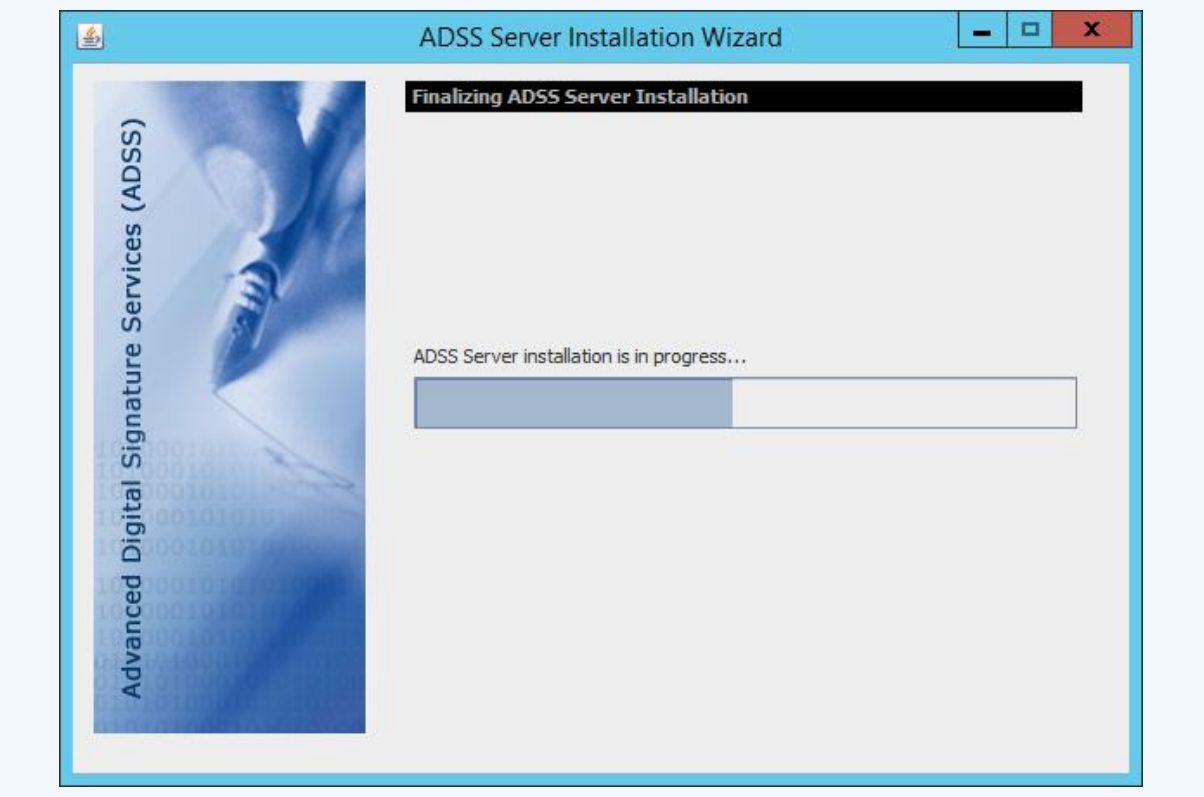


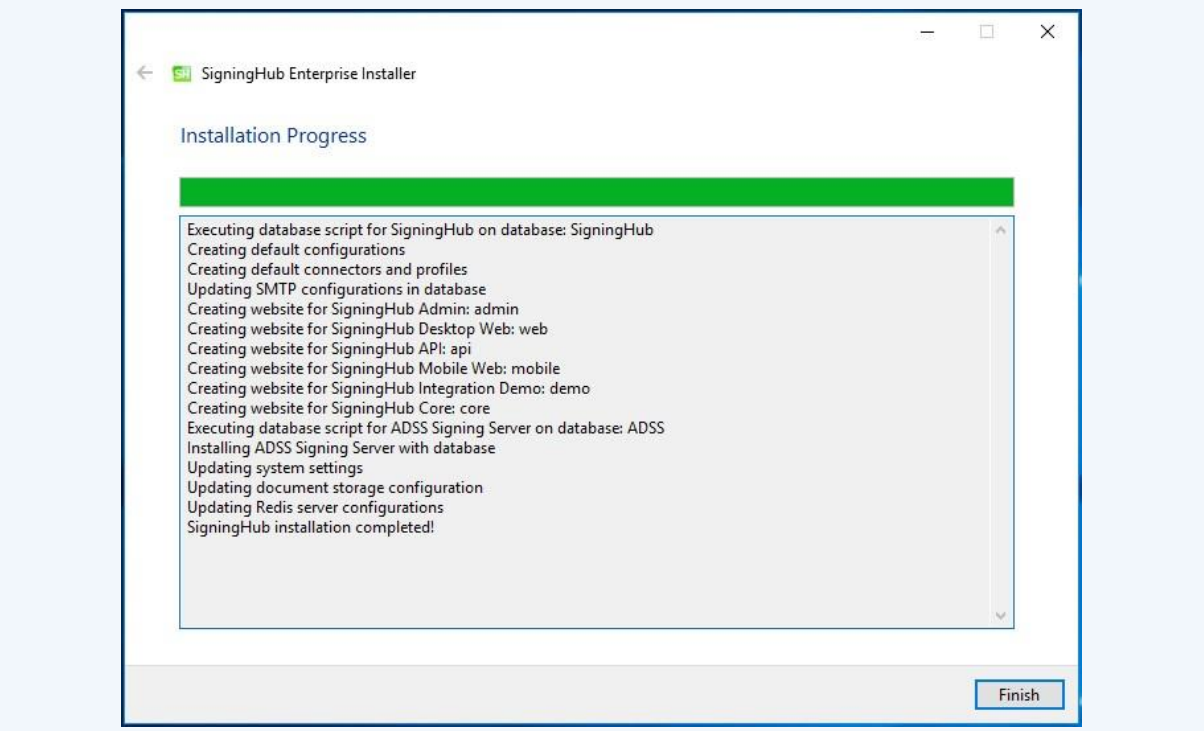
This screen shows the installation summary by listing the different product modules that will be installed.

If you think any listed item is incorrect then use the **Back** button (arrow towards the top-left of the dialogue box) to correct your choices before proceeding ahead.

Otherwise, click the **Next** button to continue with the installation.

The following screen may also occur to show the ADSS Signing Server installation progress if its a fresh installation:





Click **Finish** to complete the installation process.

Please note that Local Signing will not be available by using sessionState mode "SQLServer". However, if users still require performing Local Signing, then please follow these steps to replace sessionState mode with" InProc":

1. *Open the **[SigningHub-Installation-Dir]/web/web.config** file of all deployments of SigningHub:*
2. *Replace:*

"<sessionState mode="SQLServer" allowCustomSqlDatabase="true" sqlConnectionString="AdocsEntities" cookieName="SH_ID" timeout="60" compressionEnabled="true">
</sessionState>"

3. *With:*

"<sessionState mode="InProc" timeout="60" cookieName="SH_ID" />"

You can configure an AJP Connector as mentioned in Appendix A

Please note that Local Signing will not be available by using sessionState mode "SQLServer". However, if users still require performing Local Signing, then please follow these steps to replace sessionState mode with" InProc":

1) OPEN the **[SigningHub-Installation-Dir]/web/web.config** file of all deployments of SigningHub:

13) Replace:

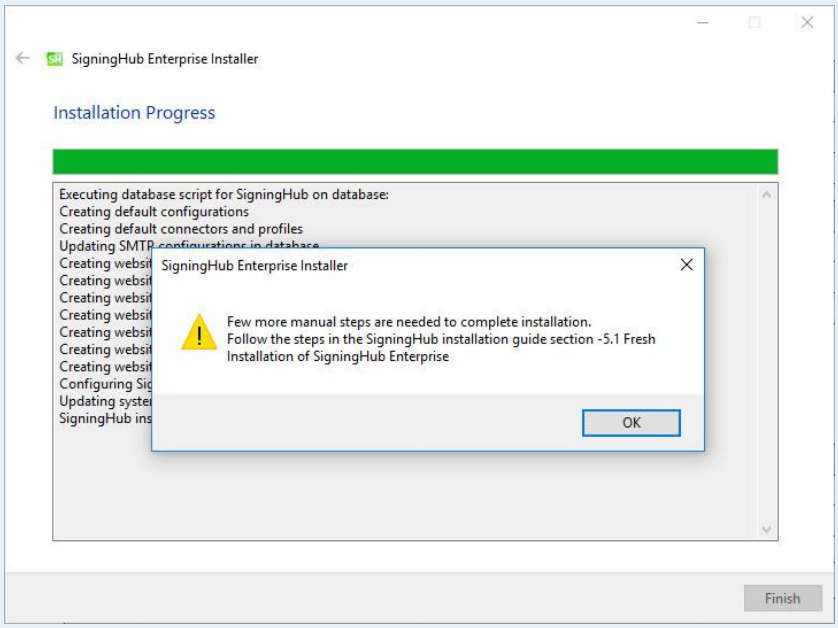
```
"<sessionState mode="SQLServer"
allowCustomSqlDatabase="true"
sqlConnectionString="AdocsEntities" cookieName="SH_ID"
timeout="60" compressionEnabled="true">
   </sessionState>"
```

With:

```
"<sessionState mode="InProc" timeout="60"
cookieName="SH_ID" />"
```

You can configure an AJP Connector as mentioned in Appendix A

Once the installation progress is complete for Oracle DBMS, an alert will be displayed before you could click the Finish button, as shown in the following image:



You must execute some scripts manually when SigningHub Enterprise is installed with Oracle Database Management System:

14) Copy "**session**" folder from **[Installation Directory/setup/db-scripts/Oracle** and paste it in any directory on the Oracle server.

15) Login to the Oracle Server (For Linux, use Putty while for Windows use cmd)

16) Go to directory (CD [File Path])

17) Write command "`sqlplus`"

18) Connect to database using "**User Name**" and "**Password**" which was given for installation of SigningHub Enterprise (e.g. `[signinghub user name]@[service_name]`)

19) Execute command: `SQL>@InstallAllOracleASPNETProviders.sql`;

5.1.1 SigningHub Enterprise URLs

Use these URLs to access the SigningHub Enterprise web sites:

Service	URL Format	Example
SigningHub Enterprise Admin	https://<machine-name>:PORT	https://localhost:443
SigningHub Enterprise Desktop Web	http://<machine-name>:PORT	http://localhost:81
SigningHub Enterprise API	http://<machine-name>:PORT	http://localhost:82
SigningHub Mobile Web	http://<machine-name>:PORT	http://localhost:83
SigningHub Enterprise Demo	http://<machine-name>:PORT	http://localhost:85
SigningHub Core	http://<machine-name>:PORT	http://localhost:86
ADSS Signing Server	https://<machine-name>:8774/adss/console	https://localhost:8774/adss/console

Where necessary (i.e. browsing Admin website or ADSS Signing Server Console) your web browser will prompt you to select the appropriate certificate for authentication purposes. Note the installation process places the necessary certificates into the Windows Security Store, so Internet Explorer, Edge, Chrome and related browsers that rely on the security store can use them as such.

If you wish to use Firefox and similar web browsers that utilise their own respective security stores you will need to import **"ads-default-admin.pfx"** and **"signinghub-default-admin.cer"** from **"**`[SigningHub Installation Directory]\setup\certs`**"** directory.

When SigningHub Mobile Web is installed as part of SigningHub installation, the installer adds the redirection URLs for Mobile Web e.g. http://machine-name:83/ in [SigningHub-Installation-Dir]/web/web.config file under rewrite element. This is enough if the SigningHub Mobile Web needs to be accessed only within the organization's internal network.

If the SigningHub Mobile Web needs to be publicly accessible, then it is required to manually update these redirection URLs in the web.config file so that interactive version of SigningHub Mobile Web is launched when accessed from a mobile device e.g. replace all http://machine-name:83/ with the configured mobile URL https://mobile.signinghub.com/ .

There are two options to set secure binding against each SigningHub site:

1. Using standard IIS web server HTTP redirects. This means the basic installation is done with various SigningHub sites, where each site has their respective default port/binding but no hostname. You can then add new sites for each web site and bind this to the desired external public facing hostname and secure port, likely to be 443. Each site can be configured in such a fashion. Each default SigningHub site can then be configured to permanently redirect to the secure version.

2. Once the deployment of SigningHub is complete the bindings of each site can be changed to use a secure (443) port. The new binding will include the appropriate public facing hostname.

The preferred one is option two.

Once the bindings of IIS web sites have been put in place, access the SigningHub administration console and make changes to the general configuration settings. This means changing the public and private URLs for the Desktop Web and API sites accordingly. Once done save the changes and Publish them.

Public addresses should also be updated in the following files:

1) Configure the **Mobile Web** public URL in **[SigningHub-Installation-Dir]/web/web.config** file e.g. replace http://machine-name:83/ with the configured mobile URL https://mobile.signinghub.com/.

20) Configure the **API** public URL in **[SigningHub-Installation-Dir]/mobile/web.config** file e.g. replace http://machine-name:82/ with the configured mobile URL https://api.signinghub.com/.

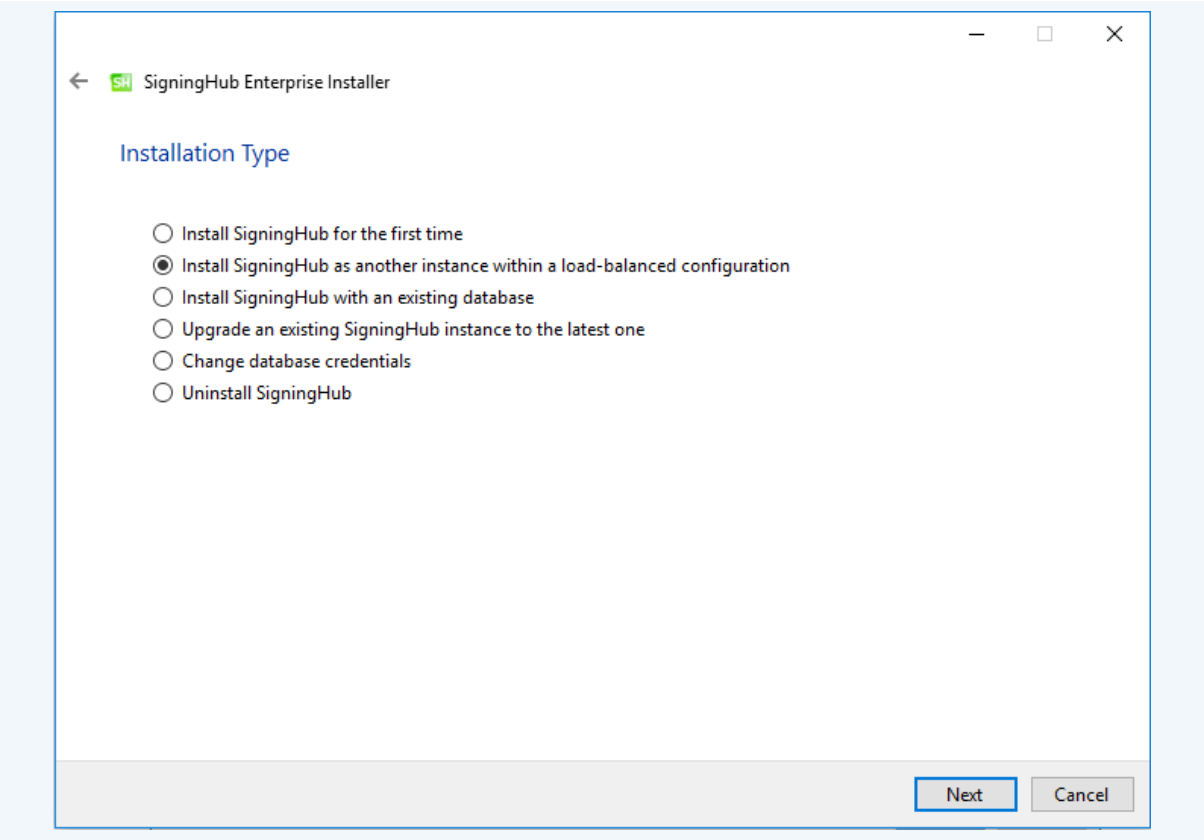
Please note that for securing the websites you have to follow the Appendix B, C and D of this document.

5.2 Installing SigningHub Enterprise with a Load-Balanced Configuration

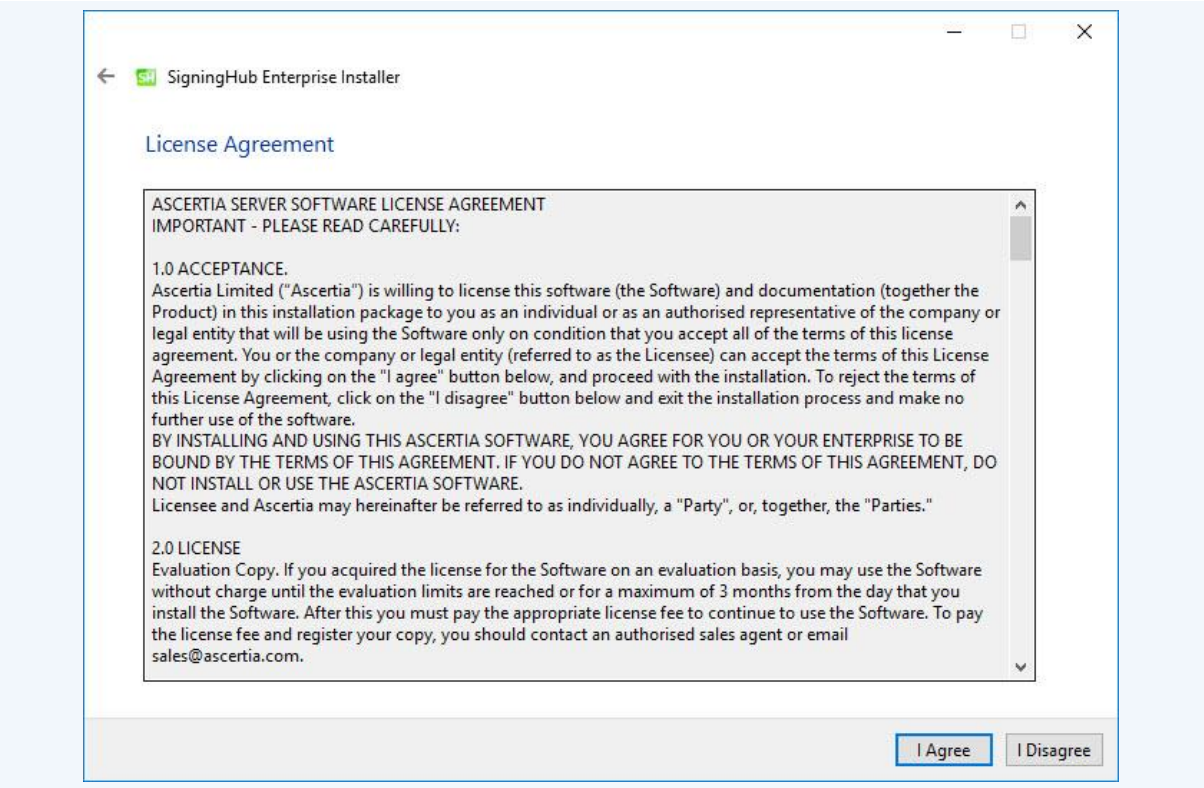
Follow these instructions to install SigningHub Enterprise with a load-balanced configuration.

Launch the installer by right-clicking the file name **"[SigningHub Installation Directory]/setup/install.bat**" and select Run as administrator.

Follow the installation wizard as described previously until the '**Installation Type'** screen is shown:

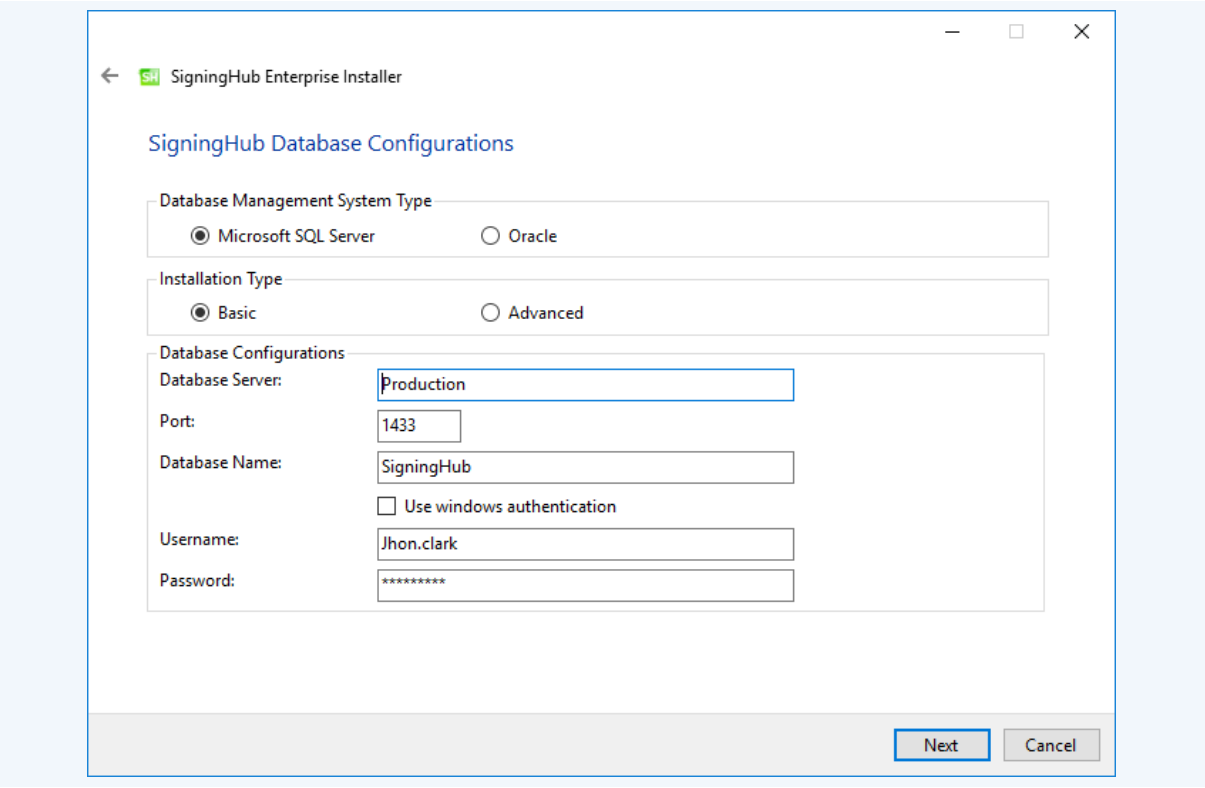


Click the **Next** button to show the License Agreement:



Click the **I Agree** button to proceed.

The following screen to prompt for database details is displayed:

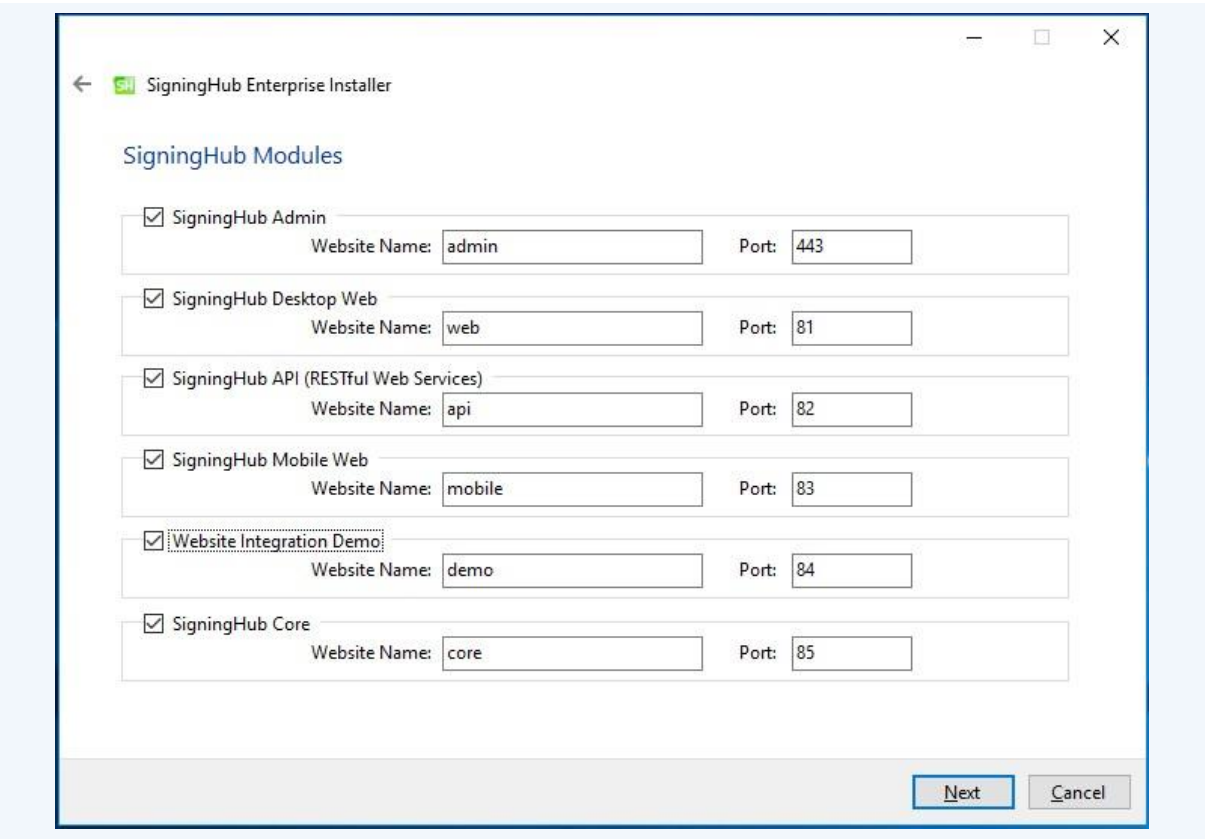


The information displayed above is an example and you should configure the relevant settings for your own environment.

The SigningHub Enterprise database schema and the version required by the installer must be the same.

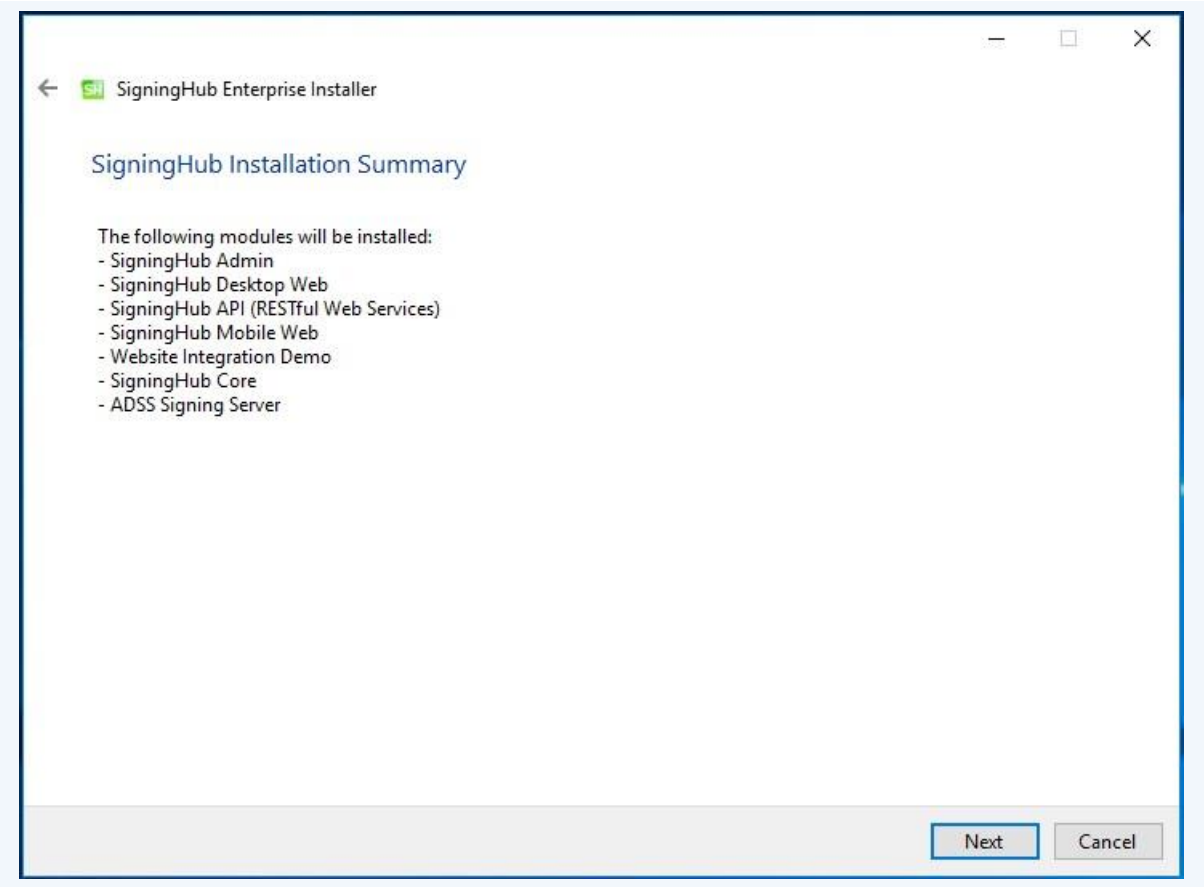
If the current SigningHub Enterprise database schema is older than the version required by the installer, and you click Next, the installer will prompt you that SigningHub Enterprise database schema will be upgraded to the latest version. Click **OK** to authorise the schema update.

Click the **Next** button to select specific modules:



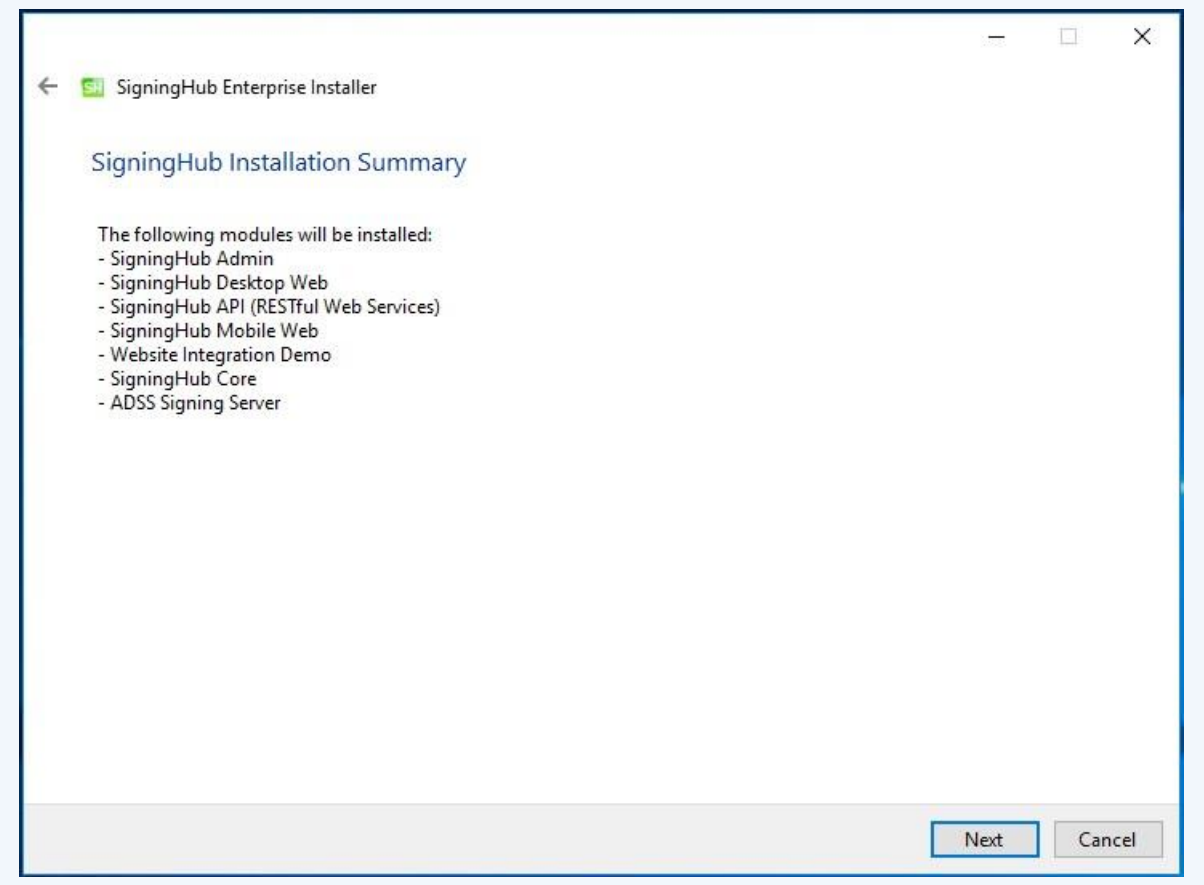
Select the appropriate modules to install the required features.

Click the **Next** button to show the summary and complete the installation:

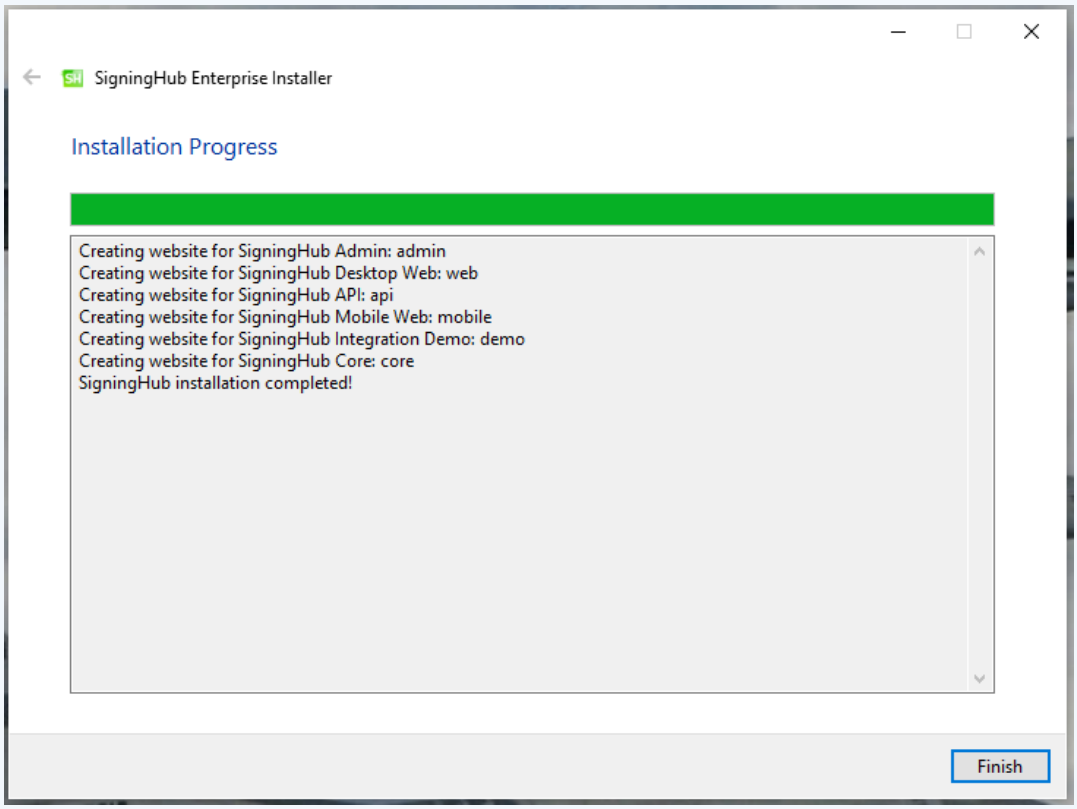


This screen shows the installation summary by listing the different product modules that will be installed.

If you think any listed item is incorrect then use the **Back** button (arrow towards the top-left of the dialogue box) to correct your choices before proceeding ahead.



Click the **Next** button to continue with the installation.



Click **Finish** to complete the installation process.

The site IDs of deployed IIS websites should be the same across all the instances in a load balanced environment to run SigningHub Enterprise properly.

Please note that Local Signing will not be available in a Load Balanced environment by using sessionState mode “SQLServer”. However, if users still require performing Local Signing, then please follow these steps to replace sessionState mode with” InProc”:

1) Open the **[SigningHub-Installation-Dir]/web/web.config** file of all deployments of SigningHub:

2) Replace:

```
“<sessionState mode="SQLServer"
allowCustomSqlDatabase="true"
sqlConnectionString="AdocsEntities" cookieName="SH_ID"
timeout="60" compressionEnabled="true">
    </sessionState>”
```

With:

```
“<sessionState mode="InProc" timeout="60"
cookieName="SH_ID" />”
```

3) Configure sticky session on load balancer server

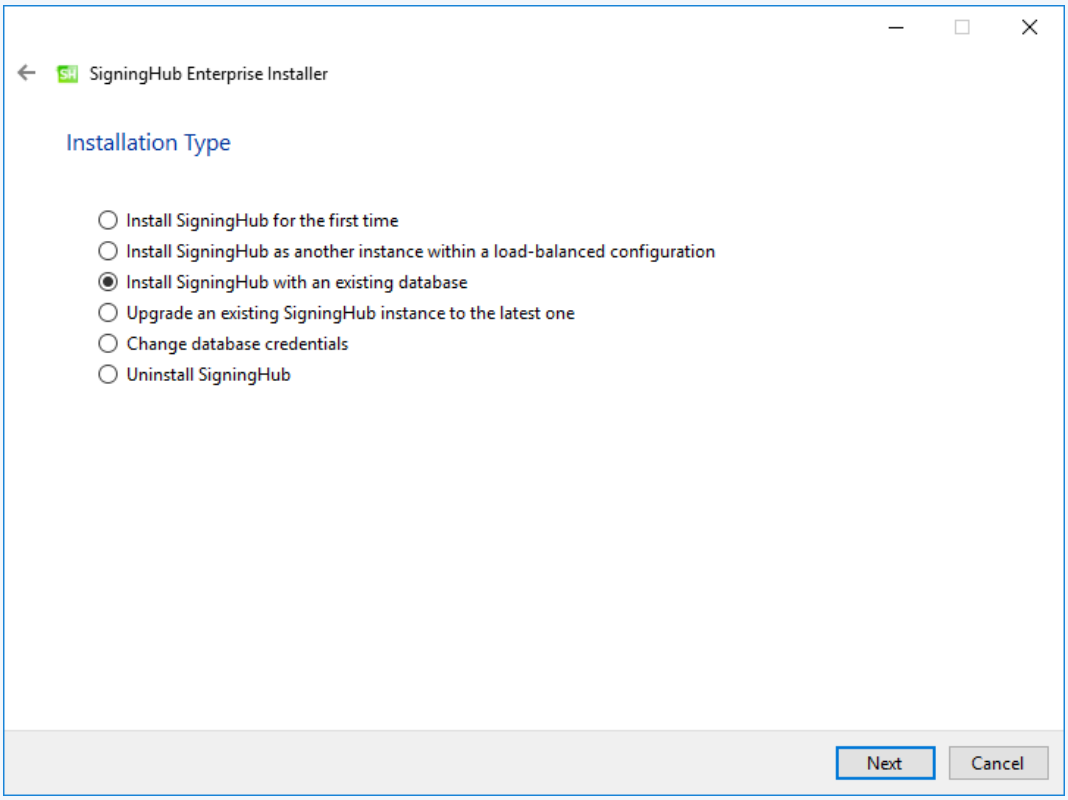
4) Configure an AJP Connector as mentioned in Appendix A

Also note that if someone wants to change email templates in case of a Load Balanced environment, then email templates will have to be replaced manually across all instances.

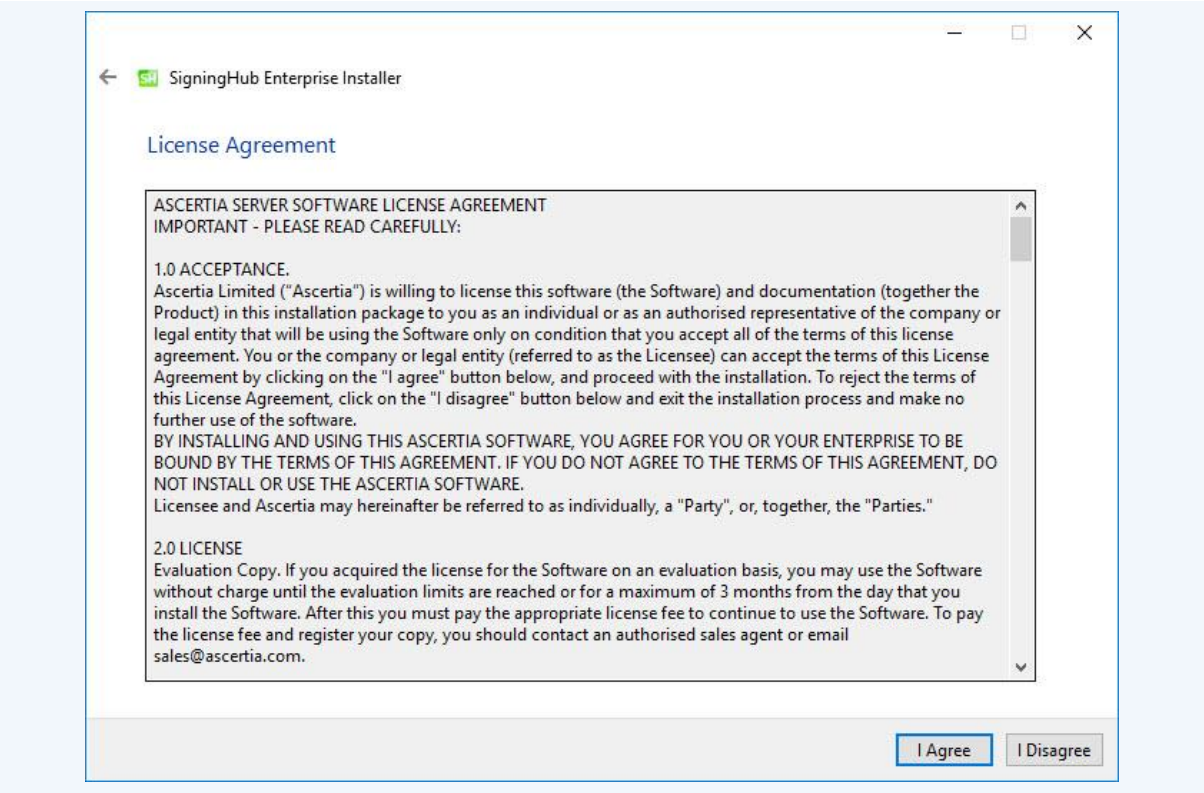
5.3 Installing SigningHub Enterprise with an Existing Database

Follow these instructions to install SigningHub Enterprise with an existing database.

Launch the installer by right-clicking the file name **"[SigningHub Installation Directory]/setup/install.bat**" and select **Run as administrator**. Follow the installation wizard as described previously until the '**Installation Type'** screen is shown:

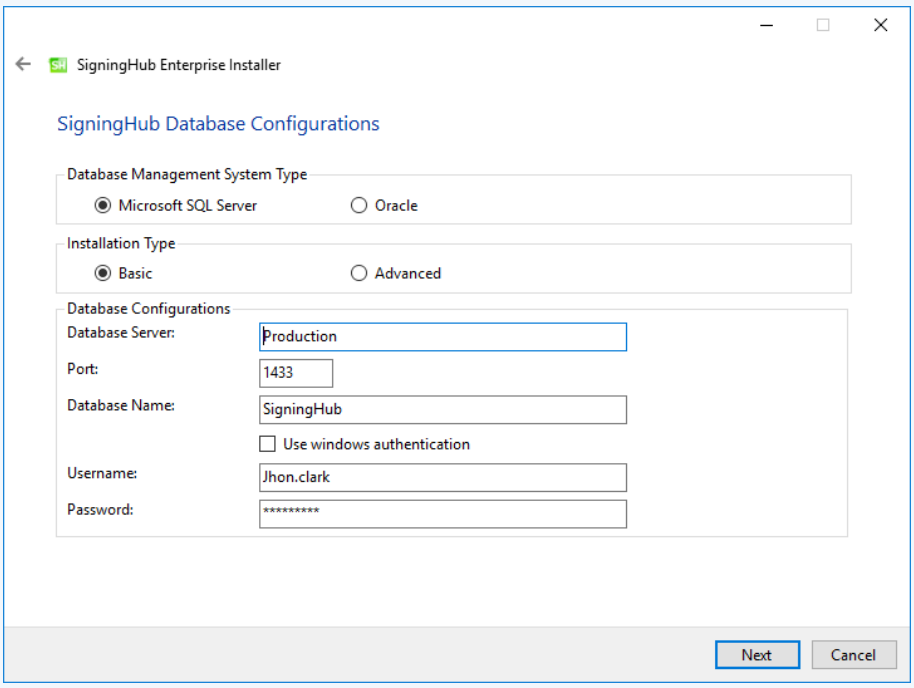


Click the **Next** button to show the License Agreement:



Click the **I Agree** button to proceed.

The following screen to prompt for database details is displayed:



Username and Password must not contain the following characters: [] () , ; ? * ! @ =. These characters are used to either initialize a connection or separate connection values.

The information displayed above is an example and you should configure the relevant settings for your own environment.

The SigningHub Enterprise database schema and the version required by the installer must be the same.

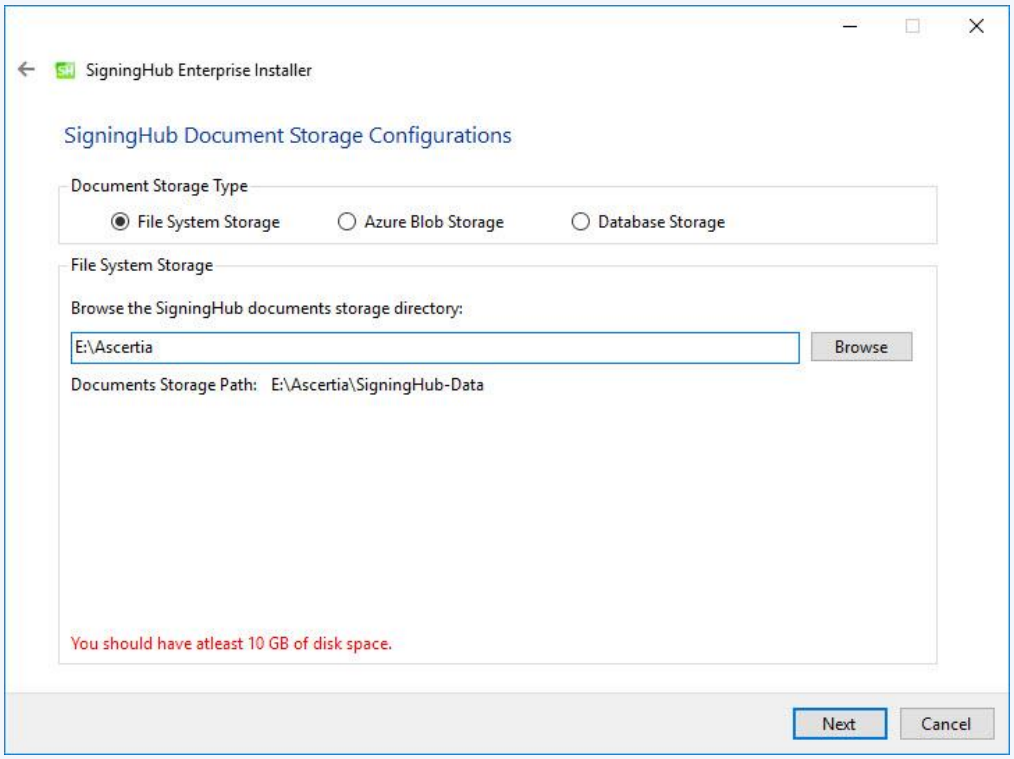
If the current SigningHub Enterprise database schema is older than the version required by the installer, and you click **Next**, the installer will prompt you that SigningHub Enterprise database schema will be upgraded to the latest version. Click OK to authorise the schema update.

Click the **Next** button to select the SigningHub data storage directory:

Document Storage Configurations screen will not appear if it is already configured in existing database.

On the SigningHub Document Storage Configurations screen you can either choose a "**File System Storage"** or **"Azure Blob Storage"** or "**Database Storage"**.

If the Document Storage is either on local file system or on the local network path, then select the option **"File System Storage"**.



The information displayed above is an example and you should configure the relevant settings for your own environment.

Click **Browse** and specify a storage path to store the SigningHub data.

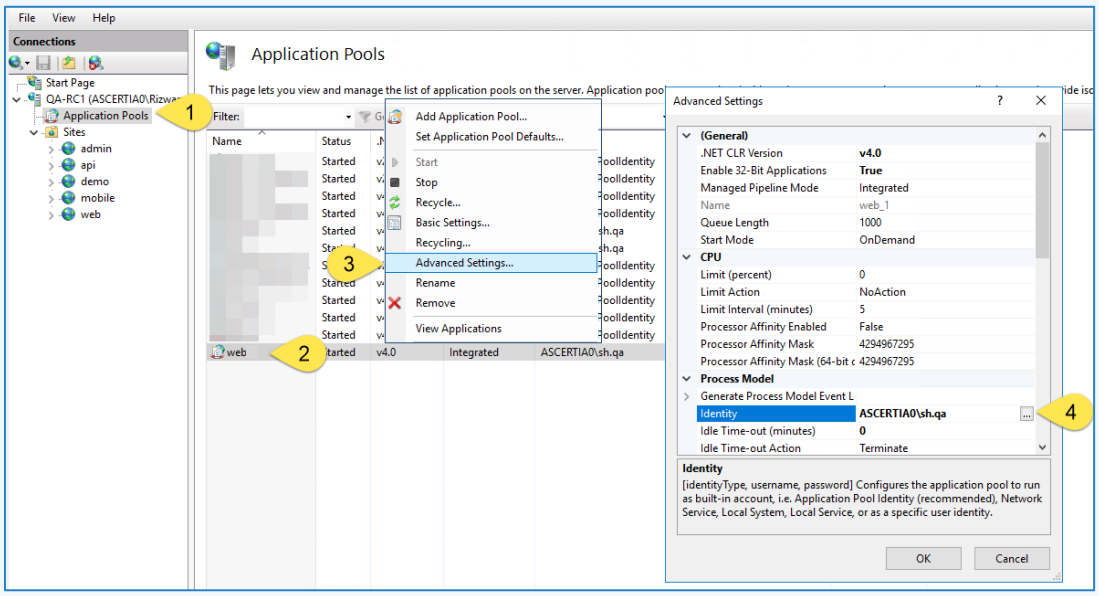
Document Storage path can be a local drive, a network drive or an Azure blob. If the path is on a local drive, then the installer will automatically assign the read/write permissions to the "IIS_IUSRS" user group.

If the path is on a network/Azure drive, then the permissions should be assigned manually to a user before continuing the installation process. To add the permissions on a network drive, follow these instructions:

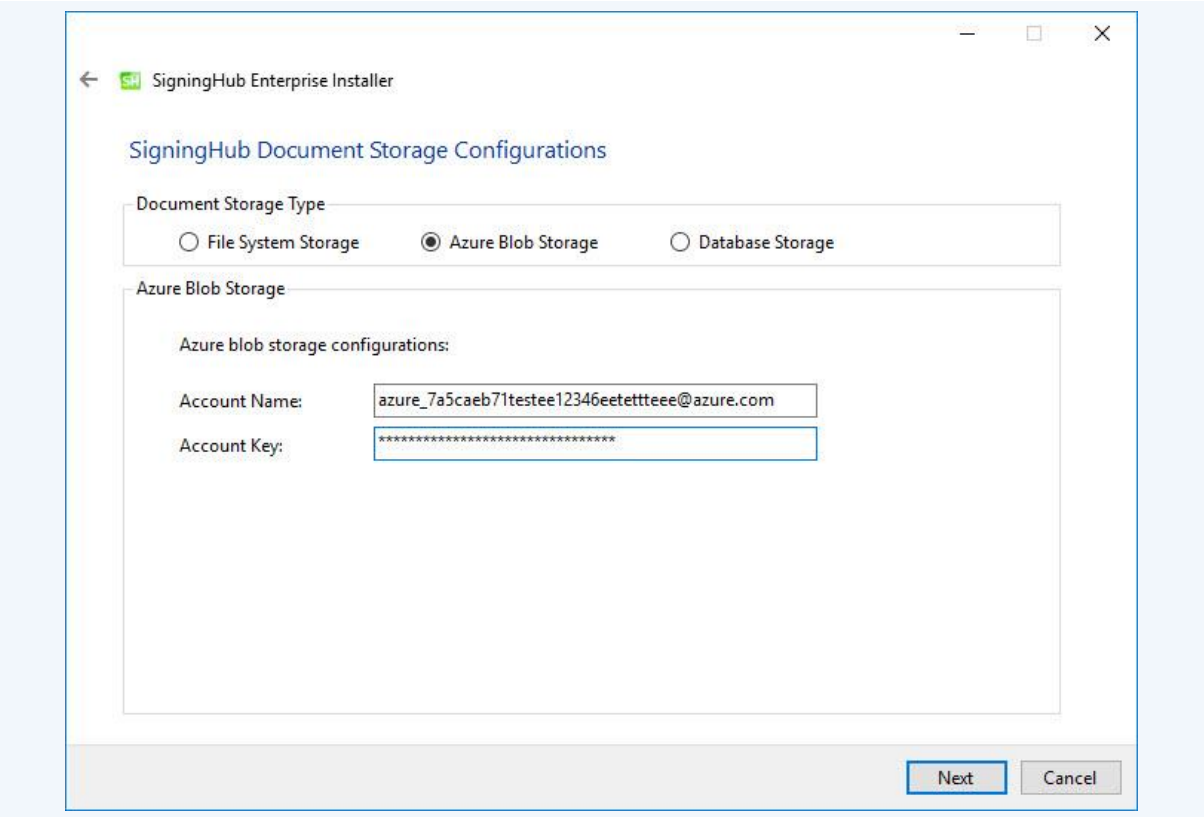
1) Create a domain/Azure user with read/write permissions.

5) Add the read/write permissions on the directory [Document Storage Path] and complete the installation process.

6) Now go to **IIS Manager** and add the user created in step 1 in Application Pool against all SigningHub websites one by one as shown below (Skip this step if SigningHub Enterprise is installed by using Windows Authentication):



If this is not a File System Storage and you choose the second option to **"Azure Blob Storage"** then the following screen is shown:

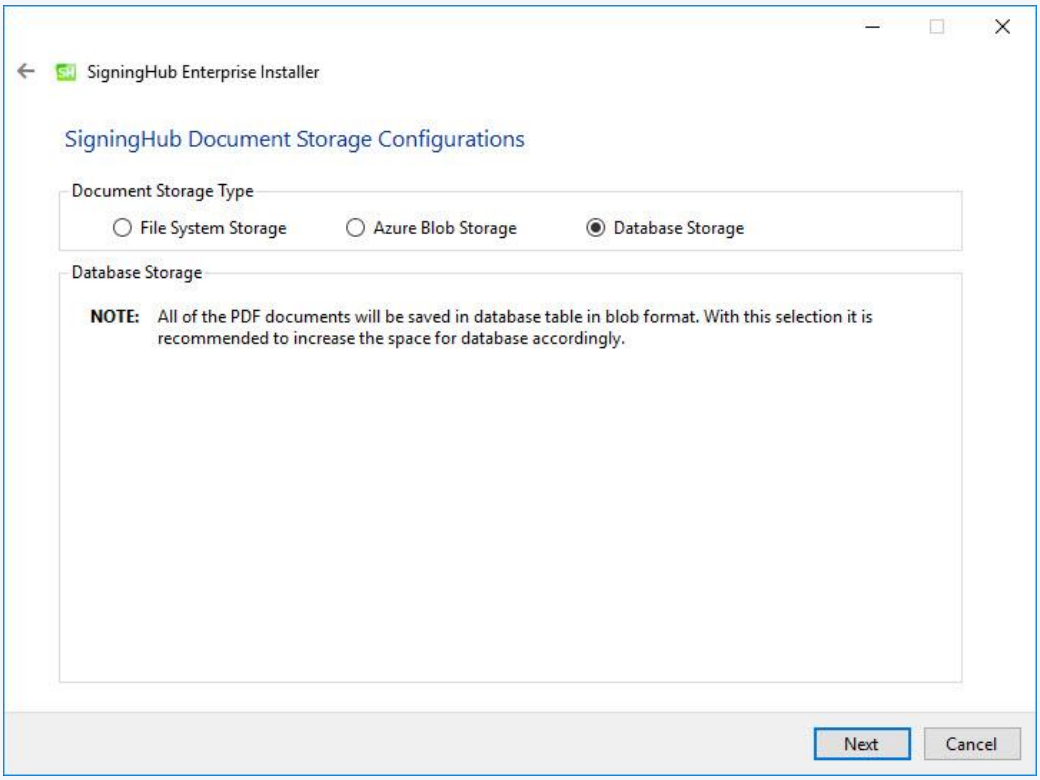


The information displayed above is an example and you should configure the relevant settings for your own environment.

The following table details the configuration options:

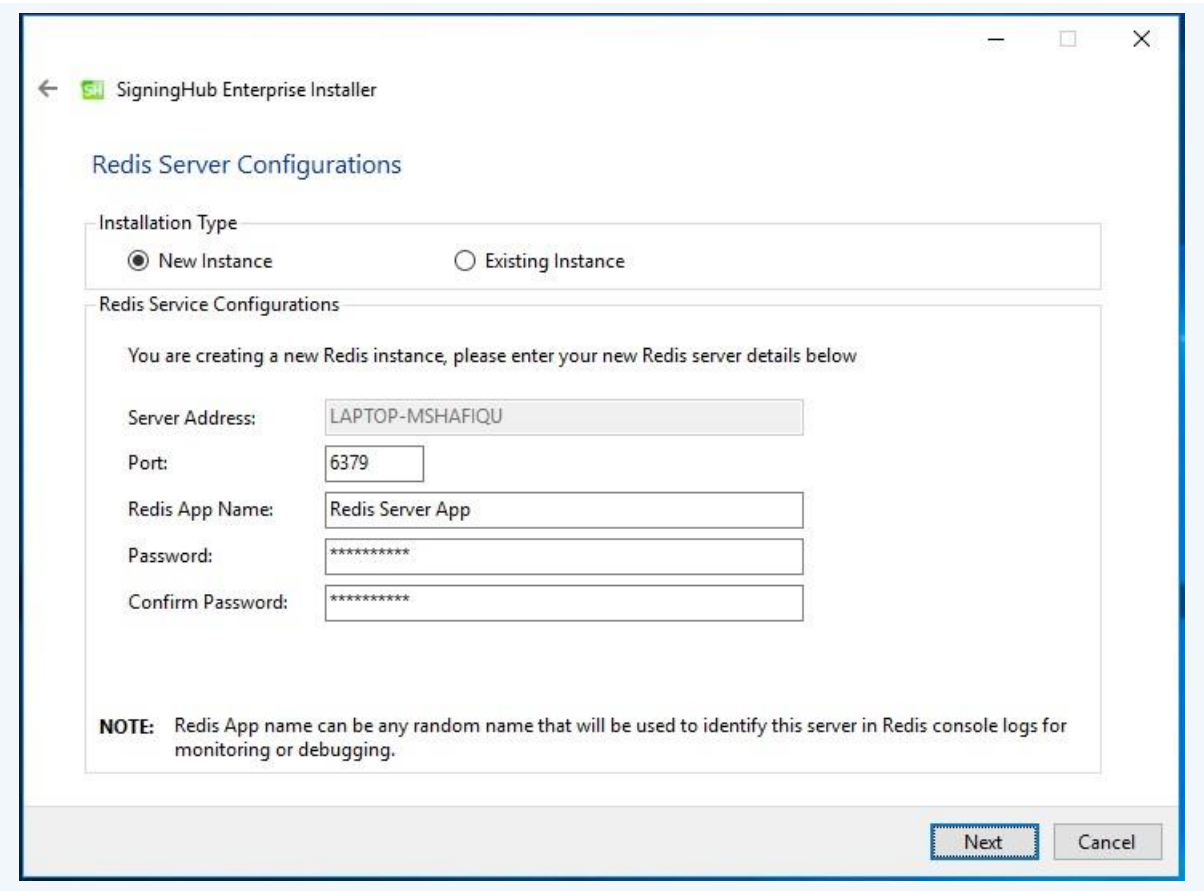
Item	Description
Account Name	Account Name of the Azure. Note this must exist prior to the installation.
Account Key	Account Key of the Azure. Note this must exist prior to the installation.

If this is not a File System Storage and you choose the third option to **"Database Storage"** then the following screen is shown:



Click the **Next** button to proceed. The following screen for Redis Server will appear:

Redis Server Configurations screen will not appear if it is already configured in existing database.



Redis is a light weight server, which works as backplane and message broker for SigningHub application over an HTTP/s port. For further information, https://redis.io/

It is bundled within SigningHub package. SigningHub will communicate with the server on that HTTP/s port. It is like SQL Server you need to have an account on Redis server, password is optional.

1) For new instance of Redis Server we create that account with provided details automatically in **New Instance** screen. It will install a new instance of Redis on the current machine and the configurations will be saved in SigningHub database.

7) If by any chance you have Redis server installed or you want to use Redis server from Azure or Amazon, you need to know the app name, password and port to connect to that instance. In that case, select **Existing Instance** in the above screen.

On the Redis Server Configurations screen you can either choose a **"New Instance"** or **"Existing Instance"** option.

If this is a new instance, then use the first option i.e. "**New Instance**" and provide the appropriate Redis server configurations.

The information displayed above is an example and you should configure the relevant settings for your own environment.

The following table details the configuration options:

Item	Description
Server Address	Specify the Redis server address. This server is used to send real-time on-screen notifications for document sharing. In future will be used for cache and message broker.
Port	Specify the service port for the Redis server. It is automatically populated the available port
Redis App Name	Specify the name of Redis App. This can be any random name that will be used to identify this server in Redis console logs for monitoring or debugging.
Password	Specify the password to authenticate the Redis server.
Confirm Password	Specify the same password again as provided in the above password field to confirm it.

Redis can enforce password-based security to save or read the key value pairs from the Redis server. To enable password-based security, follow these instructions:

1) Go to `[SigningHub Installation Directory]/Redis`

8) Run the Redis command line interface by click on “redis-cli” application in administrator mode

9) Run the command **“CONFIG SET requirepass "[password]"”**

10) Sign into SigningHub Administrator account

11) Go to **Configurations>Redis** and change the password in Redis Server Connection String

12) Update the settings and **Restart IIS**

Redis can disable asking for password for saving and reading the key value pairs from the Redis server. To turn off the password, follow these instructions:

1) Go to `[SigningHub Installation Director]/Redis`

13) Run the Redis command line interface by click on “**redis-cli**” application in administrator mode

14) Run the command “**CONFIG SET requirepass ""**”

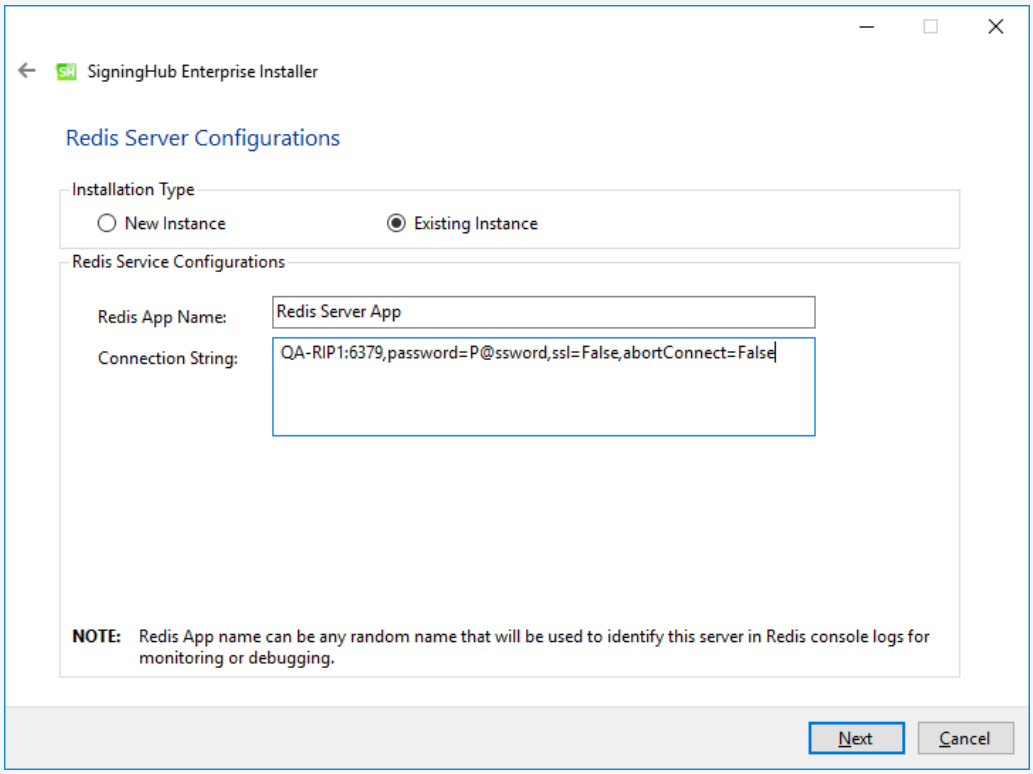
15) Sign into SigningHub Administartor account

16) Go to Configurations>Redis and change the password as empty in **Redis Server Connection String**

17) Update the settings and **Restart IIS**

For Load balanced deployments, only one instance of Redis is needed for SigningHub to work with. Rest of the instances of SignignHub will communicate with Redis using HTTP/s address and Port configured in SigningHub Admin.

If this is not a new instance, and you are choosing the second option i.e. "**Existing Instance"** then the following screen will appear:

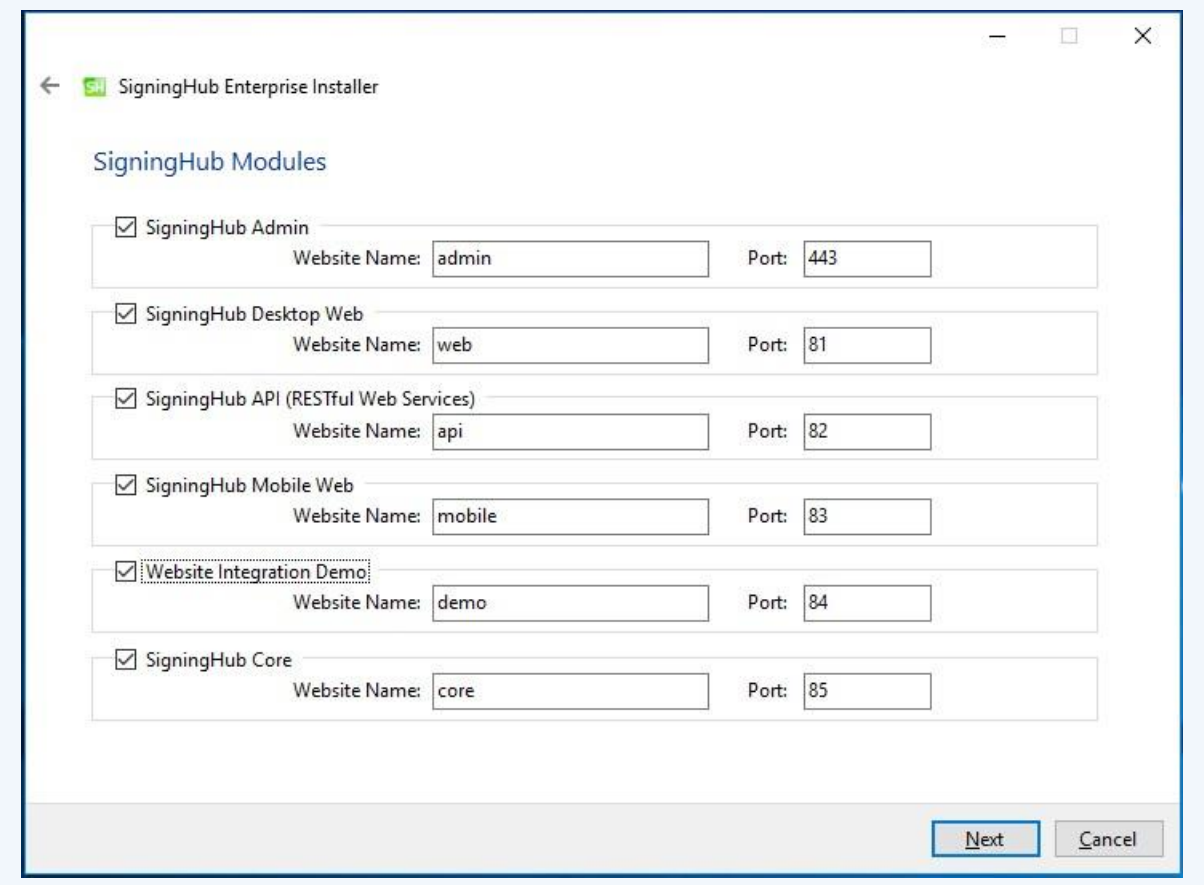


The information displayed above is an example and you should configure the relevant settings for your own environment.

The following table details the configuration options:

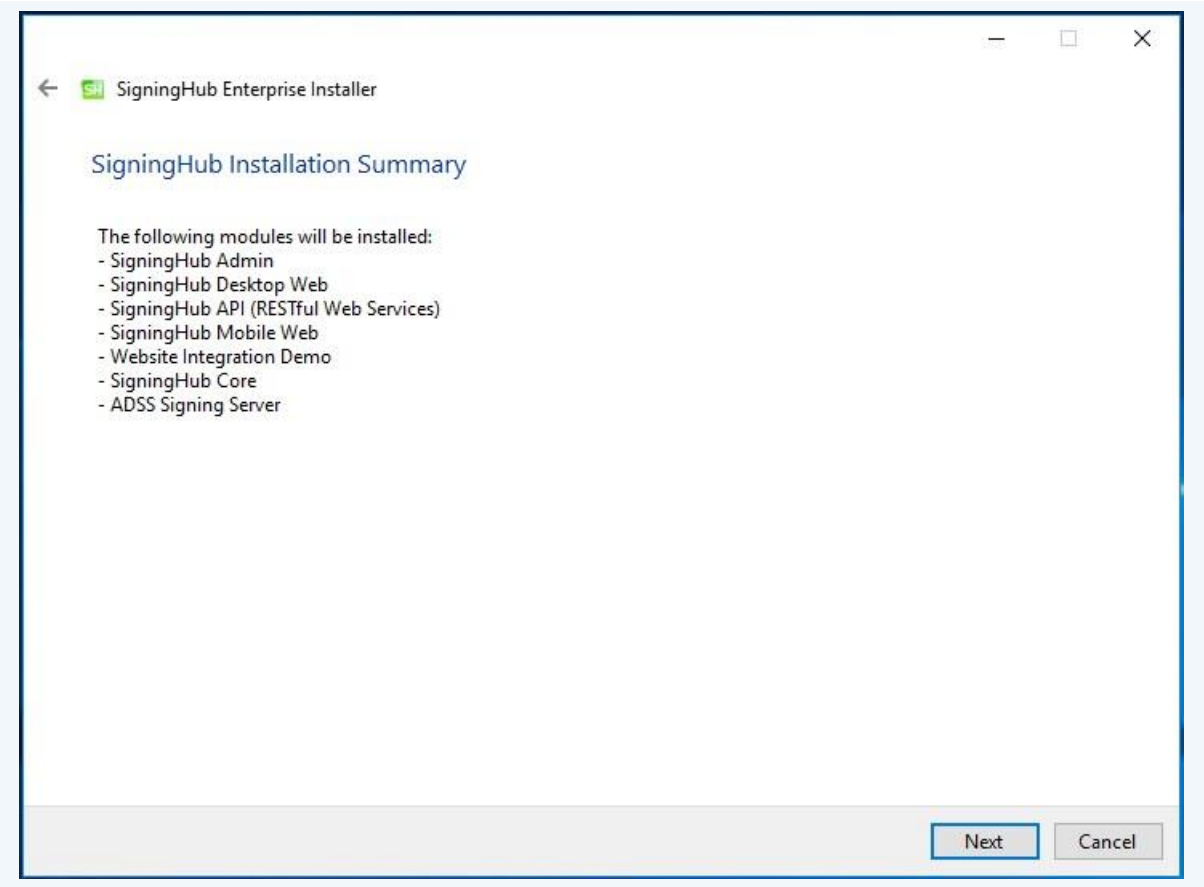
Item	Description
Redis App Name	Specify the name of Redis App. This can be any random name that will be used to identify this server in Redis console logs for monitoring or debugging.
Connection String	The following is a sample connection string for a Redis Server: `"[Redis Server Address]: [port], password=[Redis Server Password],ssl=False,abortConnect=False"`

Click the **Next** button to select specific modules: -



Select the appropriate modules to install the required features.

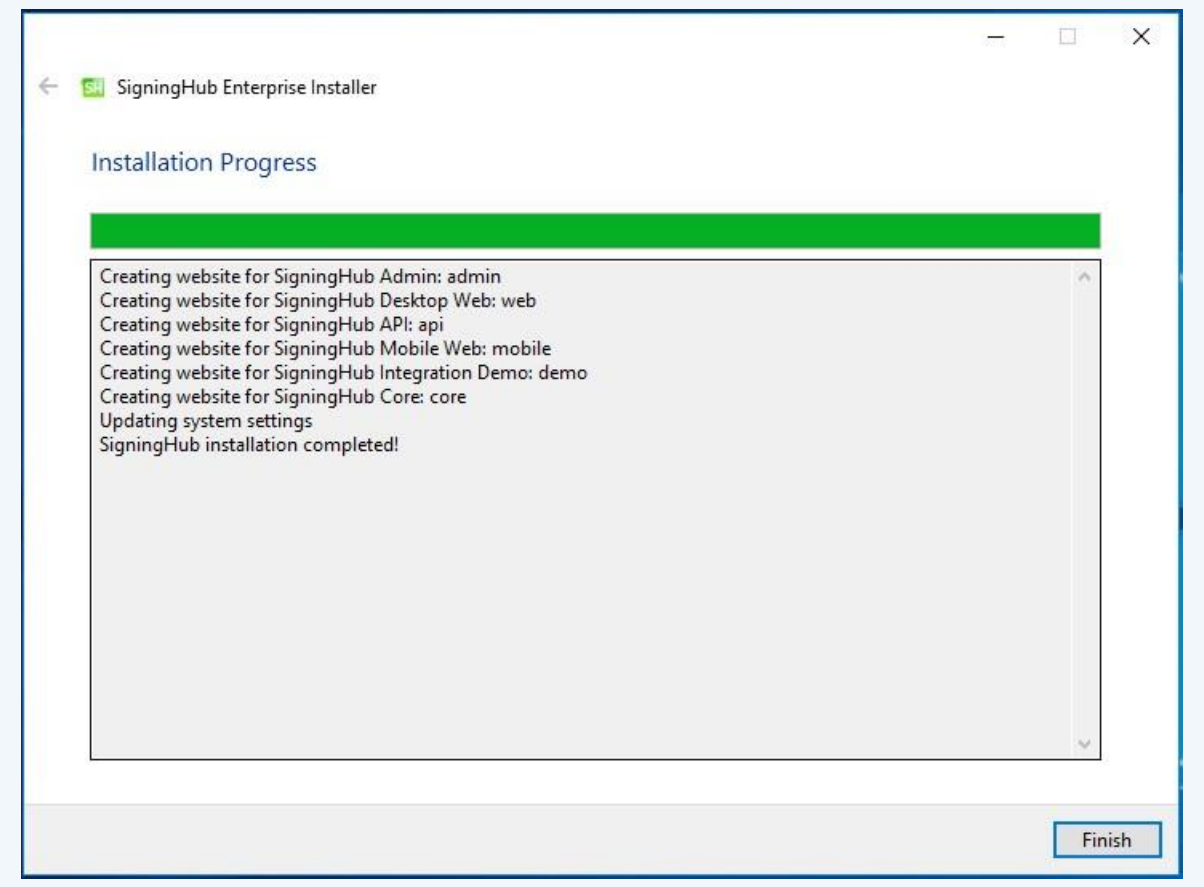
Click the **Next** button to show the summary and complete the installation:



This screen shows the installation summary by listing the different product modules that will be installed.

If you think any listed item is incorrect then use the **Back** button (arrow towards the top-left of the dialogue box) to correct your choices before proceeding ahead.

Click the **Next** button to continue with the installation.



Click **Finish** to complete the installation process.

5.4 Post Installation Steps

1) After installing SigningHub Enterprise, provide read/write access to the **IIS_IUSRS** user on **Document Storage Directory** if it is created on a network path.
2) If you have installed ADSS Signing Server with Windows Authentication, then make sure that the following services in **Windows Services Panel** are running under a domain user, otherwise these services may fail to start:
3) Ascertia-ADSS-Console
4) Ascertia-ADSS-Core
5) Ascertia-ADSS-Service

5.5 Upgrading SigningHub Enterprise

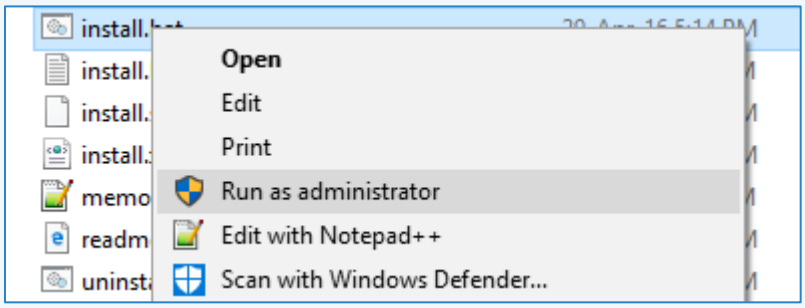
The upgrade process for SigningHub Enterprise is quick and easy. The existing data files, database schema and database entries are automatically upgraded during the process when the base (current) installation is v6.2 or higher. To upgrade to version 6.2 from version 5.x contact support@ascertia.com.

Read the information in SigningHub-Upgrade-Information.pdf file before proceeding the upgrade.

5.5.1 ADSS Signing Server Dependency

ADSS Signing Server should be upgraded to the latest version before upgrading SigningHub Enterprise. The instructions to complete the ADSS Signing Server upgrade are:

- Take a backup of the ADSS Signing Server and SigningHub Enterprise databases.
- Download the latest SigningHub Enterprise package and extract it on the same server machine, where the system is currently deployed but in a different folder.
- Go to the location "**[New SigningHub Installation Directory]/tools/adss-server/setup**" folder.
- Take a backup of the "**install.xml**" file by renaming it to a different name, e.g. "**install.xml.backup**".
- Rename the "**install.xml.upgrade**" file to "**install.xml**".
- Run the **install.bat** file with administrative privileges (otherwise ADSS Signing Server services will not be registered in Windows Services Panel) to launch the installer.



For further details, follow the instructions in the **section 3.1.3** of **ADSS-Server-Installation-Guide.pdf** which can be found in the **[SH-Home]/tools/adss-server/docs** directory.

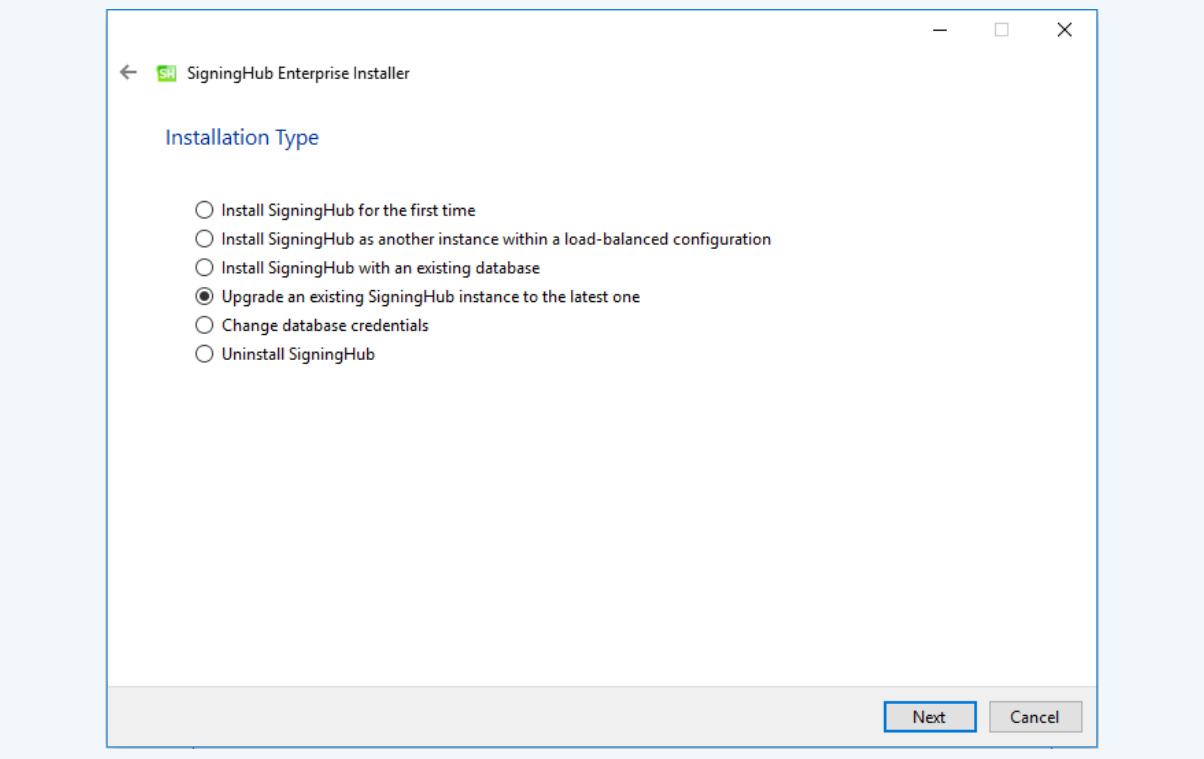
Note if you are upgrading from ADSS Signing Server v4.5.7 or an earlier version to the latest version, then you must obtain a new ADSS Signing Server license file.

5.5.2 Upgrade Procedure

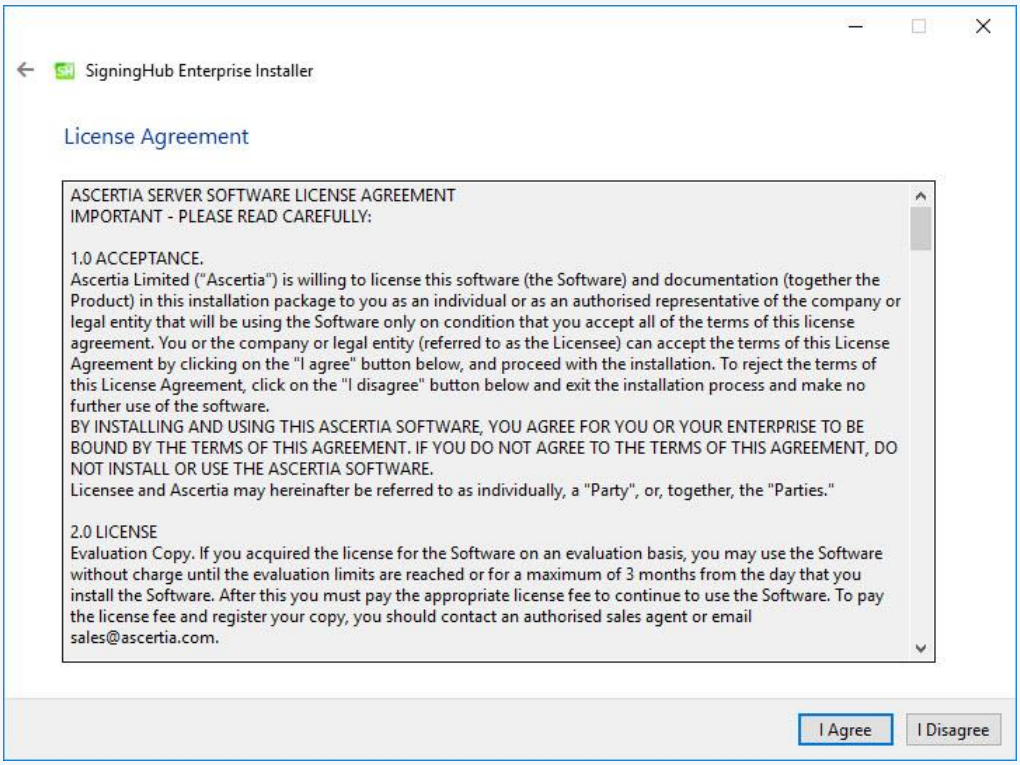
Follow these instructions to upgrade an older version of SigningHub Enterprise to the latest version.

Launch the installer by right-clicking the file name "`[SigningHub Installation Directory]/setup/install.bat`" and select **Run as administrator**.

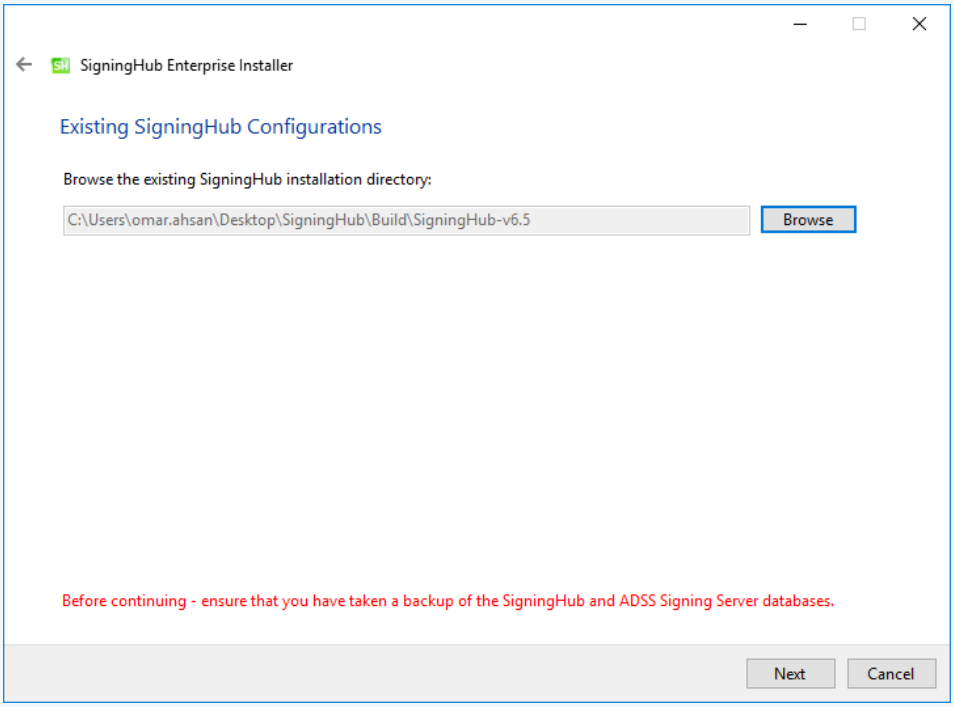
Follow the installation wizard as described previously until the '**Installation Type'** screen is shown:



Click the **Next** button to view and accept the license agreement:



Click the **I Agree** button to proceed.

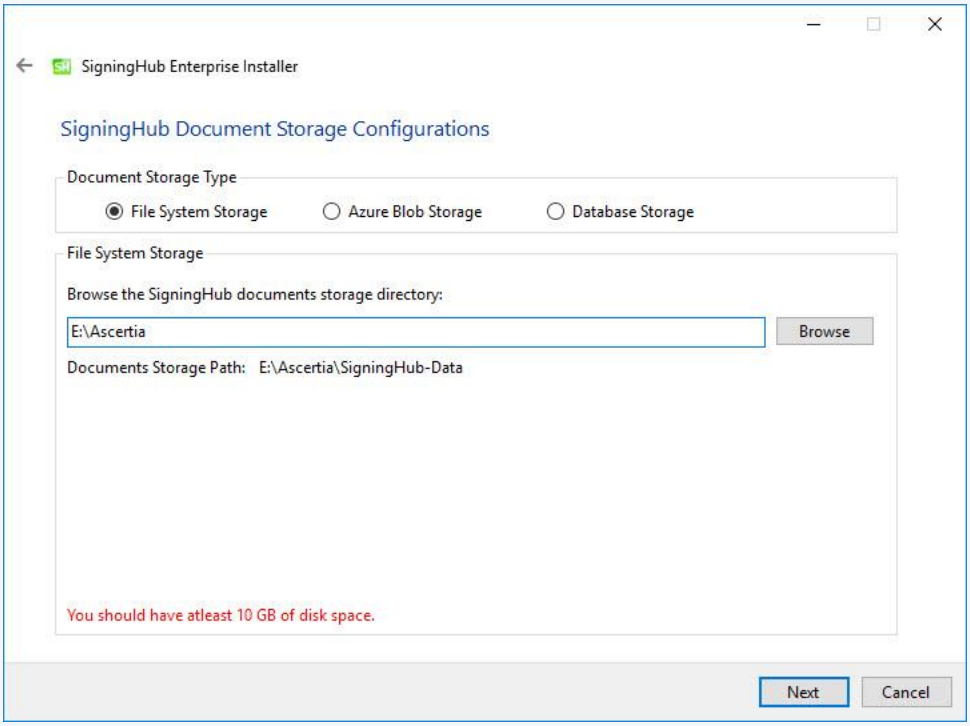


Click **Browse** and define the path to the existing SigningHub Enterprise installation directory.

Click the **Next** button to select the SigningHub data storage directory:

On the SigningHub Document Storage Configurations screen you can either choose a "**File System Storage"** or **"Azure Blob Storage"** or "**Database Storage"**.

If the Document Storage is either on local file system or on the local network path, then select the option **"File System Storage"**.



The information displayed above is an example and you should configure the relevant settings for your own environment.

Click **Browse** and specify a storage path to store the SigningHub data.

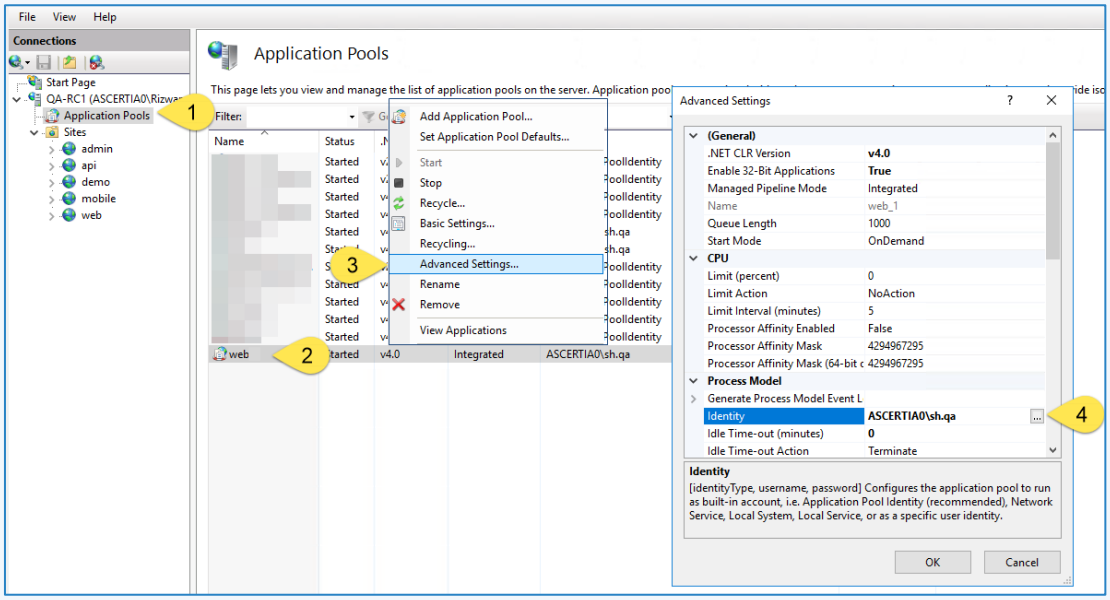
Document Storage path can be a local drive, a network drive or an Azure blob. If the path is on a local drive, then the installer will automatically assign the read/write permissions to the "IIS_IUSRS" user group.

If the path is on a network/Azure drive, then the permissions should be assigned manually to a user before continuing the installation process. To add the permissions on a network drive, follow these instructions:

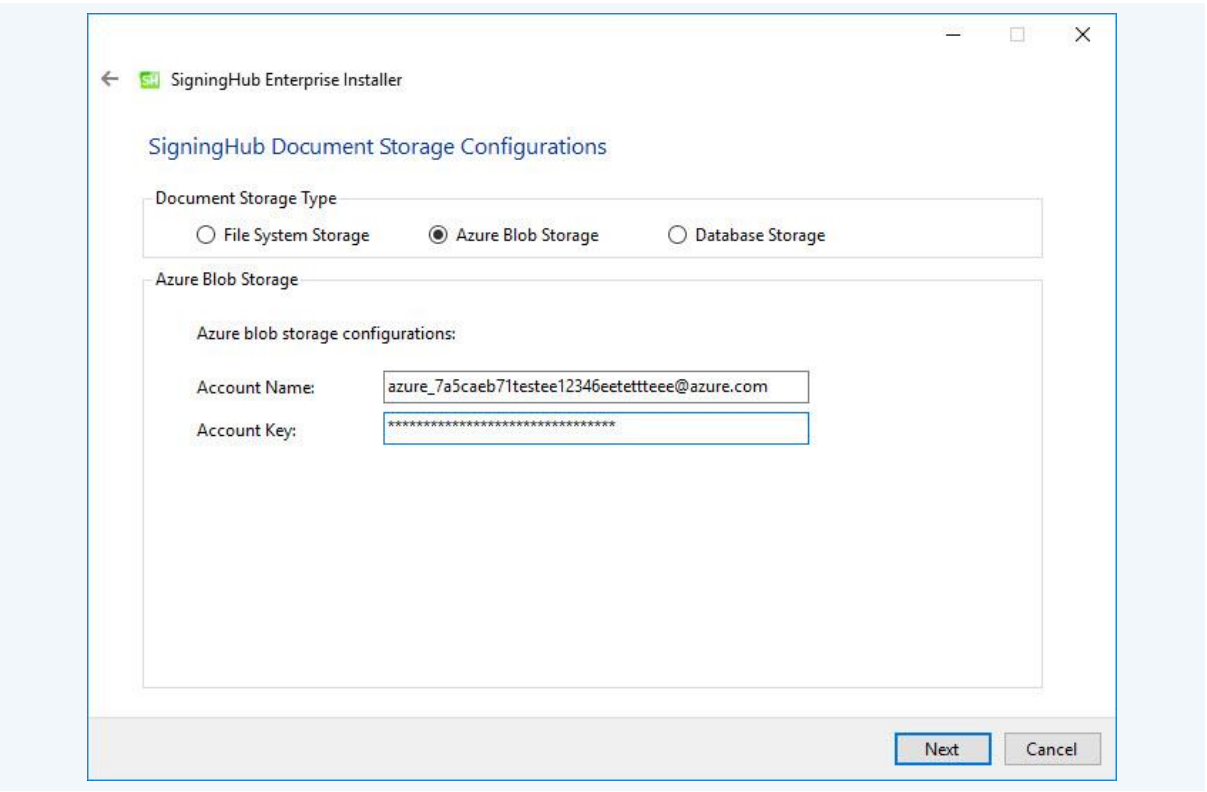
1) Create a domain/Azure user with read/write permissions.

18) Add the read/write permissions on the directory [Document Storage Path] and complete the installation process.

Now go to IIS Manager and add the user created in step 1 in Application Pool against all SigningHub websites one by one as shown below (Skip this step if SigningHub Enterprise is installed by using Windows Authentication):



If this is not a File System Storage and you choose the second option to **"Azure Blob Storage"** then the following screen is shown:

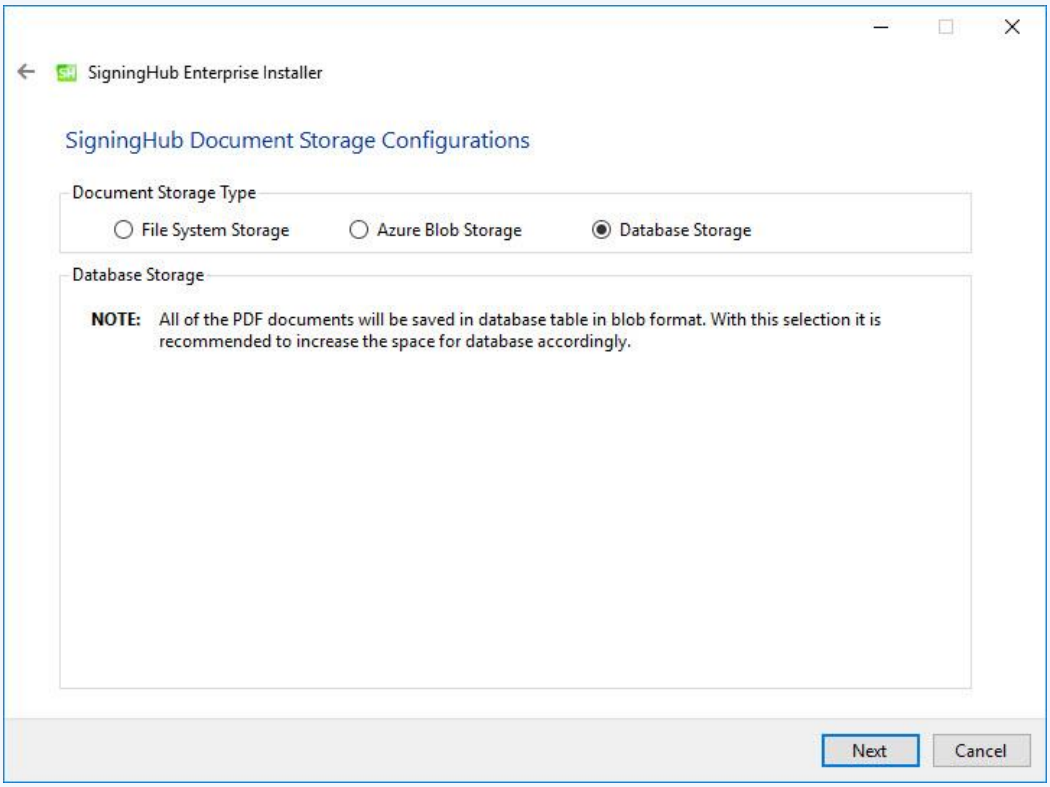


The information displayed above is an example and you should configure the relevant settings for your own environment.

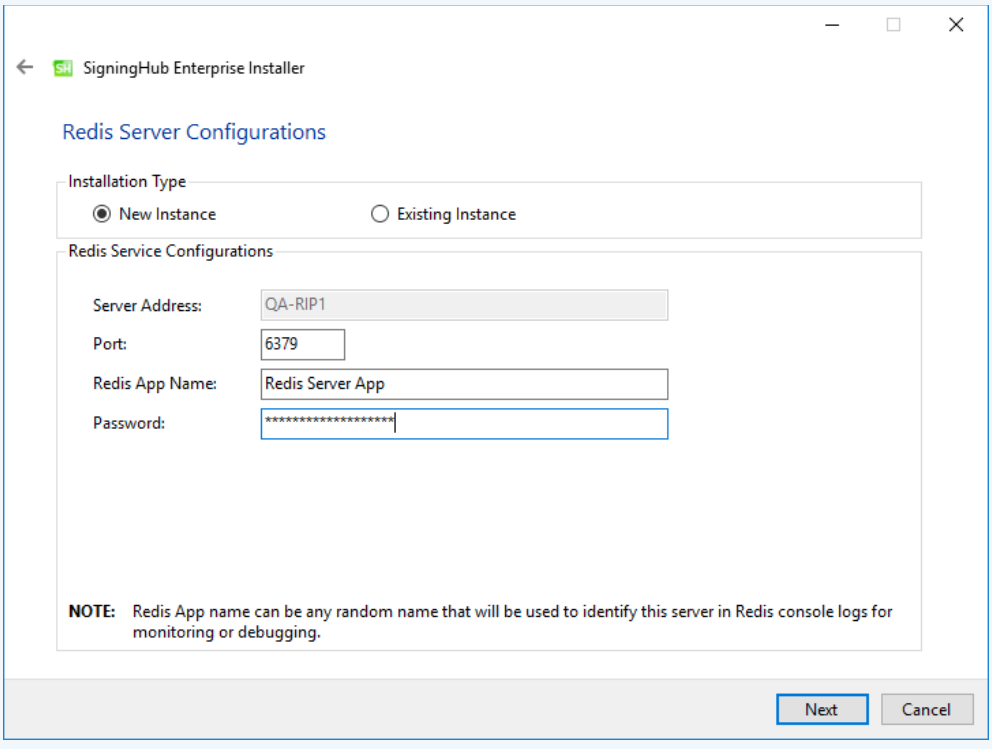
The following table details the configuration options:

Item	Description
Account Name	Account Name of the Azure. Note this must exist prior to the installation.
Account Key	Account Key of the Azure. Note this must exist prior to the installation.

If this is not a File System Storage and you choose the third option to **"Database Storage"** then the following screen is shown:



Click the **Next** button to proceed. The following screen for Redis Server will appear:



Redis is a light weight server, which works as backplane and message broker for SigningHub application over an HTTP/s port. For further information, https://redis.io/

It is bundled within SigningHub package. SigningHub will communicate with the server on that HTTP/s port. It is like SQL Server you need to have an account on Redis server, password is optional.

1) For new instance of Redis Server we create that account with provided details automatically in New Instance screen. It will install a new instance of Redis on the current machine and the configurations will be saved in SigningHub database.

19) If by any chance you have Redis server installed or you want to use Redis server from Azure or Amazon, you need to know the app name, password and port to connect to that instance. In that case, select Existing Instance in the above screen.

On the Redis Server Configurations screen you can either choose a **"New Instance"** or **"Existing Instance"** option.

If this is a new instance, then use the first option i.e. "**New Instance**" and provide the appropriate Redis server configurations.

The information displayed above is an example and you should configure the relevant settings for your own environment.

The following table details the configuration options:

Item	Description
Server Address	Specify the Redis server address. This server is used to send real time on screen notifications for document sharing.
Port	Specify the service port for the Redis server.
Redis App Name	Specify the name of Redis App. This can be any random name that will be used to identify this server in Redis console logs for monitoring or debugging.
Password	Specify the password to authenticate the Redis server.

This is sample text for adding special notes in the document

Redis can enforce password-based security to save or read the key value pairs from the Redis server. To enable password-based security, follow these instructions:

1) Go to `[SigningHub Installation Directory]/Redis`

20) Run the Redis command line interface by click on “redis-cli” application in administrator mode

21) Run the command **“CONFIG SET requirepass "[password]"”**

22) Sign into SigningHub Administrator account

23) Go to **Configurations>Redis** and change the password in Redis Server Connection String

24) Update the settings and **Restart IIS**

Redis can disable asking for password for saving and reading the key value pairs from the Redis server. To turn off the password, follow these instructions:

1) Go to `[SigningHub Installation Director]/Redis`

25) Run the Redis command line interface by click on “**redis-cli**” application in administrator mode

26) Run the command “**CONFIG SET requirepass ""**”

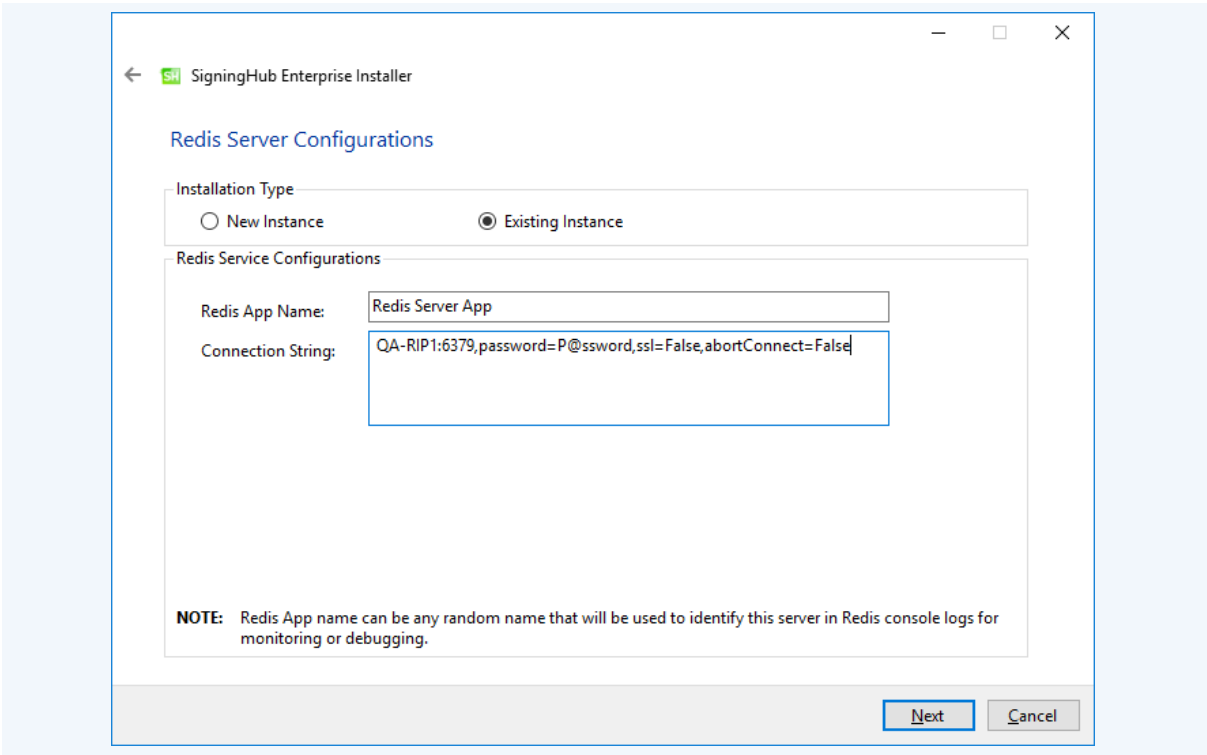
27) Sign into SigningHub Administartor account

28) Go to Configurations>Redis and change the password as empty in **Redis Server Connection String**

29) Update the settings and **Restart IIS**

For Load balanced deployments, only one instance of Redis is needed for SigningHub to work with. Rest of the instances of SignignHub will communicate with Redis using HTTP/s address and Port configured in SigningHub Admin.

If this is not a new instance, and you are choosing the second option i.e. "**Existing Instance"** then the following screen will appear:

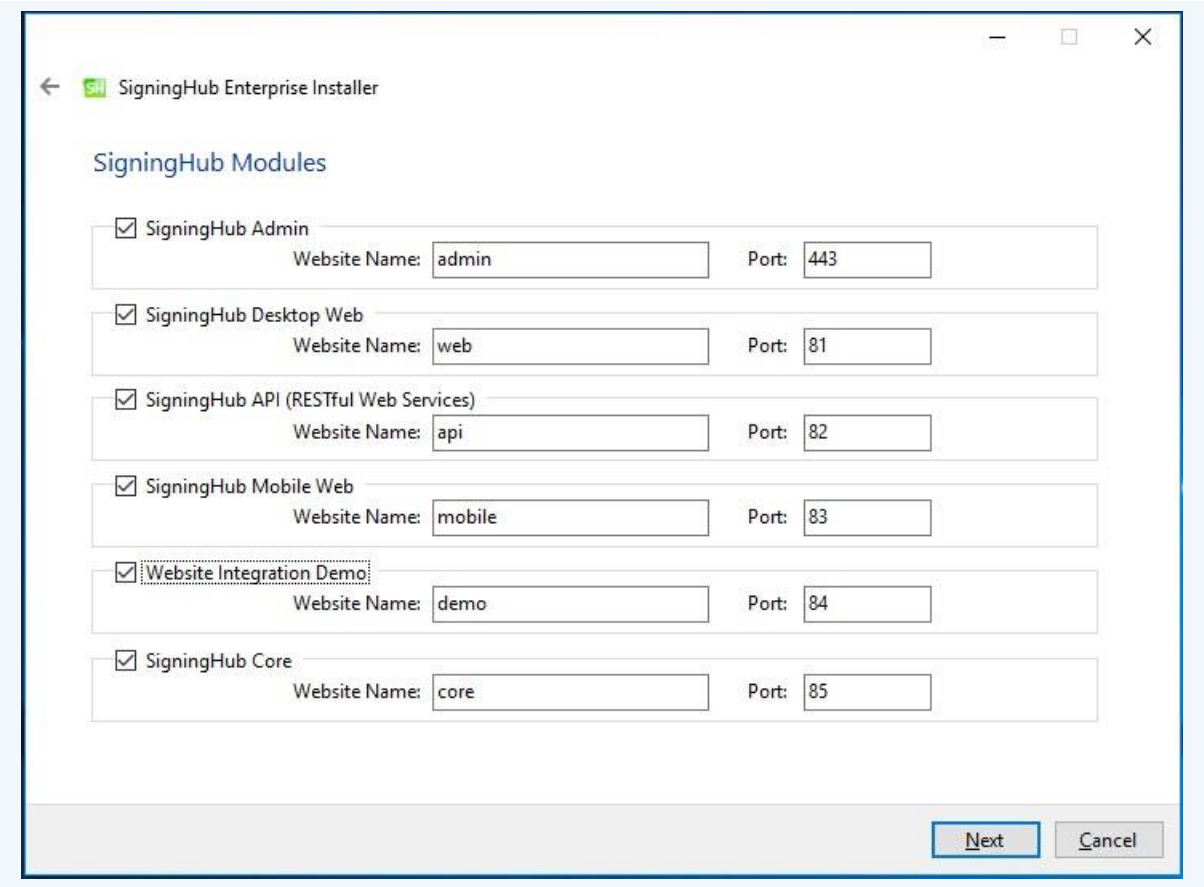


The information displayed above is an example and you should configure the relevant settings for your own environment.

The following table details the configuration options:

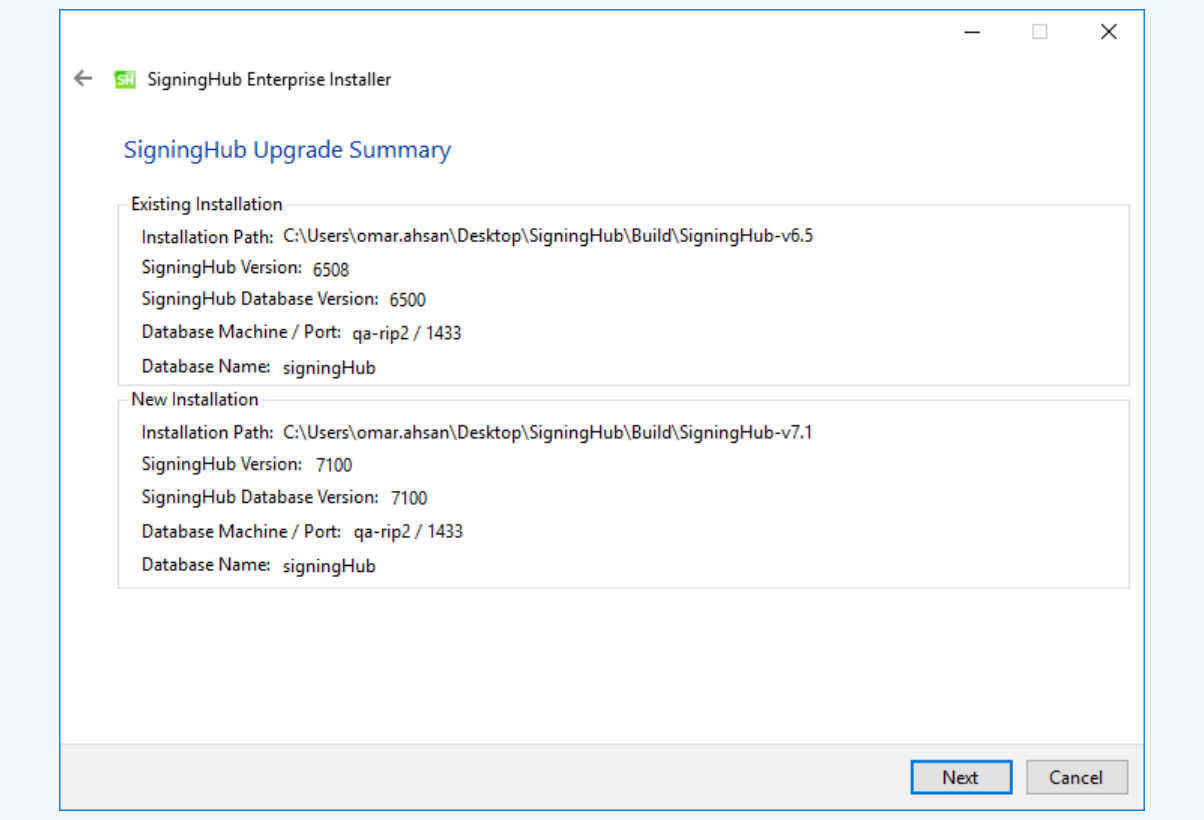
Item	Description
Redis App Name	Specify the name of Redis App. This can be any random name that will be used to identify this server in Redis console logs for monitoring or debugging.
Connection String	The following is a sample connection string for a Redis Server: `"[Redis Server Address]: [port], password=[Redis Server Password],ssl=False,abortConnect=False"`

Click the **Next** button to select specific modules: -

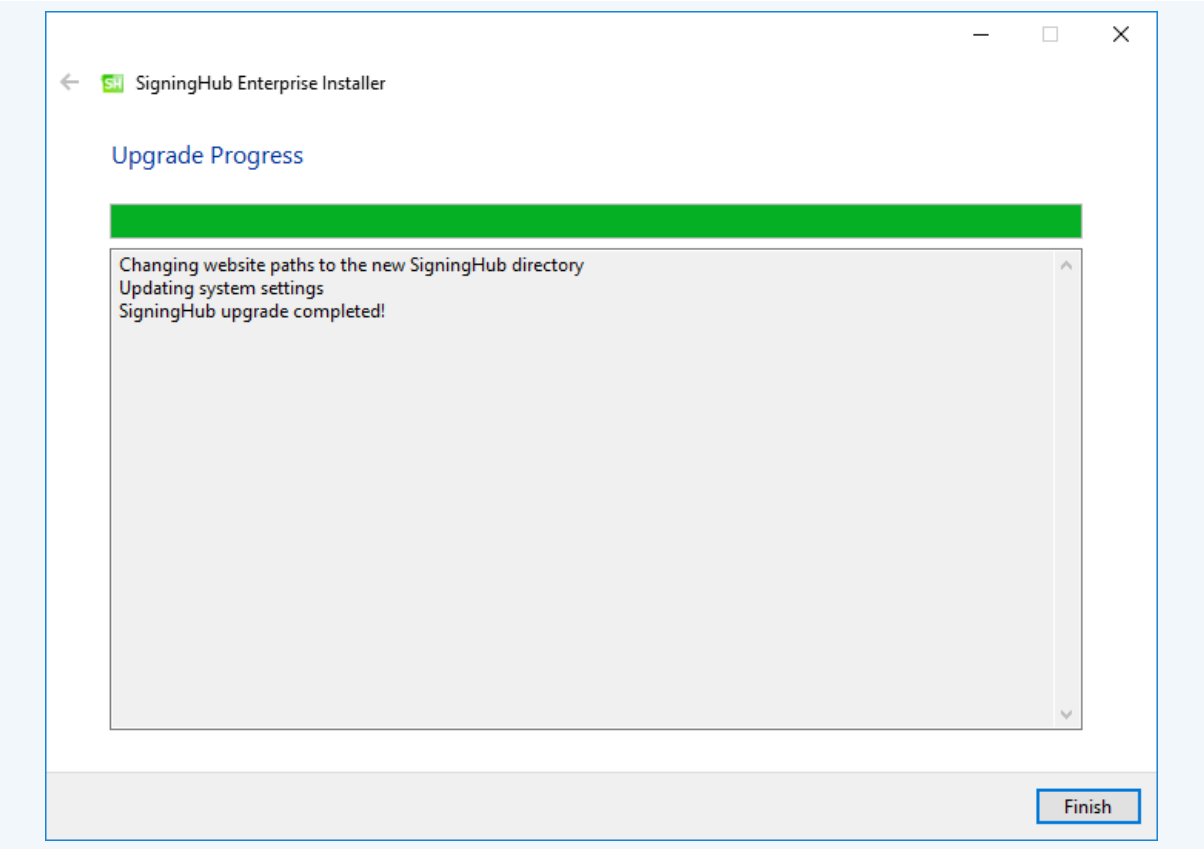


This screen shows a list of all SigningHub Enterprise modules. Components that are already installed are displayed but 'greyed out', while any SigningHub Enterprise module(s) that have not been installed previously can be selected for installation during the upgrade.

Click the **Next** button to see the upgrade summary:



Click the **Next** button to start the upgrade progress.



Click the **Finish** button to complete the SigningHub Enterprise upgrade process.

5.6 Changing Database Credentials for an Existing Installation

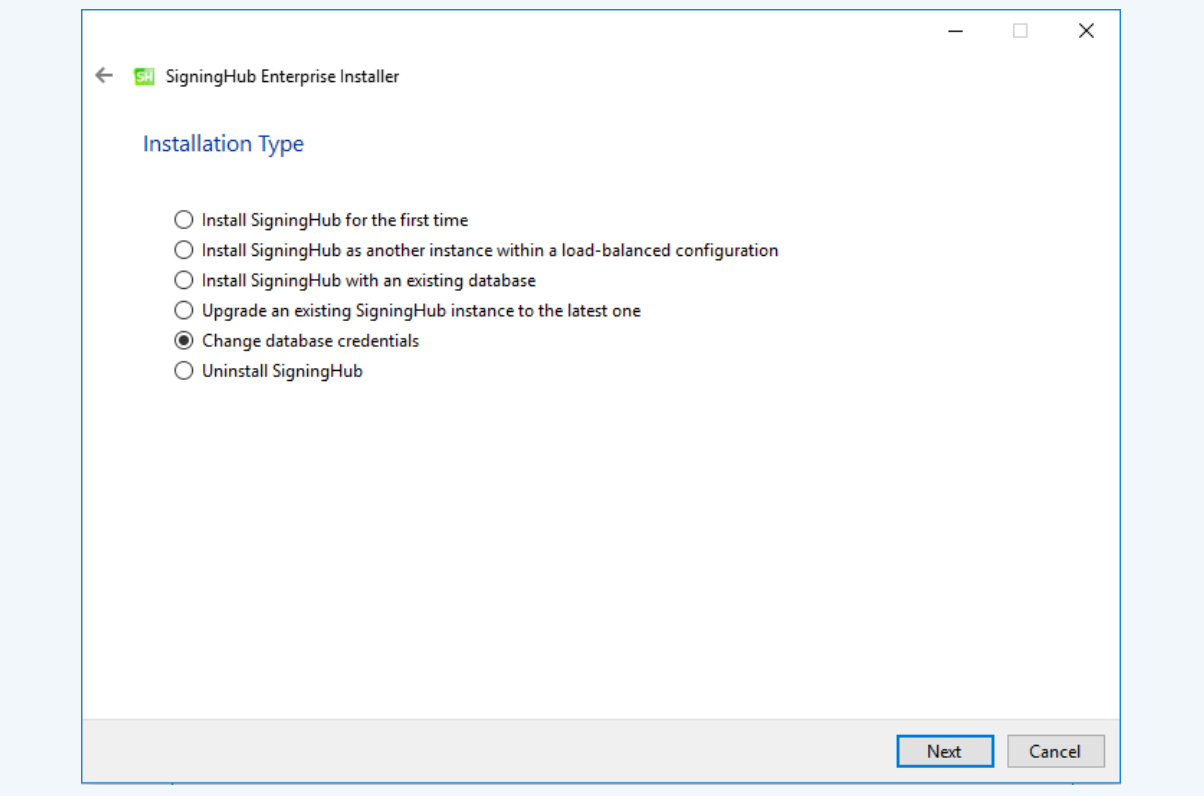
Database credentials stored by SigningHub and ADSS Signing Server are encrypted for security purpose. If you need to make changes in your database server configurations, then these changes must be reflected in the SigningHub and ADSS Signing Server installations for the signing operations to continue.

SigningHub provides an option through the installer to update the following types of database related information:

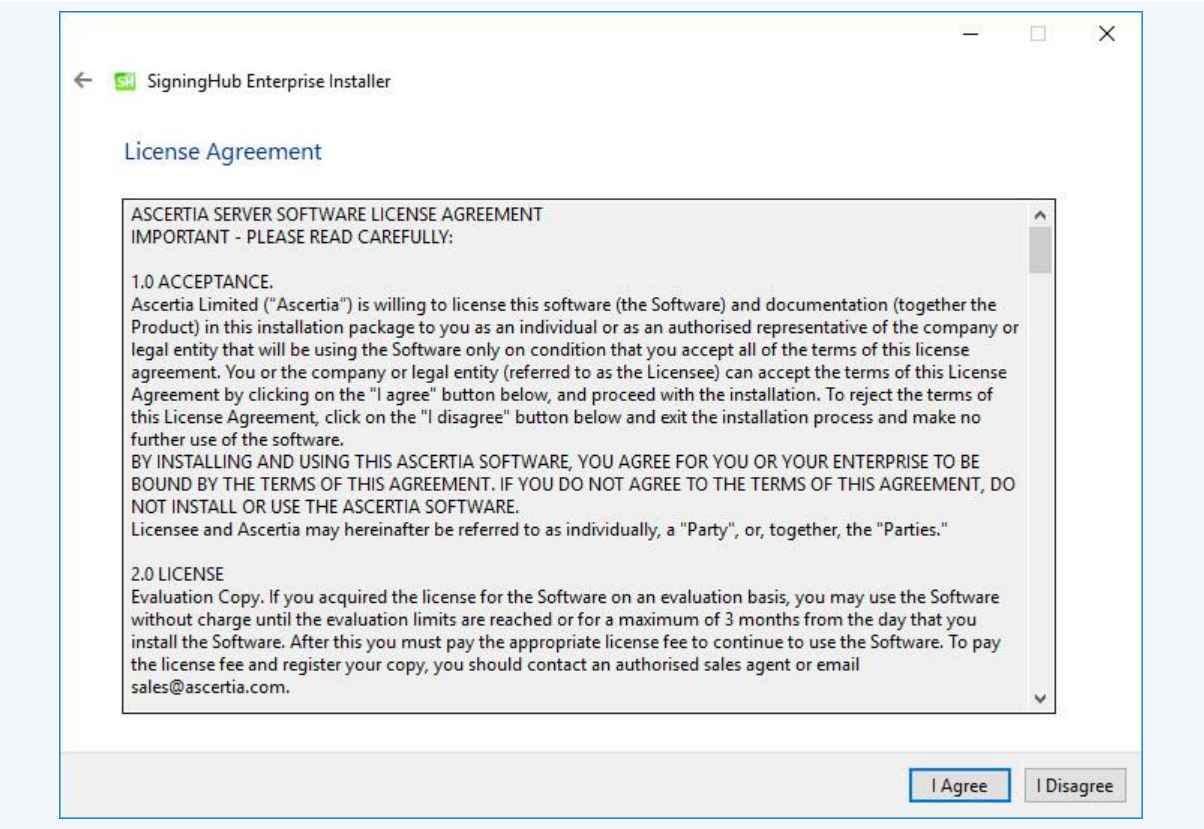
1. Database username and password
2. Database name and/or server (in case if database is restored from production database otherwise you need to install with existing database option)
3. Authentication types (from SQL Server to Windows authentication and vice versa)

Click here for instructions to change ADSS Signing Server database credentials.

Follow the installation wizard until the **Installation Type** screen is shown:

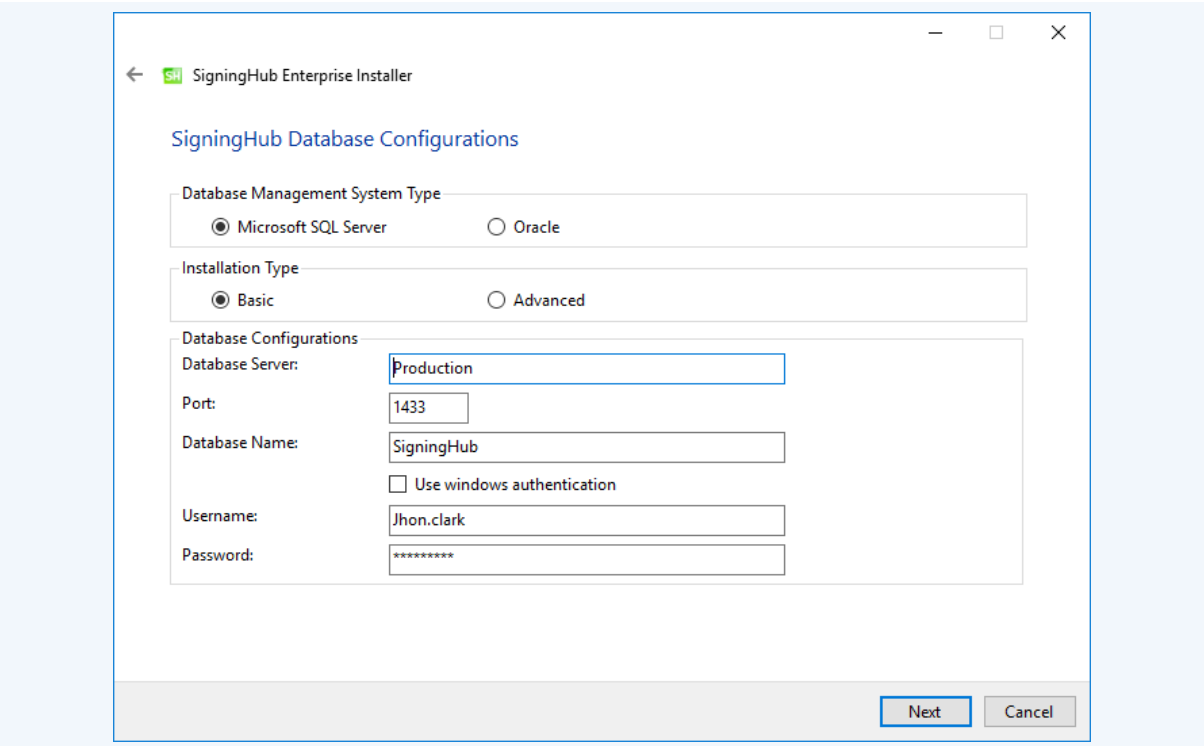


Click the **Next** button to show the License Agreement:



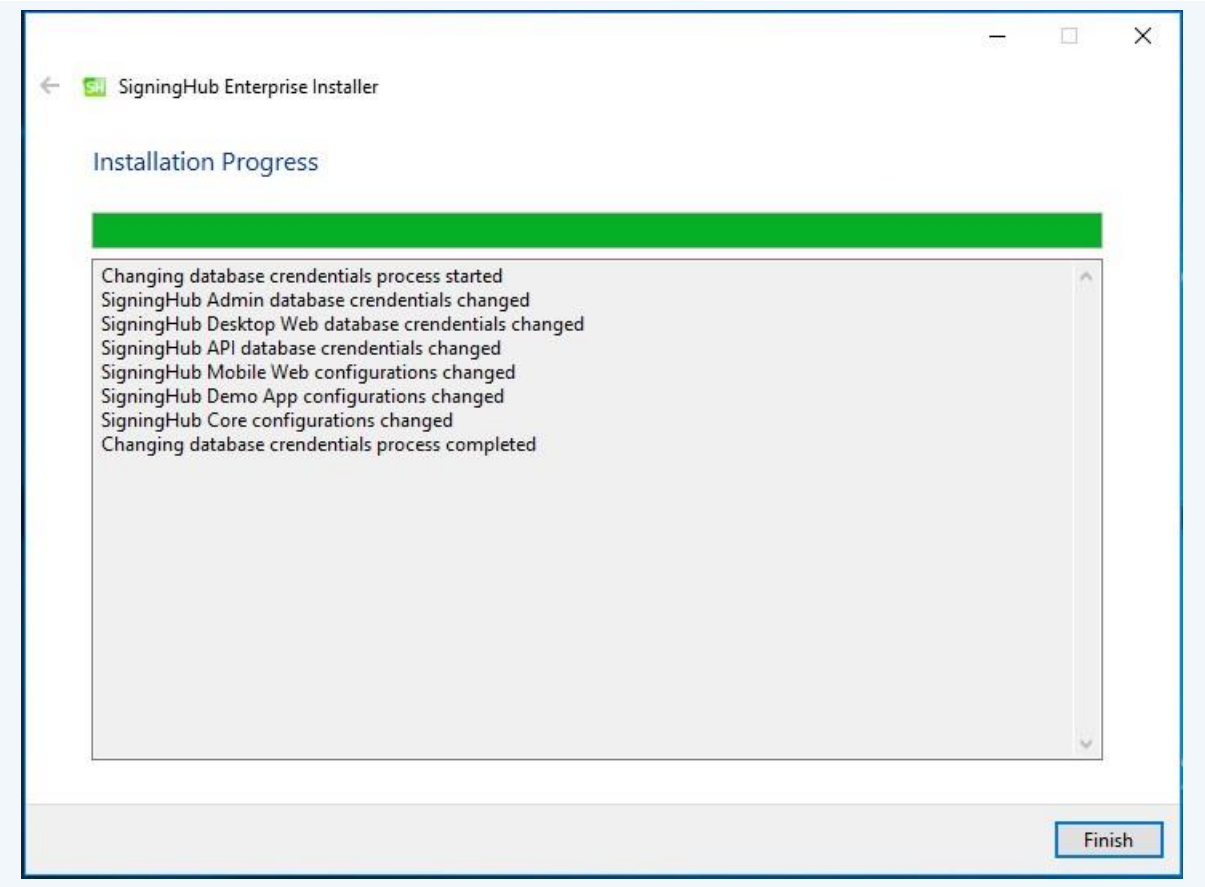
Click the **I Agree** button to proceed.

The following screen to prompt for database details is displayed:



Username and Password must not contain the following characters: [] () , ; ? * ! @ =. These characters are used to either initialize a connection or separate connection values.

Click the **Next** button to update the database configurations.



5.7 Troubleshooting

1. If SigningHub Enterprise Admin module is installed on Windows 2012 R2, then the HTTP 403.16 error code may occur when you access the SigningHub Admin console from web browser.

   Follow these instructions to solve this issue:

   a. Open registry and add the key:

   b. KEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL

   c. Create a new key with **Value Type**: REG_DWORD (32-bit)

   d. Set **Value Name**: ClientAuthTrustMode

   e. **Edit the field and set Value Data**: 2

   If you are interested to know more details about it, browse the following Microsoft KB link: https://support.microsoft.com/en-us/kb/2464556

2. If you receive the HTTP error code 500.19 whilst accessing Admin, Web or API then:

   a. Open IIS Management Console

   b. Go to Application Pools

   c. Select a site and click the **Advanced Setting**

   d. In **General**, make sure that **Enable 32-Bit Applications** is set to **False**

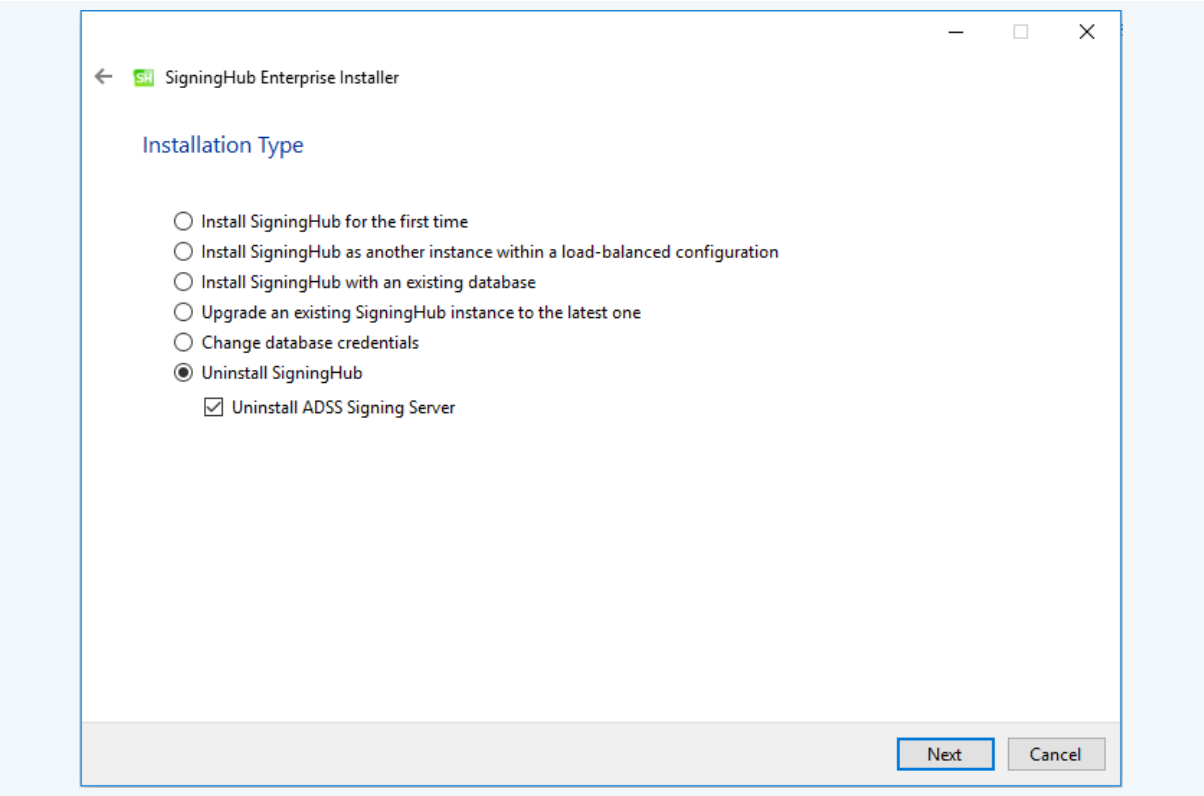
3. If you cannot start ADSS Signing Server from Windows Services panel on Azure, then make sure that you are not starting those services under Windows user that you have created while creating the Azure instance. You must create another Windows user with Administrative rights and start the services under that user.

6 SigningHub Enterprise Uninstallation

Though we will not be pleased to let you go, but sometimes we have to say goodbye. You may uninstall SigningHub Enterprise anytime. For this:

Right click the "**[SigningHub Installation Directory]/setup/install.bat**" file and choose **Run as administrator**.

Follow the installation wizard until the **Installation Type** screen is shown:

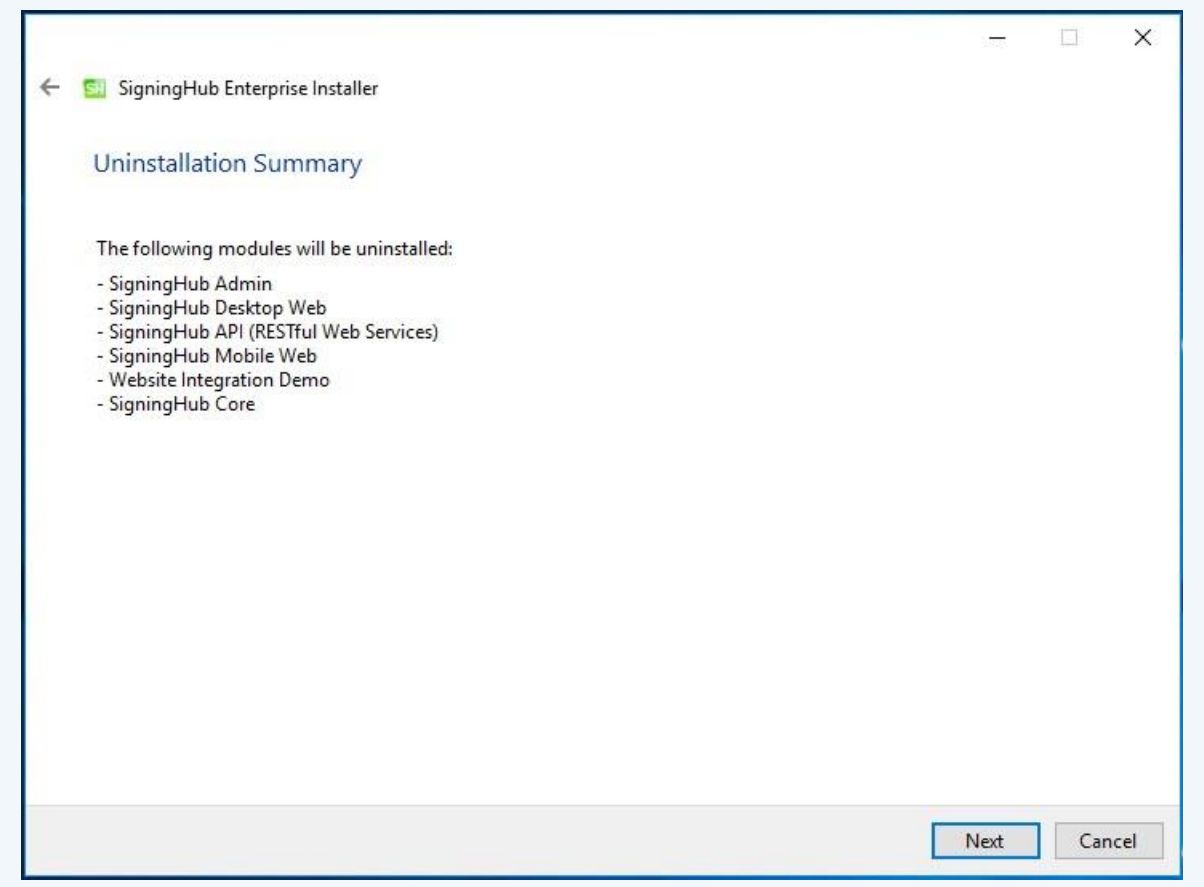


Select "**Uninstall SigningHub**" to remove all websites from IIS mapped, this directory and from the SigningHub Core. If you select the "**Uninstall ADSS Signing Server**" option as well, then ADSS Signing Server will also be uninstalled, provided it is installed from this installation directory. If it is installed from a different installation directory, then you need to uninstall ADSS Signing Server by executing:

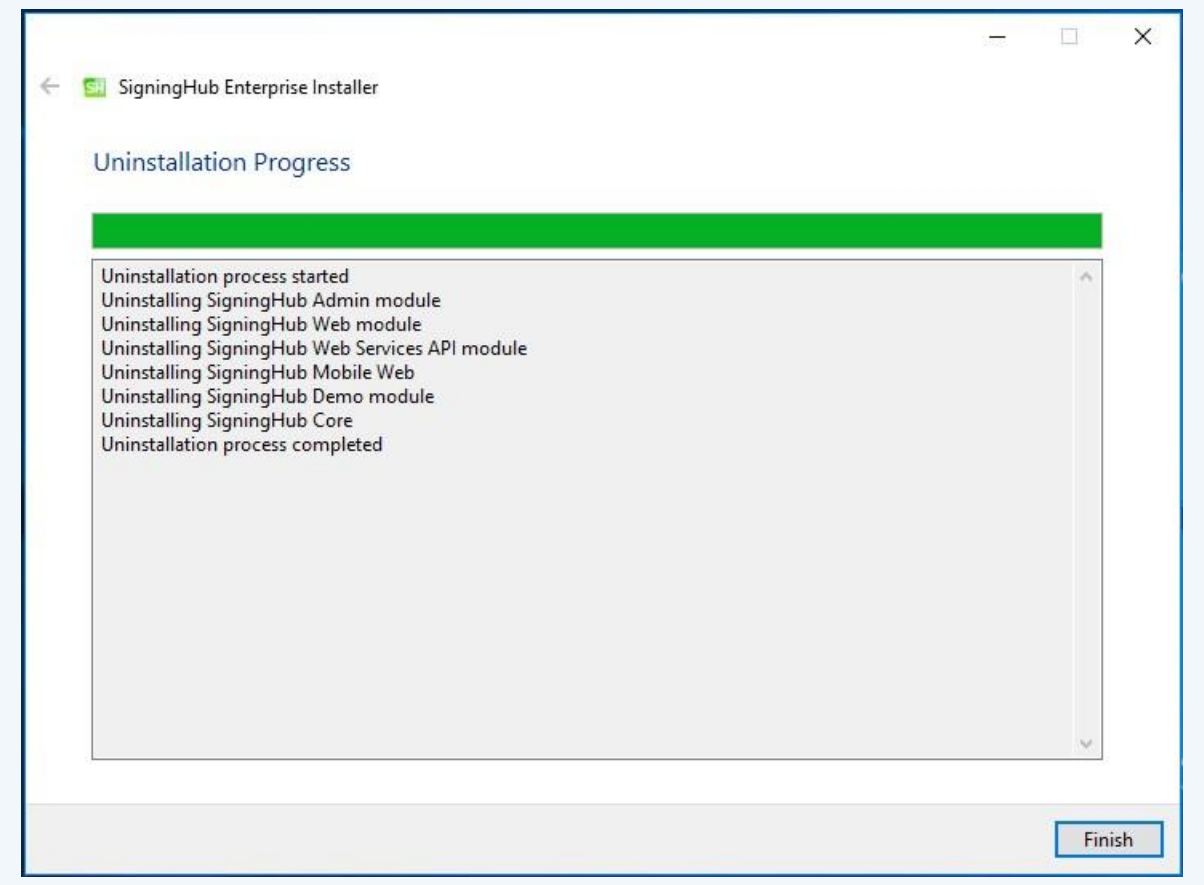
[ADSS-Server-Home]/setup/uninstall.bat script from there.

Click the **Next** button to proceed further.

The following screen is shown:



Click the **Next** button to proceed with the uninstallation.



Click the **Finish** button to complete the process.

Note this procedure does not remove the system database and its respective contents.

If you intent to uninstall ADSS Signing Server, read ADSS Signing Server Installation Guide which is located at: **[ADSS Signing Server installation directory]/docs/**.

Appendix A - Configuring AJP Connector for Local Signing

When you need to sign using local smartcards or USB Tokens, then **ADSS Signing Server Go>Sign Service** is required. This relies on proper configuration of an AJP connector on the proxy server (if used).

Consult the following points to configure AJP services.

A.1 Prepare the Packages

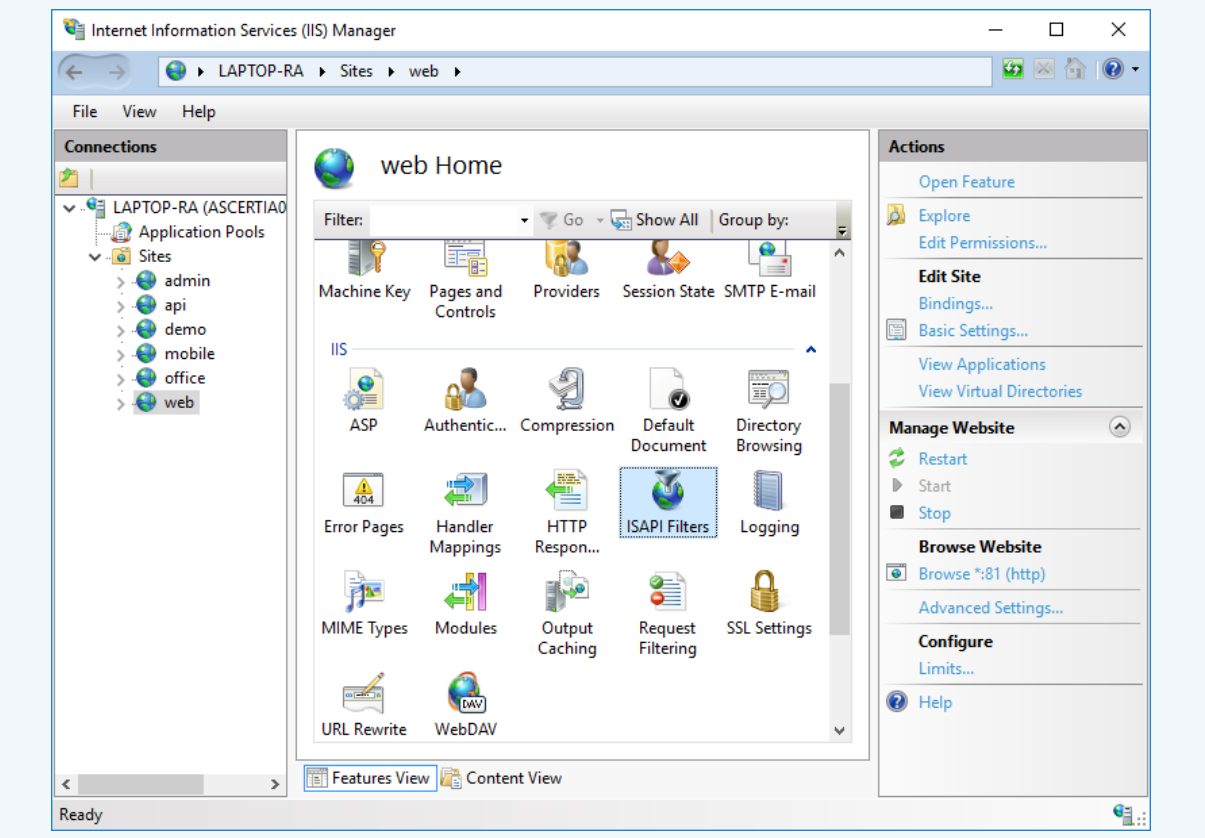
- Retrieve the package **"tomcat_iis_connector.zip"** from the path **"[SigningHub Installation Directory]/tools/adss-server/support**". It contains the configuration files necessary for the ISAPI filter to run and communicate with ADSS Signing Server.
- Extract the zip file and place the contents in a folder at a convenient location on your server machine. The default and recommended location is **C:\tomcat_iis_connector**.
- Copy the **'isapi_redirect.dll**' file from either of the x32 or x64 directories (based on your hardware architecture on the server machine). These folders are in the **'tomcat_iis_connector'** directory. Place this dll file at the root level of '**tomcat_iis_connector**' directory.
- If you extracted the AJP Connector to a directory other than the default recommended directory (i.e. C:\tomcat_iis_connector), then edit the '**isapi_redirect.properties**' file and ensure that the `'log_file'`, `'worker_file'`, `'worker_mount_file'` and `'rewrite_rule_file'` properties are pointing to the correct locations.

If your ADSS Signing Server is not running on the same server as SigningHub Enterprise, then edit the '**worker.properties.minimal**' file in the 'conf' directory so that the 'worker.worker1.host' and 'worker.worker2.host' properties point to the IP address or host name of your ADSS Signing Server installation.

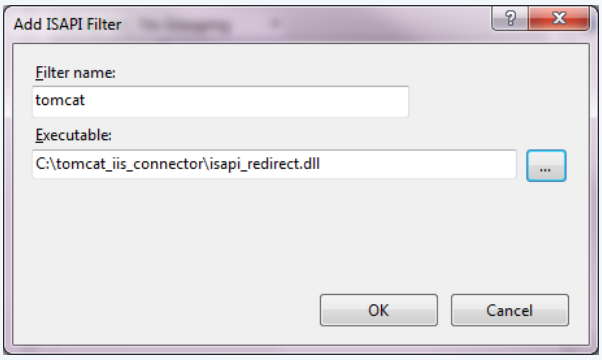
A.2 Add ISAPI Filter for SigningHub Enterprise

Note if ISAPI Filter element (IIS feature and not SigningHub Enterprise) is not installed on your IIS then click here for help.

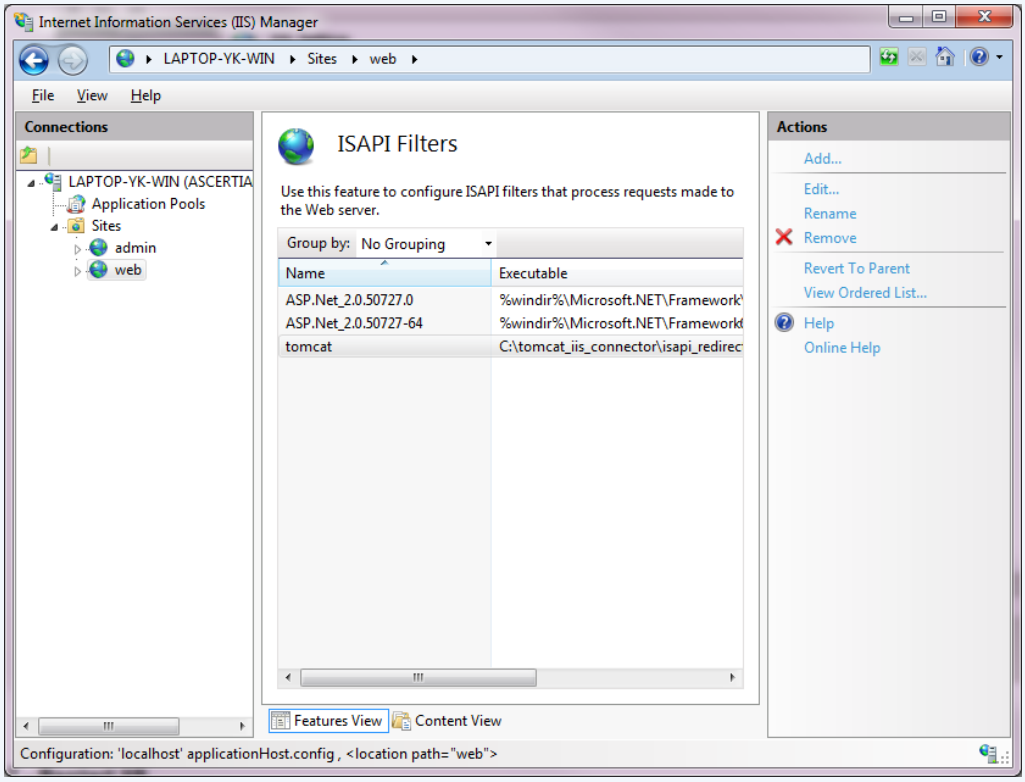
- Open Internet Information Services (IIS) Manager.
- In the '**Connections**' panel, ensure that the SigningHub Enterprise Desktop Web application (i.e. by default named as 'Web') is selected. Now double-click the **'ISAPI Filters**' icon in **'Features View**':



- From the **'Actions'** panel on the right, click **'Add'**. Set the **'Filter name'** to '`tomcat`' and set the **'Executable'** to point to the **'isapi_redirect.dll'** file that you placed in root level of **'tomcat_iis_connector'** in the above step:



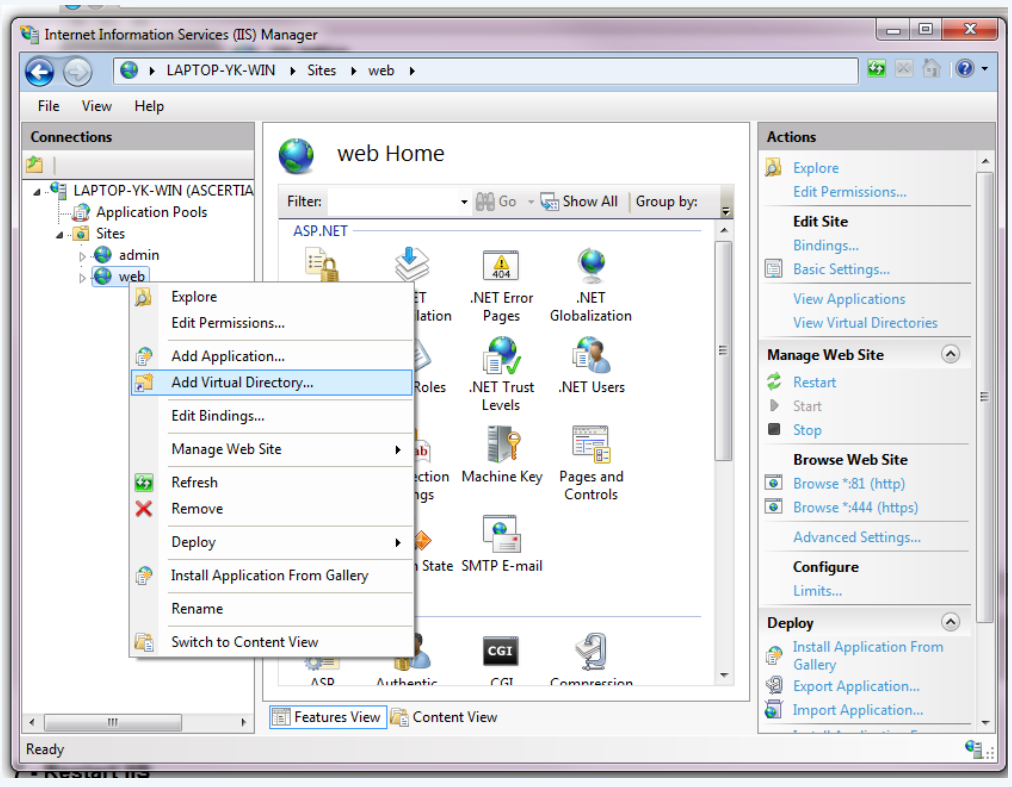
- Click **'OK'**. The new filter should now be listed in the ISAPI Filters list for the website:



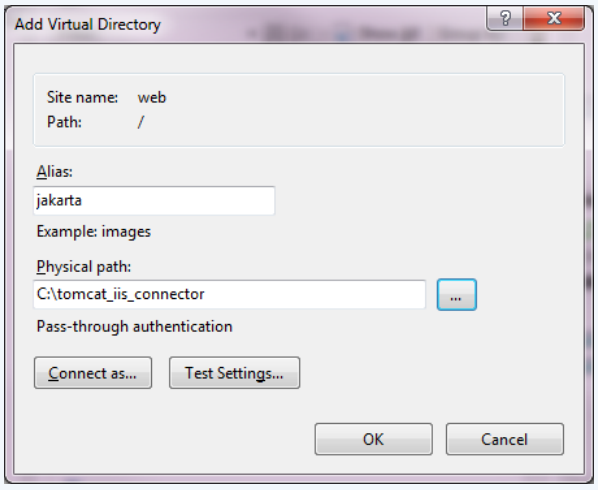
A.3 Add Virtual Directory

Now add a virtual directory in the SigningHub Enterprise Desktop Web application to host the **ISAPI Filter**.

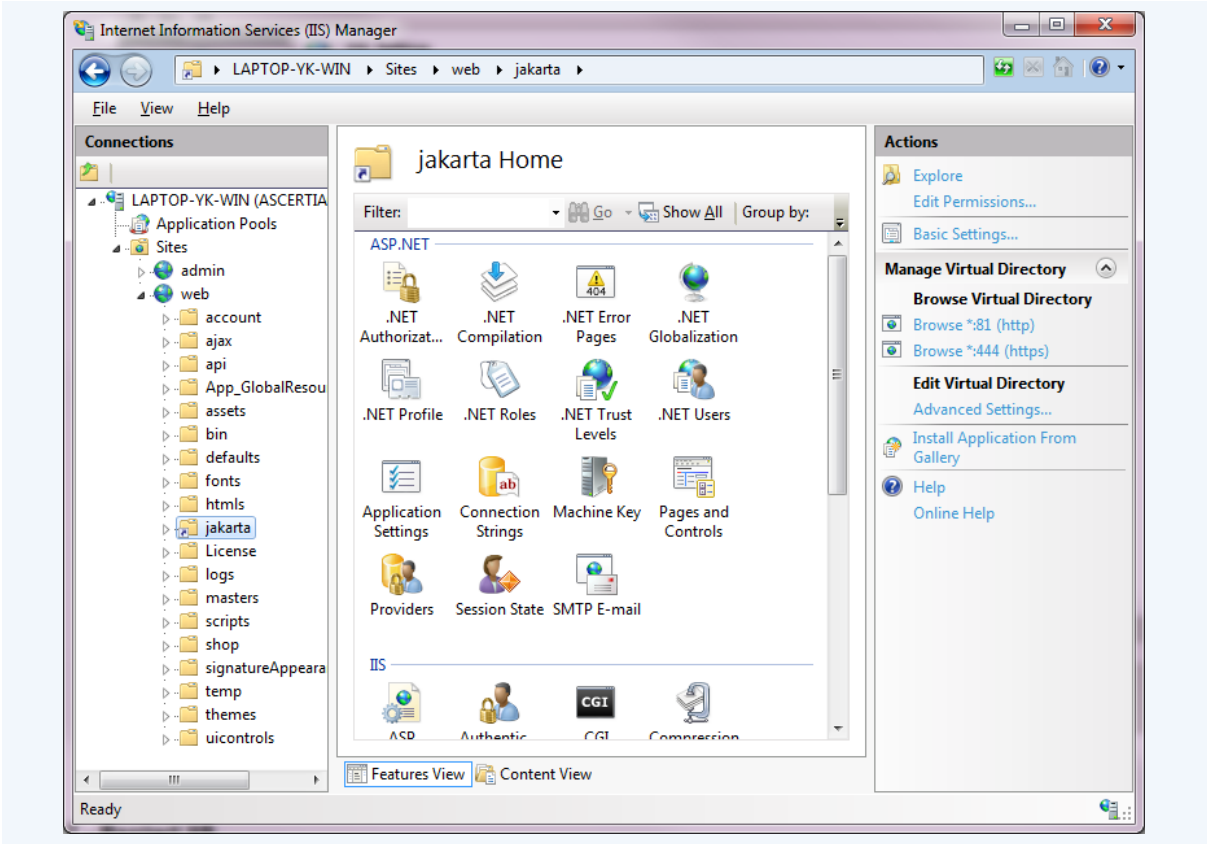
- In the **'Connections'** panel, ensure that the SigningHub Enterprise Desktop Web application (i.e. by default named as 'Web') is selected. Right-click **Web** and select **"Add Virtual Directory"**:



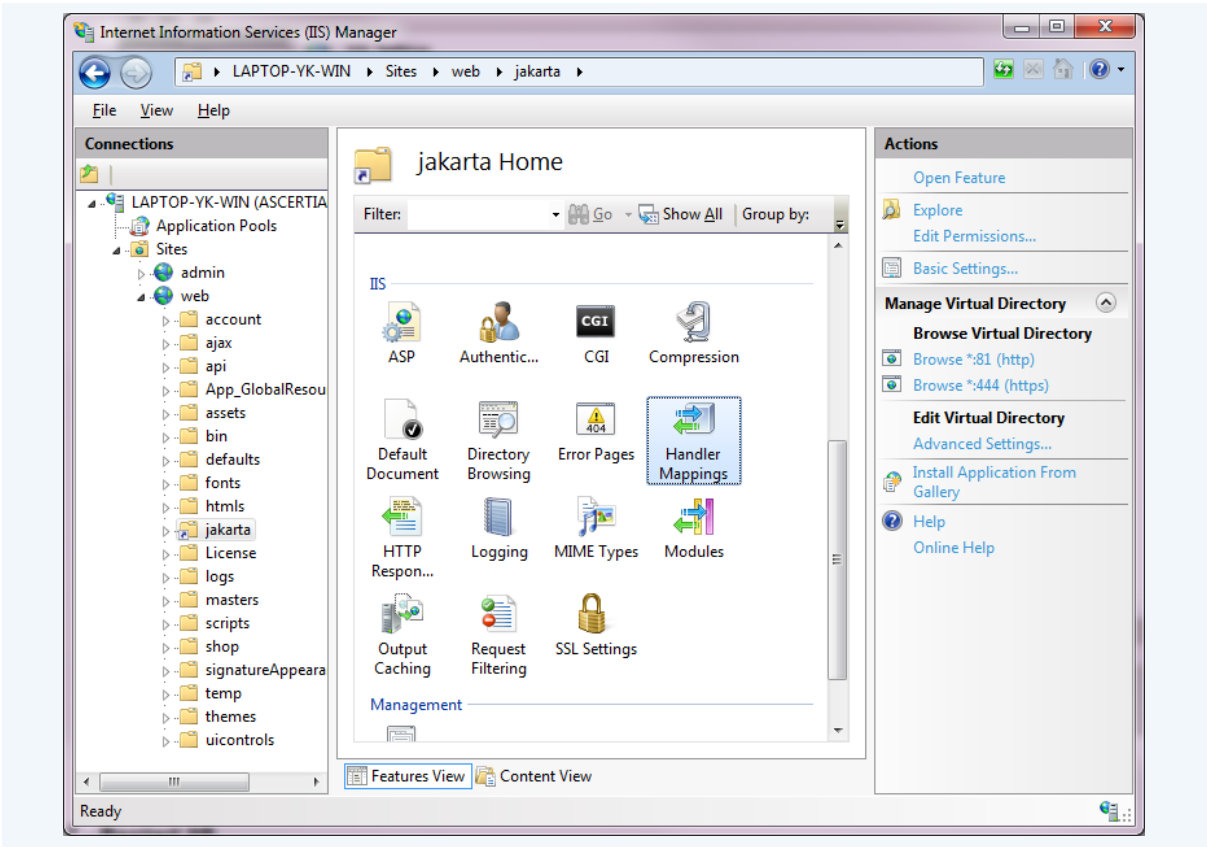
- Set **'Alias'** to `'jakarta'` and **'Physical Path'** to **'tomcat_iis_connector'** directory (e.g. **C:\tomcat_iis_connector**). Click **OK**:



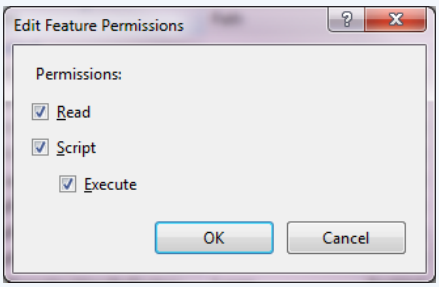
- Verify that the **'jakarta'** virtual directory is now present under the SigningHub Enterprise Desktop Web application:



- Now select the **'jakarta'** virtual directory from the **'Connections'** panel, and double-click the **'Handler Mappings'** icon in the **'Features View'**:



- Click the **'Edit Feature Permissions'** link in the **'Actions'** panel. Ensure that the **'Execute'** option is selected along with the '**Read'** and '**Script'** options:

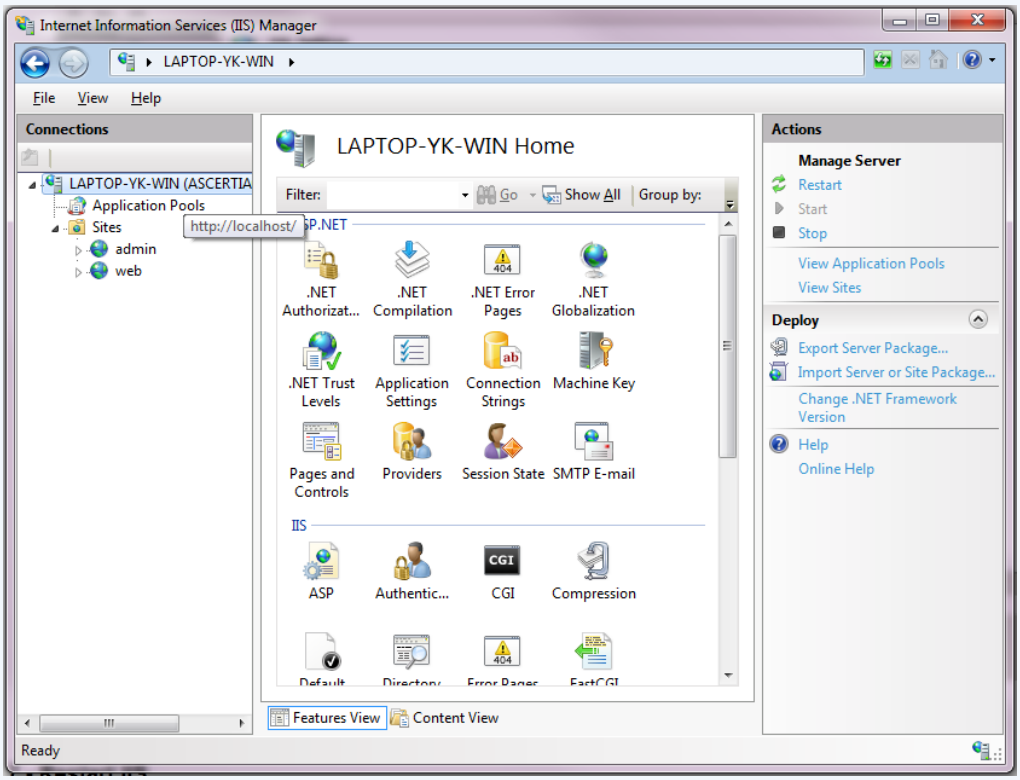


- Click **OK** to close the dialogue message box.

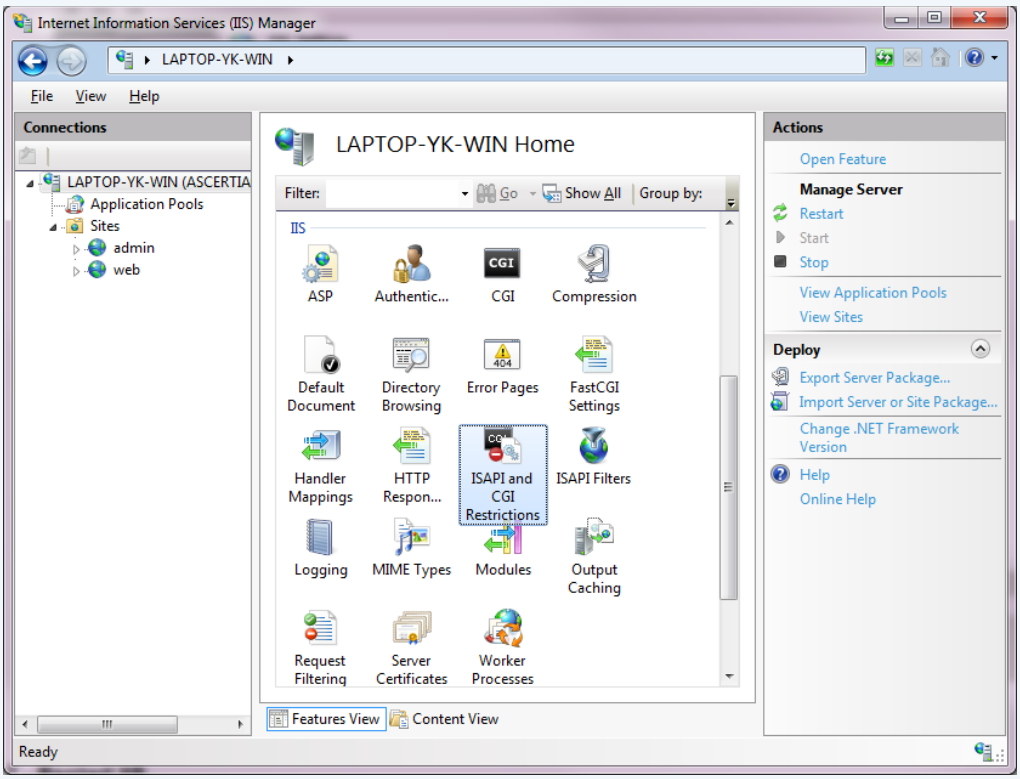
A.4 Register ISAPI Extension

Next, register **'isapi_redirect.dll'** as an authorised ISAPI Extension.

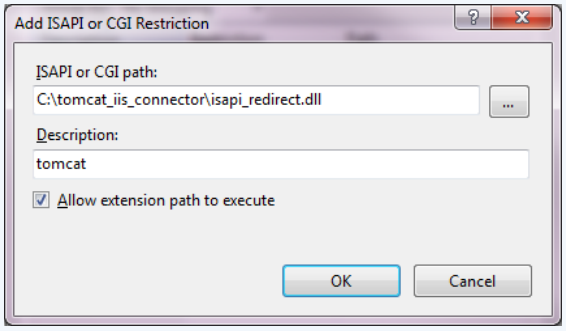
- In the **'Connections'** panel, ensure that the **local IIS Server** instance is selected:



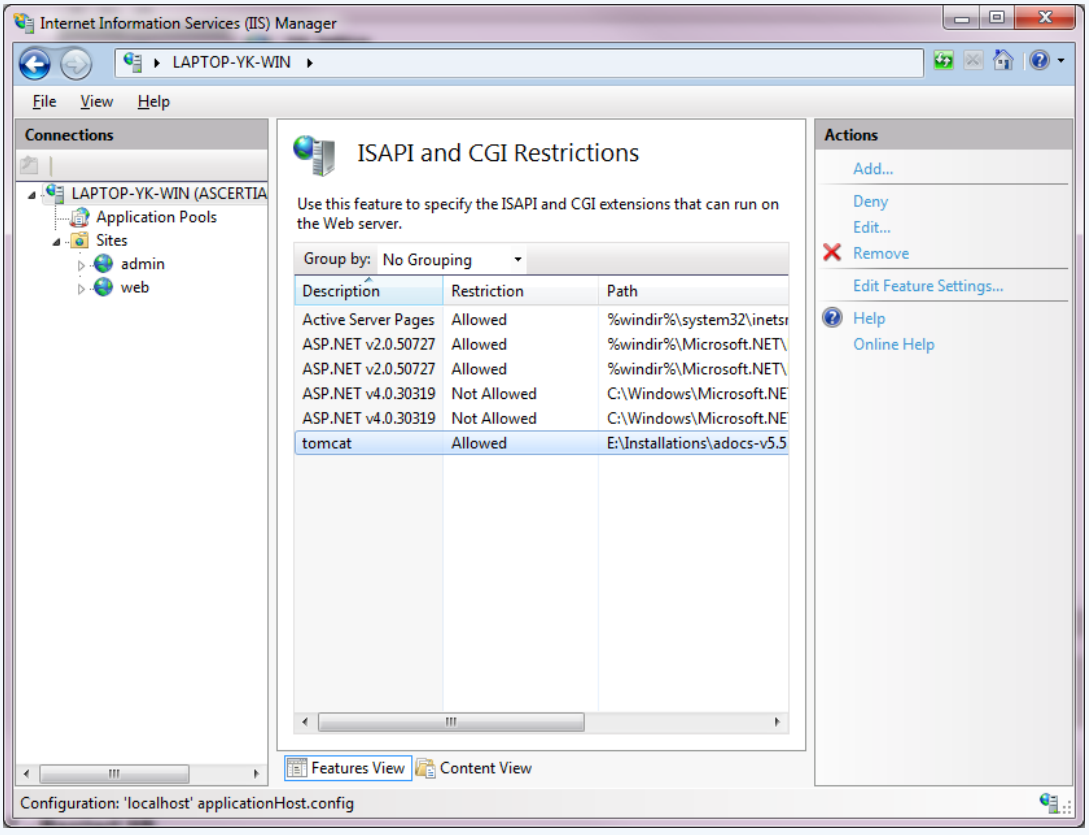
- Double-click the **'ISAPI and CGI Restrictions'** icon in **'Features View'**:



- Click **'Add'** in the **'Actions'** panel, and set the **'ISAPI or CGI path'** to the 'isapi_redirect.dll' you placed at the root level of directory **'tomcat_iis_connector'** in the step above. Set the **'Description'** to `'tomcat'`. Ensure that the **'Allow extension path to execute'** option is selected:



- Click **OK** to close the dialogue message box.
- Verify that the new ISAPI item (i.e. tomcat) is listed in the table with the **'Allowed'** restriction:



A.5 Update ADSS Signing Server & SigningHub Enterprise Configuration

ADSS Signing Server Go>Sign Service must be updated to cater for local signing in SigningHub. To do this follow these instructions: -

- Open the ADSS Signing Server console and login.
- Click on Go>Sign Service menu option.
- Click on the **Service Manager** option from the left-hand menu.
- In the Go>Sign Service Address field, paste the URL that points at the AJP connector server. This is the address of the '**web**' web site that includes the AJP connector configured in IIS, i.e. https://<your_domain>. Note no postfix URL settings is required.
- Click on the **Update** button.
- Now go to Go>Sign Profiles and add/edit the profile. (Note that if sample data is added then Go>Sign Profile **adss:gosign:profile:001** is configured to work with SigningHub)
- Set the **Go>Sign Profile Type** as **PDF Hash**.
- Save the configurations.
- When prompted, click on the **Restart** button.

Next update the ADSS Signing Server connector in SigningHub Enterprise Admin to use the ADSS Signing Server host URL behind AJP connector, e.g. use http://machine-name/ instead of the default URL http://machine-name:8777/. To do this, follow these instructions: -

- Open the SigningHub Enterprise admin console and login.
- Select the **Configurations** menu.
- Select the **Connectors** menu from the right-hand side.
- Choose to edit the ADSS Signing Server connector.
- Update the **Go>Sign Service Address** entry. It should match the entry used directly above in the **Go>Sign Service Address** field, i.e. the address of the '**web**' web site that includes the AJP connector configured in IIS. **Ensure that the final '/' character at the end of the URL is not omitted**.
- Save the changes.
- Choose the **Publish Changes** option found towards the top right of the browser.

In order to deploy the AJP connector with SigningHub Enterprise x64 bit package, it is recommended to explicitly set the "**Enable 32-Bit Application**" option to "**False**" under the advanced properties in IIS, otherwise AJP connector might not function correctly.

To do this go to **Internet Information Services (IIS)**, then Application Pools, and select the Web Site and click **Advanced Settings**. A small window will appear from there you can set the "**Enable 32-Bit Application**" option to "False". Restart the application pool and web site once this is done.

Please note that Local Signing will not be available by using sessionState mode “SQLServer”. However, if users still require performing Local Signing, then please follow these steps to replace sessionState mode with “InProc”:

1) Open the [SigningHub-Installation-Dir]/web/web.config file of all deployments of SigningHub:

2) Replace:

```
“<sessionState mode="SQLServer"
allowCustomSqlDatabase="true"
sqlConnectionString="AdocsEntities" cookieName="SH_ID"
timeout="60" compressionEnabled="true"></sessionState>”
```

With:

```
“<sessionState mode="InProc" timeout="60"
cookieName="SH_ID" />”
```

Restart IIS from the **'Connections'** panel. Right-click on the IIS local instance and **Restart the IIS Server**.

Appendix B - Securing SigningHub Desktop Web

B.1 Securing cookies

To secure SigningHub cookies, make sure following tag is set to true, under **[SigningHub-Installation-Dir]/web/web.config** file:

```
<httpCookies httpOnlyCookies="true" requireSSL="true"/>
```

Note that it should only be enabled when SigningHub is configured to run over SSL.

B.2 'X-XSS-Protection' header

The HTTP X-XSS-Protection response header is a feature of Internet Explorer, Chrome and Safari that stops pages from loading, when they detect reflected cross-site scripting (XSS) attacks. This header is added by default in **web.config**.

B.3 HTTP Strict Transport Security (HSTS) header

The HTTP Strict-Transport-Security response header (often abbreviated as HSTS) is a security feature that lets a website to tell browsers that it should only be communicated over HTTPS, instead of HTTP.

To enable HSTS, add the following header in HTTP Response Headers of IIS against SigningHub Desktop Web website:

```
Strict-Transport-Security: max-age=<expire-time>; includeSubDomains
```

max-age

The time, in seconds, that a browser should remember that this site is only to be accessed using HTTPS e.g. 31536000.

```
includeSubDomains [Optional]
```

If this **Optional** parameter is specified, this rule applies to all of the site's subdomains as well.

B.4 HTTP Public Key Pinning header

The HTTP Public-Key-Pins response header associates a specific cryptographic public key with a certain web server to decrease the risk of MITM attacks with forged certificates. If one or several keys are pinned and none of them is used by the server, the browser will not accept the response as legitimate, and will not display it. To enable Key Pinning, add the following header in HTTP Response Headers of IIS against SigningHub Desktop Web website:

```
Public-Key-Pins: pin-sha256="<pin-value>"; max-age=<expire-time>;
includeSubDomains
```

pin-sha256

The quoted string is the Base64 encoded Subject Public Key Information (SPKI) fingerprint. It is possible to specify multiple pins for different public keys. Some browsers might allow other hashing algorithms than SHA-256 in future.

max-age

The time, in seconds, that a browser should remember that this site is only to be accessed using one of the defined keys e.g. 31536000.

includeSubDomains [Optional]

If this **Optional** parameter is specified, this rule applies to all of the site's subdomains as well.

B.5 TLS Fallback SCSV

To work around interoperability problems with legacy servers, many TLS client implementations do not rely on the TLS protocol version negotiation mechanism alone. They will intentionally reconnect using a downgraded protocol if initial handshake attempts fail. Such clients may fallback to connections in which they announce a version as low as TLS 1.0 (or even its predecessor, Secure Socket Layer (SSL) 3.0) as the highest supported version. To avoid the TLS Fallback SCSV attacks, it is recommended to disable all TLS protocols except TLS 1.2. Click here for instructions to disable the weak protocols.

B.6 SSL Medium Strength Cipher Suites

SigningHub does not use the Medium Strength Ciphers (> 64-bit and < 112-bit key, or 3DES), so you can disable them to avoid any misuse. Click here for instructions to disable the weak or medium ciphers.

B.7 Hiding Application Errors and Server Information

Printing of an exception in browser, server OS information, application data or version number can be of great value for an attacker. By default this is turned off to help troubleshooting errors, however when deployed in production then this should be turned on. It can be turned on by setting `customErrors="On"` in **[SigningHub-Installation-Dir]/web/web.config**.

B.8 Content Security Policy Header

This header helps to prevent code injection attacks like cross-site scripting and clickjacking by telling the browser which dynamic resources are allowed to load. The value of the Content-Security-Policy header is made up of x segments separated by a semicolon; ***self*** translates to the same origin as the HTML resource. With this minimum configuration, your HTML is allowed to fetch JavaScripts, stylesheets etc. from the same domain that served the HTML referencing of the resources. You won't be able to include external scripts from CDNs and similar. This header is added in **web.config and you need to change the SigningHub URLs accordingly:**

```
<add name="Content-Security-Policy" value="default-src 'self'
https://client.go-sign-desktop.com:8782/gosign-desktop;connect-src 'self'
https://graph.microsoft.com/v1.0/
https://dc.services.visualstudio.com/v2/track https://www.facebook.com/tr/
https://client.go-sign-desktop.com:8782/gosign-desktop
https://graph.microsoft.com/v1.0/drive/items/
https://graph.microsoft.com/v1.0/me/drive/items/ https://client.go-sign-
desktop.com:8782 https://web.signinghub.com/adss/gosign/handler
ws://web.signinghub.com/; child-src 'self' https://docs.google.com/picker
https://client.go-sign-desktop.com:8782/gosign-desktop
https://accounts.google.com https://www.google.com/; script-src 'self'
'unsafe-inline' 'unsafe-eval'
https://www.googleadservices.com/pagead/conversion.js
https://bat.bing.com/bat.js  http://apis.google.com
https://apis.google.com/js/api.js https://docs.google.com/picker
https://js.live.net https://www.google-analytics.com https://client.go-
sign-desktop.com:8782/gosign-desktop https://web.signinghub.com
https://api.taxamo.com/js/v1/taxamo.all.js
https://graph.microsoft.com/v1.0/me/drive/items/
https://www.gstatic.com/recaptcha/api2/ https://bat.bing.com/
https://www.googleadservices.com/; style-src 'self' 'unsafe-inline'; img-
src 'self' * data: blob:;frame-src 'self' *;" />
```

This is sample text for adding special notes in the document

Add the following URLs in connect-src for Belgian eID Card, in addition to the above CSP headers:

```
https://client.localmiddleware.be:20202/version
https://client.localmiddleware.be:20202/status
https://client.localmiddleware.be:20202/events
https://client.localmiddleware.be:20202/session
https://client.localmiddleware.be:20202/eID/signingSession
https://client.localmiddleware.be:20202/eID/id
https://client.localmiddleware.be:20202/eID/nonRepudiationCertif
icate
https://client.localmiddleware.be:20202/eID/citizenCertificate
https://client.localmiddleware.be:20202/eID/rootCertificate
https://client.localmiddleware.be:20202/eID/signRsa
```

Add the following URLs in child-src and the last one in script-src for Stripe, in addition to the above CSP headers:

```
https://api.taxamo.com/
```

```
https://c.taxamo.com/
https://p.taxamo.com/
https://api.taxamo.com/js/v1/taxamo.all.js
```

Add the following URLs in **connect-src** for T1C Signing, in addition to the above CSP headers:

```
https://accapim.t1t.be:443 https://localhost:10443/v2/
```

This configuration lets your web application to load resources and styles from its own domain plus scripts from http://apis.google.com, https://js.live.net, and https://www.google-analytics.com

B.9 Referrer Policy Header

Browsers automatically add the Referrer header, when a user clicks a link on your site. This means that a linked website will be able to see where the users are coming from. While this is a great feature for Analytics, you may have sensitive information in your URLs which you don't want to forward to other domains. To remove the referrer entirely, add the following header in **web.config**:

```
<add name="Referrer-Policy" value=" origin" />
```

B.10 'X-FRAME-OPTIONS' Response Header

To restrict frameable response vulnerability. X-FRAME_OPTIONS can be set to 'DENY' in **web.config**. However, it is not recommended when SigningHub has to be used within IFrame.

```
<add name="X-Frame-Options" value="DENY" />
```

B.11 Cacheable HTTPS Response

To prevent sensitive information to be stored in browsers local cache, set no-cache option by adding following header under **web.config.**

```
<add name="Cache-Control" value="no-cache" />
```

B.12 CAPTCHA Configurations

Google CAPTCHA must be configured in SigningHub application to prevent brute force attack. This can be configured in SigningHub admin console under connectors and has to be set as default Google CAPTCHA under Global Settings.

Appendix C - Securing SigningHub API

C.1 'X-XSS-Protection' header

The HTTP X-XSS-Protection response header is a feature of Internet Explorer, Chrome and Safari that stops pages from loading, when they detect reflected cross-site scripting (XSS) attacks. This header is added by default in **web.config**.

C.2 Content Security Policy Header

This header helps to prevent code injection attacks like cross-site scripting and clickjacking by telling the browser which dynamic resources are allowed to load. The value of the Content-Security-Policy header is made up of x segments separated by a semicolon; ***self*** translates to the same origin as the HTML resource. With this minimum configuration, your HTML is allowed to fetch JavaScripts, stylesheets etc. from the same domain that served the HTML referencing of the resources. You won't be able to include external scripts from CDNs and similar. This header is added in **web.config and you need to change the SigningHub URLs accordingly:**

```
<add name="Content-Security-Policy" value="default-src 'self'
https://client.go-sign-desktop.com:8782/gosign-desktop ;connect-src 'self'
https://dc.services.visualstudio.com/v2/track https://www.facebook.com/tr/
https://client.go-sign-desktop.com:8782/gosign-desktop
https://graph.microsoft.com/v1.0/drive/items/
https://graph.microsoft.com/v1.0/me/drive/items/ https://client.go-sign-
desktop.com:8782 https://web.signinghub.com/adss/gosign/handler
ws://web.signinghub.com/; child-src 'self' https://docs.google.com/picker
https://client.go-sign-desktop.com:8782/gosign-desktop
https://accounts.google.com https://www.google.com/ ; script-src 'self'
'unsafe-inline' 'unsafe-eval'
https://www.googleadservices.com/pagead/conversion.js
https://bat.bing.com/bat.js  http://apis.google.com
https://apis.google.com/js/api.js https://docs.google.com/picker
https://js.live.net https://www.google-analytics.com https://client.go-
sign-desktop.com:8782/gosign-desktop https://web.signinghub.com
https://api.taxamo.com/js/v1/taxamo.all.js
https://graph.microsoft.com/v1.0/me/drive/items/
https://www.gstatic.com/recaptcha/api2/; style-src 'self' 'unsafe-inline';
img-src 'self' * data: blob:;frame-src 'self' *;" />
```

This configuration lets your web application to load resources and styles from its own domain plus scripts from http://apis.google.com, https://js.live.net and https://www.google-analytics.com

C.3 'X-FRAME-OPTIONS' Response Header

To restrict frameable response vulnerability. X-FRAME_OPTIONS can be set to 'DENY' in **web.config**. However, it is not recommended when SigningHub has to be used within IFrame.

```
<add name="X-Frame-Options" value="DENY" />
```

C.4 Cacheable HTTPS Response

To prevent sensitive information to be stored in browsers local cache, set no-cache option by adding following header under **web.config.**

```
<add name="Cache-Control" value="no-cache" />
```

Appendix D - Application Settings

The Application Settings is an <appSettings> tag in the “web.config” file of SigningHub application that includes multiple <add> tags. Each <add> tag is uniquely identified by its “key” attribute.

The setting of these <add> tags affect how the relevant SigningHub features would work. By default, these settings are pre-configured with the installation; however, you may modify them as per your requirements.

The following sections illustrate the features that you can configure in App Settings for Desktop Web, Mobile Web, API and SigningHub Admin.

D.1 Desktop Web

Make the following changes in the “web.config” file to configure Application Settings for SigningHub Desktop Web.

To display Global Sign Logo under system tray:

```
<add key="ShowGlobalSignLogo" value="false" />
```

For the tag with the “ShowGlobalSignLogo” key, set the value to “**True”**.

To perform client-side signing using Applet option:

```
<add key="SupportApplet" value="False" />
```

For the tag with the “SupportApplet” key, set the value to “**True**”. This enables to download the JS file for performing client-side signing using the applet.

To display an error if no RUT value found in the user's settings:

```
<add key="ValidateRUT" value="False" />
```

For the tag with the “ValidateRUT” key, set the value to “**True**” than signing using ADSS Go>Sign or CSC Signing Server allows the National Identity IDs with a certificate to be filtered and checked, else system will work as of today and will let user sign the document without validation.

To add signature policy information while performing signatures:

```
<add key="SignaturePolicyURI" value=""/>
<add key="SignaturePolicyName" value=""/>
```

1) For the tag with the “SignaturePolicyURI” key, set the value to a valid downloadable link of the signature policy document.

2) For the tag with the “SignaturePolicyURI” key, set the value to a name of a signature policy document with extension, which is placed under default directory in SigningHub deployment directory at the following path:

```
[SigningHub Deployment Directory]\default\signaturepolicydocuments
```

There is a workaround needed to be done at ADSS end for verification of EPES signatures as ADSS is unable to download the document from the provided URI in signature due to which the policy document needs to be placed under policy folder at the following path:

```
[ADSS Installation Directory]/service/policy
```

Moreover, OID and path in the 'policy.properties' file must be added at the following path:

```
[ADSS Installation Directory]/service
```

Example:

```
Sample-OID = D:/Deployments/ADSS-Server/service/policy/Sample-Policy-Document.pdf
```

To add signature policy information while performing signature

```
<add key="SignaturePolicyOID" value=""/>
```

For the tag with the “SignaturePolicyOID” key, set the value to the signature policy OID according to the policy document.

To add the '00' IDD prefix (if not already present) in the mobile number:

```
<add key="MODIFY_MOBILE_NUMBER_FOR_SAM" value=""/>
```

For the tag with the “MODIFY_MOBILE_NUMBER_FOR_SAM” key, set the value “**True**” SigningHub work as of today by adding the '00' prefix (if not already present) in the mobile number before sending it to the ADSS Server's SAM.

If it's “**False**”, SigningHub will send the mobile number to the ADSS's SAM without any modifications.

To turn off default certify policy for individual users:

```
<add key="DefaultCertifyForIndividualUsers" value="False" />
```

For the tag with "DefaultCertifyForIndividualUsers" key, set the value "**True**" SigningHub will work as of today and turn on the certify policy by default for individual user. If it's "**False**", the SigningHub will turn off the certify policy.

D.2 Mobile Web

Make the following changes in the “web.config” file to configure Application Settings for SigningHub Mobile Web.

To open SigningHub Application for iOS or Android from Mobile Web Interface:

These are smart banner app settings attributes 'apple-itunes-app', 'google-play-app'. Remove these attributes if a banner is not desired.

```
<add key="apple-itunes-app" value="app-id=1546086577" />
<add key="google-play-app" value="app-id=com.shub779app" />
```

Android app parameters to open from browser.

```
<add key="andriod_intent_filter" value="mobile.shub779app.com" />
<add key="andriod_scheme" value="shub779app" />
<add key="andriod_package" value="com.shub779app" />
<add key="ios_app_url" value="https://t5u47.app.goo.gl/31dv" />
```

D.3 API

Make the following changes in the “web.config” file to configure Application Settings for SigningHub API.

To display an error if no RUT value found in the user's settings:

```
<add key="ValidateRUT" value="False" />
```

For the tag with the “ValidateRUT” key, set the value to “**True**” than signing using ADSS Go>Sign or CSC Signing Server allows the National Identity IDs with a certificate to be filtered and checked, else system will work as of today and will let user sign the document without validation.

To add signature policy information while performing signatures:

```
<add key="SignaturePolicyURI" value=""/>
<add key="SignaturePolicyName" value=""/>
```

1) For the tag with the “SignaturePolicyURI” key, set the value to a valid downloadable link of the signature policy document.
2) For the tag with the “SignaturePolicyURI” key, set the value to a name of a signature policy document with extension, which is placed under default directory in SigningHub deployment directory at the following path:

```
[SigningHub Deployment Directory]\default\signaturepolicydocuments
```

There is a workaround needed to be done at ADSS end for verification of EPES signatures as ADSS is unable to download the document from the provided URI in signature due to which the policy document needs to be placed under policy folder at the following path:

```
[ADSS Installation Directory]/service/policy
```

Moreover, OID and path in the 'policy.properties' file must be added at the following path:

```
[ADSS Installation Directory]/service
```

Example:

```
Sample-OID = D:/Deployments/ADSS-Server/service/policy/Sample-Policy-Document.pdf
```

To add signature policy information while performing signature

```
<add key="SignaturePolicyOID" value=""/>
```

For the tag with the “SignaturePolicyOID” key, set the value to the signature policy OID according to the policy document.

To display an error if no RUT value found in user identity table:

```
<add key="ValidateRUT" value="False" />
```

For the tag with the “ValidateRUT” key, set the value to “True”, else system will work as of today and will let user sign the document.

To add the '00' IDD prefix (if not already present) in the mobile number:

```
<add key="MODIFY_MOBILE_NUMBER_FOR_SAM" value=""/>
```

For the tag with the “MODIFY_MOBILE_NUMBER_FOR_SAM” key, set the value “**True**” SigningHub work as of today by adding the '00' prefix (if not already present) in the mobile number before sending it to the ADSS Server's SAM.

If it's “**False**”, SigningHub will send the mobile number to the ADSS's SAM without any modifications.

To turn off default certify policy for individual users:

```
<add key="DefaultCertifyForIndividualUsers" value="False" />
```

For the tag with "DefaultCertifyForIndividualUsers" key, set the value "**True**" SigningHub will work as of today and turn on the certify policy by default for individual user. If it's "**False**", the SigningHub will turn off the certify policy.

To enable Kerberos authentication:

```
<location path="authenticate/kerberos">
            <system.webServer>
                  <security>
                        <authentication>
                              <anonymousAuthentication enabled="true" />
                              <windowsAuthentication enabled="true" />
                        </authentication>
                  </security>
            </system.webServer>
      </location>
```

For the Kerberos authentication, uncomment the above location tag. SigningHub will work as of today and not allowed Kerberose authentication when the location tag is commented out.

To use SigningHub administrator API:

```
<location path="admin/authenticate/client_cert" overrideMode="Allow">
            <system.webServer>
                  <security>
                        <access sslFlags="SslNegotiateCert" />
                  </security>
```

```
            </system.webServer>
      </location>
```

For the SigningHub Admin API usage, uncomment the above location tag. SigningHub will work as of today and not allowed to use SigningHub Admin APIs when the location tag is commented out.

D.4 Admin

Make the following changes in the “web.config” file to configure Application Settings for SigningHub Admin.

To add the '00' IDD prefix (if not already present) in the mobile number:

```
<add key="MODIFY_MOBILE_NUMBER_FOR_SAM" value=""/>
```

For the tag with the “MODIFY_MOBILE_NUMBER_FOR_SAM” key, set the value “**True**” SigningHub work as of today by adding the '00' prefix (if not already present) in the mobile number before sending it to the ADSS Server's SAM.

If it's “**False**”, SigningHub will send the mobile number to the ADSS's SAM without any modifications.

Appendix E - Securing SigningHub Mobile Web

E.1 Securing cookies

To secure SigningHub cookies, make sure following tag is set to true, under **[SigningHub-Installation-Dir]/mobile/web.config** file:

```
<httpCookies httpOnlyCookies="true" requireSSL="true"/>
```

Note that it should only be enabled when SigningHub is configured to run over SSL.

E.2 'X-XSS-Protection' header

The HTTP X-XSS-Protection response header is a feature of Internet Explorer, Chrome and Safari that stops pages from loading, when they detect reflected cross-site scripting (XSS) attacks. This header is added by default in **web.config**.

E.3 Content Security Policy Header

This header helps to prevent code injection attacks like cross-site scripting and clickjacking by telling the browser which dynamic resources are allowed to load. The value of the Content-Security-Policy header is made up of x segments separated by a semicolon; ***self*** translates to the same origin as the HTML resource. With this minimum configuration, your HTML is allowed to fetch JavaScripts, stylesheets etc. from the same domain that served the HTML referencing of the resources. You won't be able to include external scripts from CDNs and similar. This header is added in **web.config and you need to change the SigningHub URLs accordingly:**

```
<add name="Content-Security-Policy" value="default-src 'self'
https://client.go-sign-desktop.com:8782/gosign-desktop ;connect-src 'self'
https://dc.services.visualstudio.com/v2/track https://www.facebook.com/tr/
https://client.go-sign-desktop.com:8782/gosign-desktop
https://graph.microsoft.com/v1.0/drive/items/
```

```
https://graph.microsoft.com/v1.0/me/drive/items/ https://client.go-sign-
desktop.com:8782 https://web.signinghub.com/adss/gosign/handler
ws://web.signinghub.com/; child-src 'self' https://docs.google.com/picker
https://client.go-sign-desktop.com:8782/gosign-desktop
https://accounts.google.com https://www.google.com/ ; script-src 'self'
'unsafe-inline' 'unsafe-eval'
https://www.googleadservices.com/pagead/conversion.js
https://bat.bing.com/bat.js  http://apis.google.com
https://apis.google.com/js/api.js https://docs.google.com/picker
https://js.live.net https://www.google-analytics.com https://client.go-
sign-desktop.com:8782/gosign-desktop https://web.signinghub.com
https://api.taxamo.com/js/v1/taxamo.all.js
https://graph.microsoft.com/v1.0/me/drive/items/
https://www.gstatic.com/recaptcha/api2/; style-src 'self' 'unsafe-inline';
img-src 'self' * data: blob:;frame-src 'self' *;" />
```

This configuration lets your web application to load resources and styles from its own domain plus scripts from http://apis.google.com, https://js.live.net and https://www.google-analytics.com

E.4 'X-FRAME-OPTIONS' Response Header

To restrict frameable response vulnerability. X-FRAME_OPTIONS can be set to 'DENY' in **web.config**. However, it is not recommended when SigningHub has to be used within IFrame.

```
<add name="X-Frame-Options" value="DENY" />
```

E.5 Cacheable HTTPS Response

To prevent sensitive information to be stored in browsers local cache, set no-cache option by adding following header under **web.config.**

```
<add name="Cache-Control" value="no-cache" />
```

Appendix F - Configuring SigningHub Demo Site

To configure SigningHub Integration Demo to work in your environment, follow these instructions:

1. Log into SigningHub Desktop Web.
2. Set a **Client ID** and generate the **Client Secret** from **Enterprise Settings > Integrations**. Click here for more details.
3. Create a template in **Enterprise Settings > Templates**. Add an enterprise user, add a signature field and save the template. Click here for more details.
4. Upload the same document in **Enterprise Settings > Library**. Click here for more details.
5. Open the **demo\web.config** file and set the parameters as explained below:
    a. **INTEGRATION_HOST_URL** is the address of the SigningHub web services deployment e.g. https://api.signinghub.com

b. **VIEWER_URL** is the SigningHub Desktop Web address of document viewer e.g. https://web.signinghub.com/Integration

c. Set **CLIENT_ID** and **CLIENT_SECRET** that are generated in step 2. These parameters are used to communicate with SigningHub web services.

d. **USER_NAME** / **DESKTOP_USER_NAME** is the email address of the user used in step 1.

e. **USER_PASSWORD** / **DESKTOP_USER_PASSWORD** is the password of the user used in step 1.

f. DEMO_WORKFLOW_TEMPLATE / DESKTOP_DEMO_WORKFLOW_TEMPLATE is the template name created in step 3.

g. **SERVER_SIGN_DOC_ID** and **CLIENT_ID_DOC_ID** is the document ID created in step 4.

Parameters starting with DESKTOP/CLIENT are used for client side signing while the rest ones are used for server-side signing. You can skip the parameters that are not required in your environment.

Appendix G - Proxy Settings in Internet Explorer

To use SigningHub application while using a proxy server, can be configured in IE browser. To configure proxy settings, follow these instructions:

1. Open Internet Explorer.
2. Select *Tools* from top menu bar of your Internet Explorer window.
3. Select *Internet Options.*
4. Select the *Connections* tab.
5. Select *LAN settings*.
6. Tick checkbox under '*Proxy server*' (Address and Port will now turn to be editable).
7. Enter your proxy server name and port number in their respective text boxes.
8. Click on Ok to all opened windows.
9. Relaunch your Internet Explorer and your configurations must be working.

Appendix H - Installing Redis Server

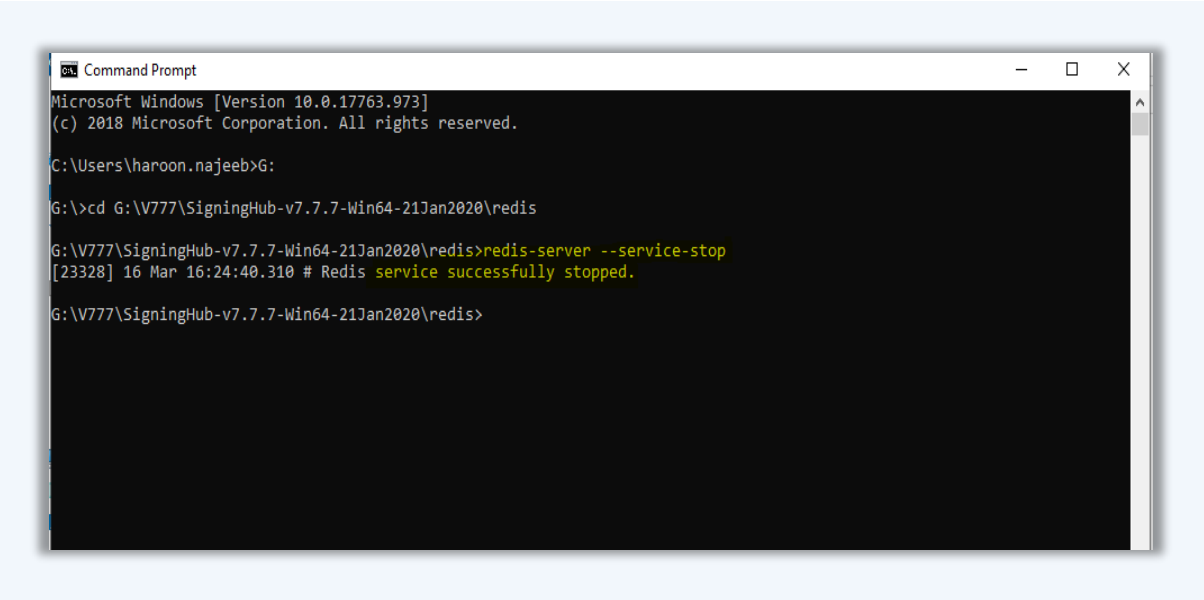
H.1 Issues in Existing Redis Installation

SigningHub installs Redis 3.0 by default with its installer. This distribution of Redis is ported specially to work with Windows operating systems. Following are the issues identified in the ported version of Redis 3.0

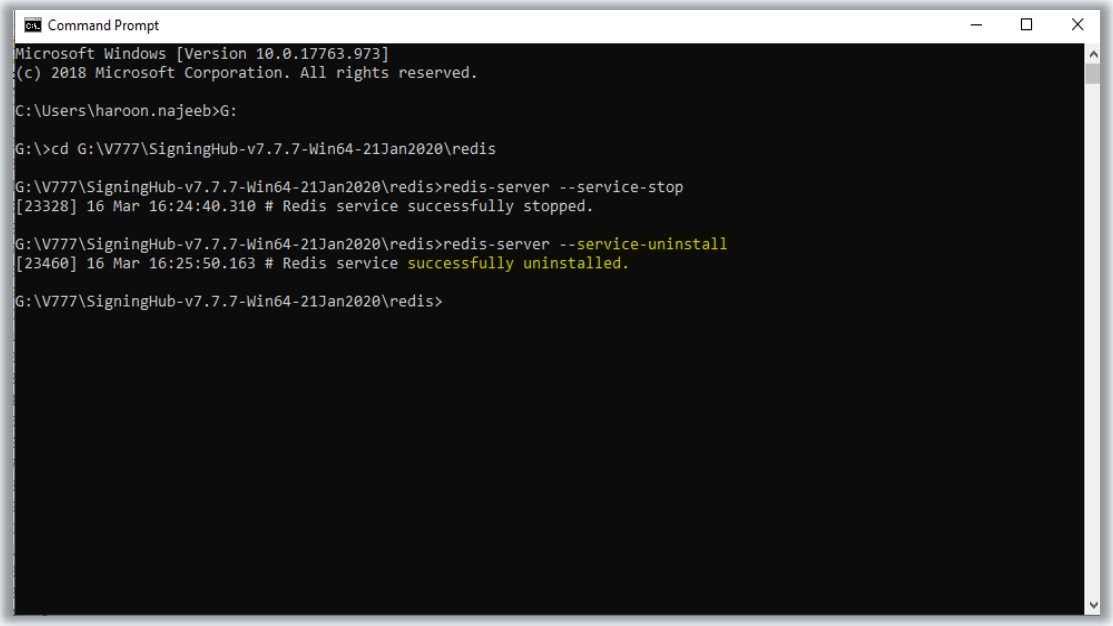
- Vulnerabilities were reported in Redis v3.0 so it is recommended to upgrade on latest stable version.
- This specialized ported version do not support upgrade to any latest version of Redis
- If Redis is already installed for your SigningHub instance then first uninstall the existing version of Redis 3.0 using the following steps.

H.2 Uninstall Redis Service

- Launch windows command prompt as an Administrator and stop the installed Redis services by typing following command. **redis-server - -service-stop**



- Now uninstall the service by using the following command **redis-server - -service-uninstall**



Existing version of Redis 3.0 will be uninstalled from the system. If you have more then one instances of Redis server installed in your environment, uninstall all of these versions using the above steps.

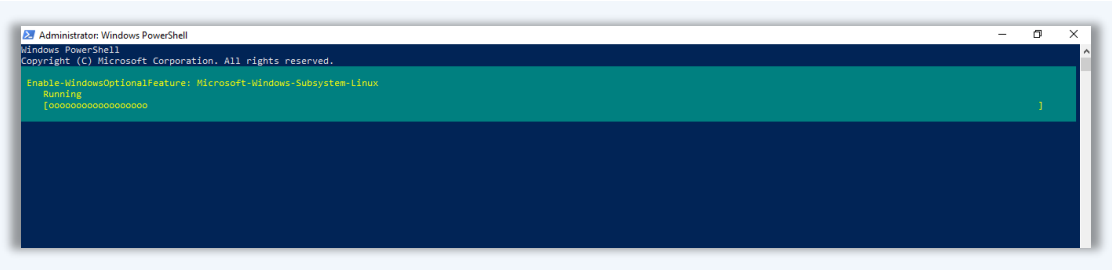
H.3 Installation of Latest Redis Server

Redis server instances are not available for windows operating systems by default. SigningHub application is usually installed on windows operating systems with IIS. To install latest Redis server on windows operating systems, we first need to enable windows sub system for Linux.

Installing Redis on Linux distribution using Windows 10 Bash

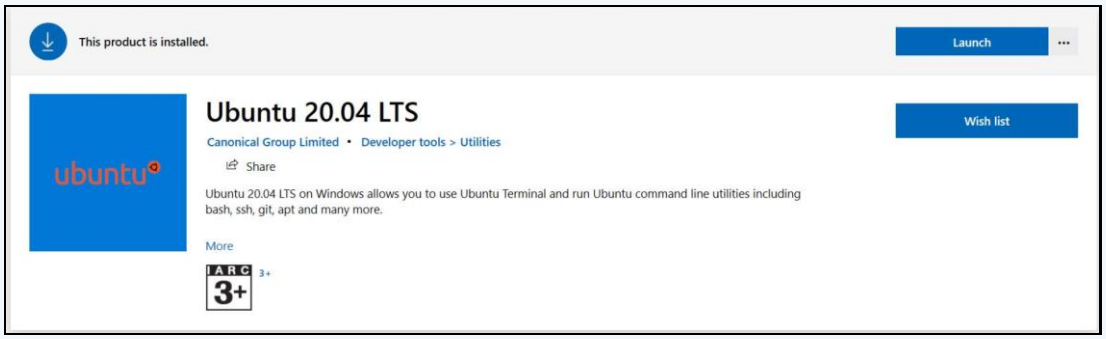
To enable 'Windows Subsystem for Linux', run 'Windows PowerShell' as an administrator

- Run following command to enable Windows Subsystem for Linux (WSL).
  `Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux`

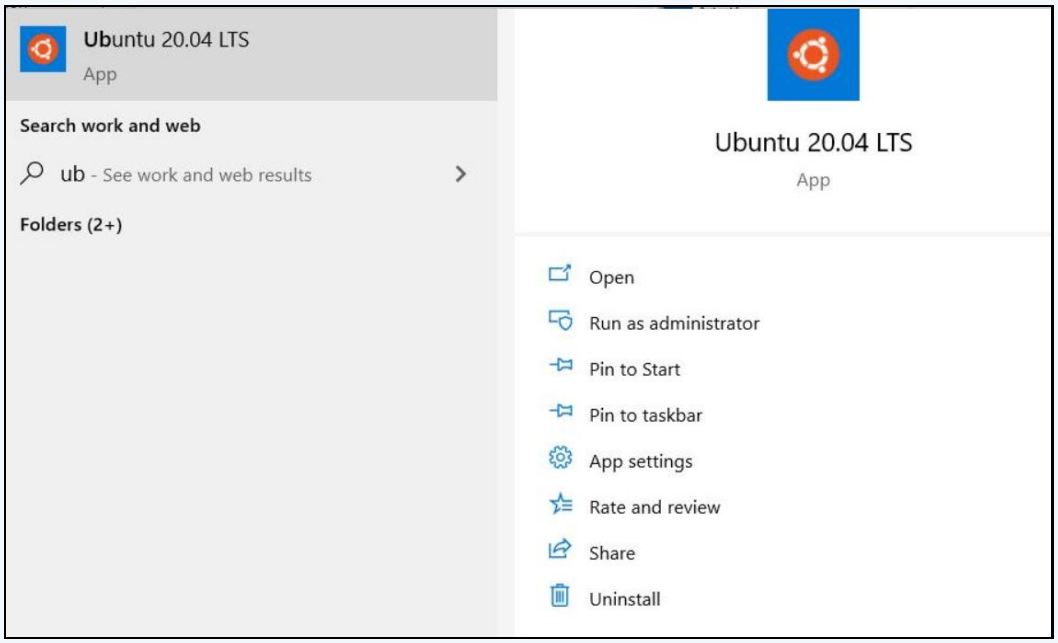


- Reboot Windows after making changes.
- Once system has rebooted, go to the Microsoft Store and search for "**Ubuntu**". Install Ubuntu.

- In current case **Ubuntu 20.04 LTS** has been installed, which have **Redis v6.2.3** (Redis v6.2.3 is currently the latest stable release, for reference on latest Redis version info see following link: https://redis.io/download)



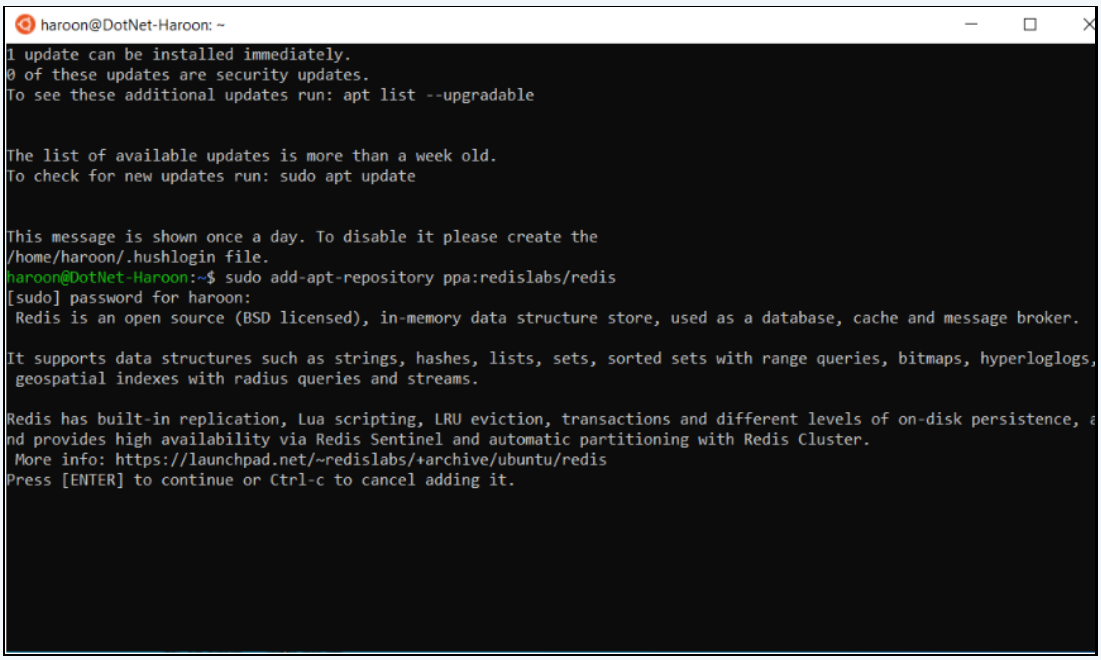
- Once the Ubuntu installed on machine, launch Ubuntu by running it as an Administrator.

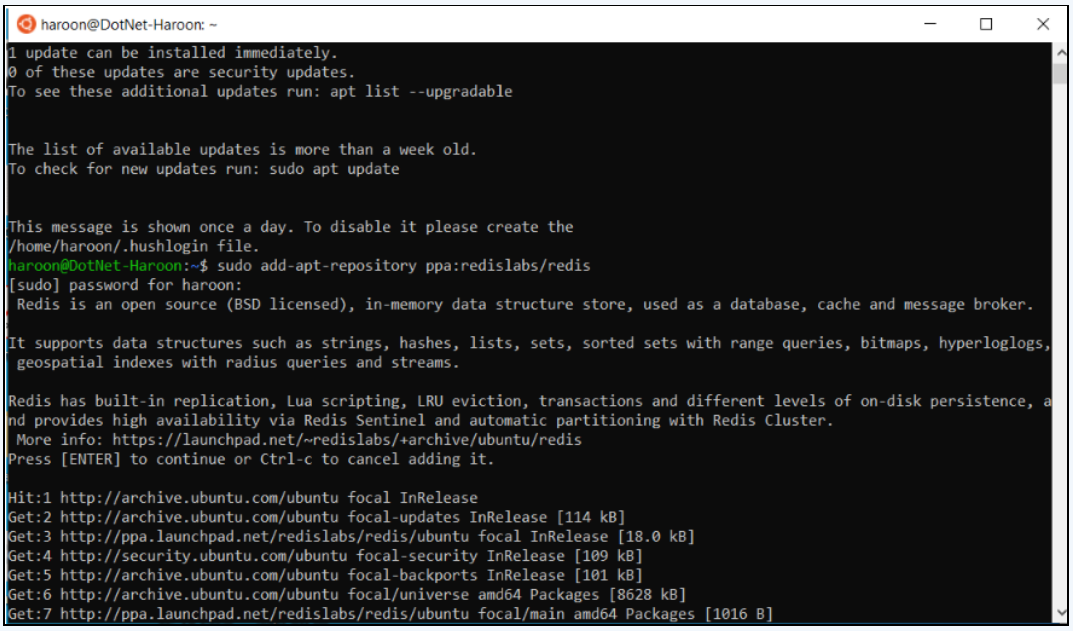


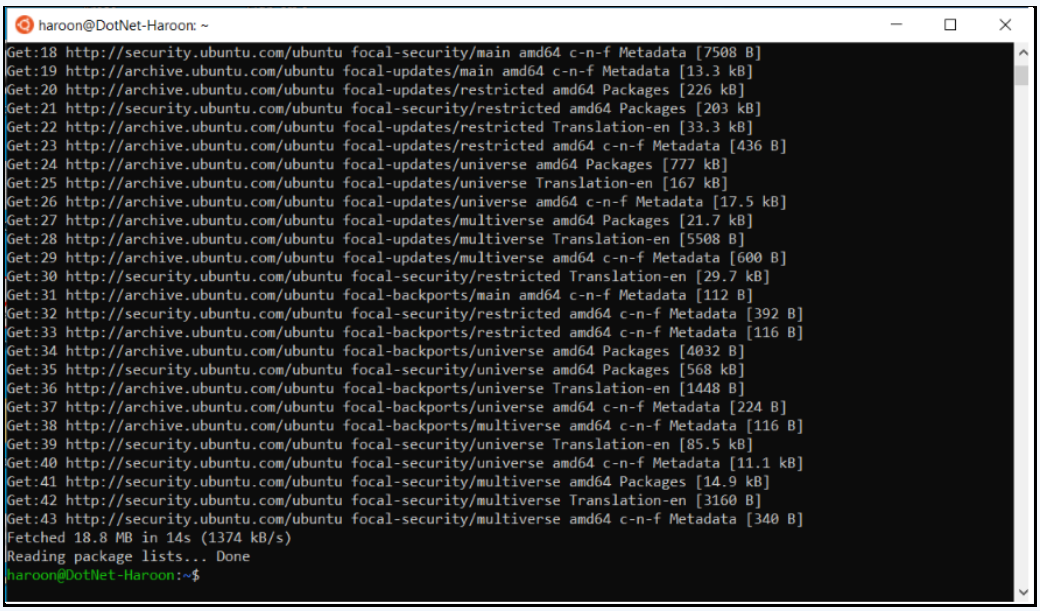
When installing Redis on Linux environment but not using Windows environment, then five steps are not required and Linux commands from here onwards needs to be executed only.

- **Advanced Package Tool (APT)** package manager use to install Redis from the official Ubuntu repositories. If you want to have more recent version of Redis Server, you may opt to use Personal Package Archives (PPA) repository maintained by Redis Development, using following command.

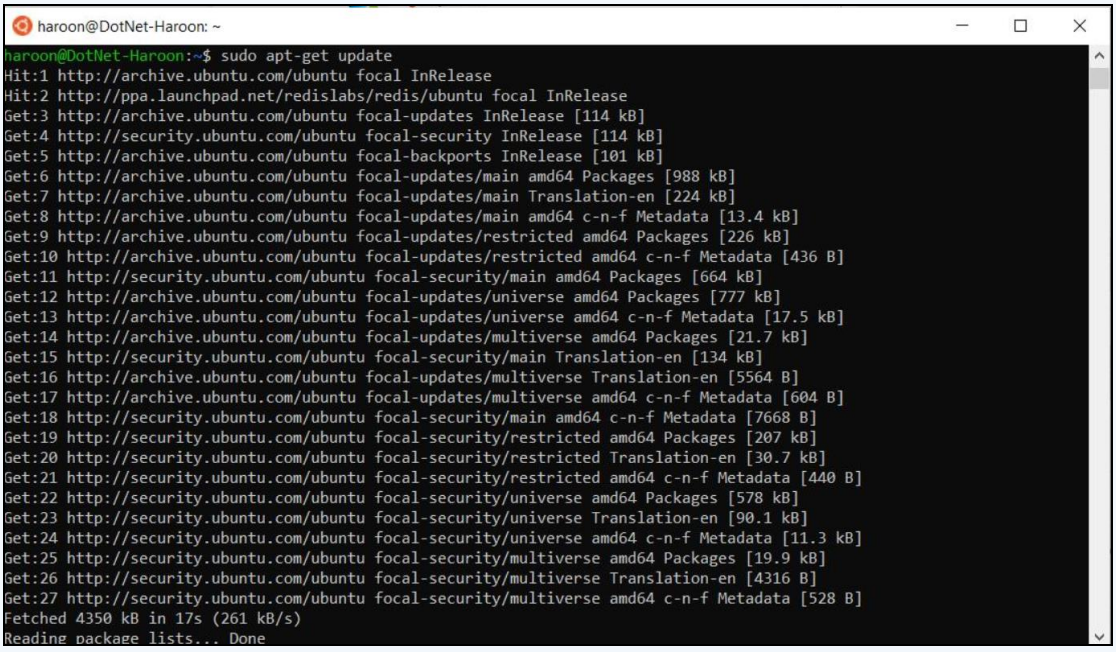
```
sudo add-apt-repository ppa:redislabs/redis
```


haroon@DotNet-Haroon: ~
1 update can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
This message is shown once a day. To disable it please create the
/home/haroon/.hushlogin file.
haroon@DotNet-Haroon:~$ sudo add-apt-repository ppa:redislabs/redis
[sudo] password for haroon:
Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker.
It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs,
geospatial indexes with radius queries and streams.
Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, a
nd provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.
More info: https://launchpad.net/~redislabs/+archive/ubuntu/redis
Press [ENTER] to continue or Ctrl-c to cancel adding it.

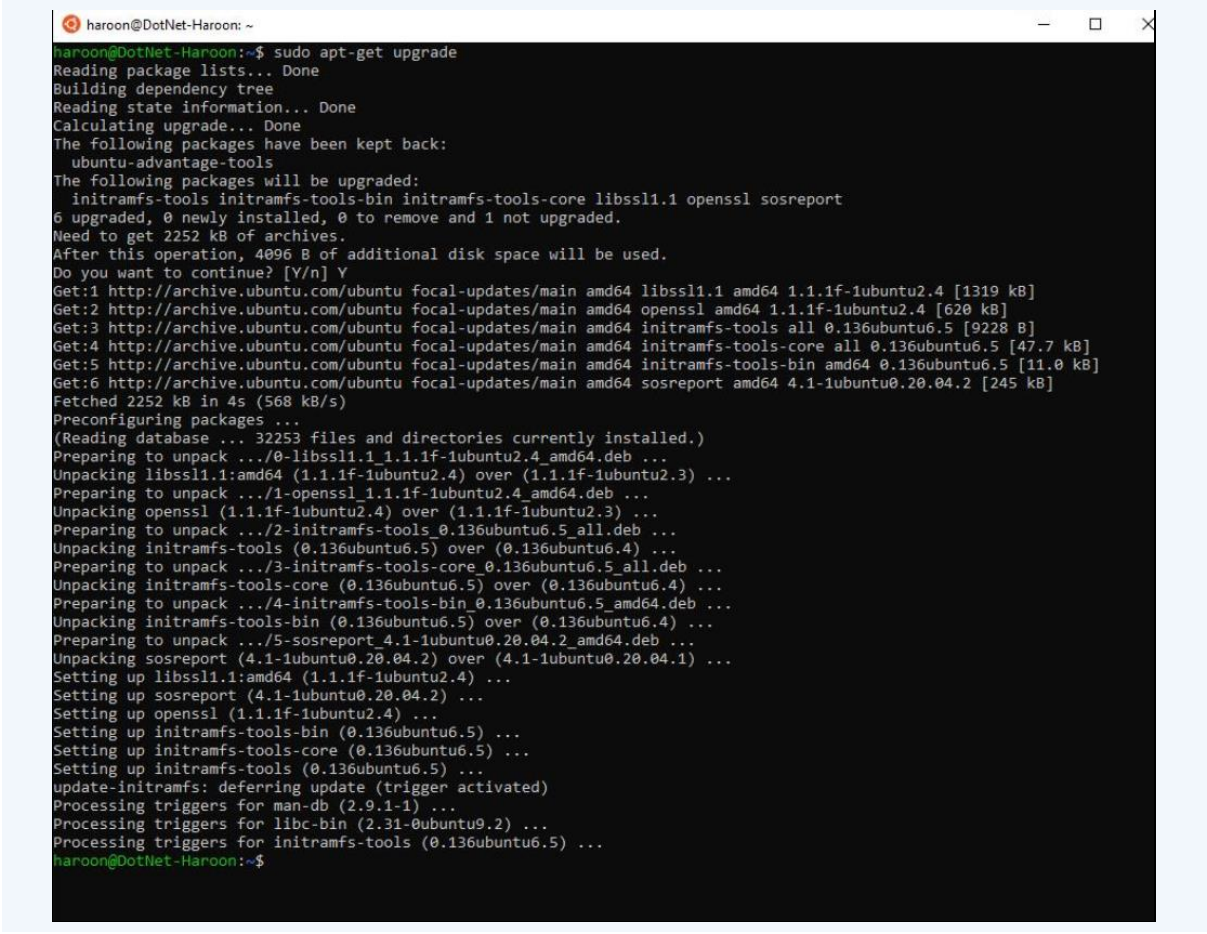

haroon@DotNet-Haroon: ~
1 update can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
This message is shown once a day. To disable it please create the
/home/haroon/.hushlogin file.
haroon@DotNet-Haroon:~$ sudo add-apt-repository ppa:redislabs/redis
[sudo] password for haroon:
Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker.
It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs,
geospatial indexes with radius queries and streams.
Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, a
nd provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.
More info: https://launchpad.net/~redislabs/+archive/ubuntu/redis
Press [ENTER] to continue or Ctrl-c to cancel adding it.
Hit:1 http://archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://ppa.launchpad.net/redislabs/redis/ubuntu focal InRelease [18.0 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security InRelease [109 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [8628 kB]
Get:7 http://ppa.launchpad.net/redislabs/redis/ubuntu focal/main amd64 Packages [1016 B]



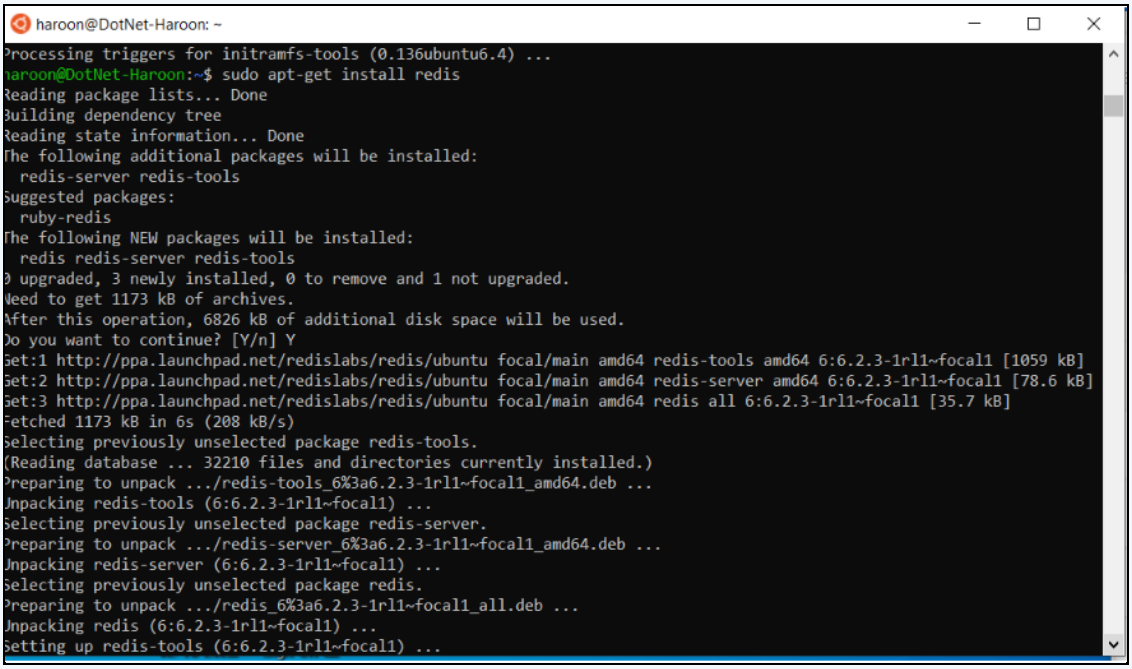
- Now that Ubuntu is running in Windows 10 environment, type following command to update newly installed ubuntu. **sudo apt-get update**



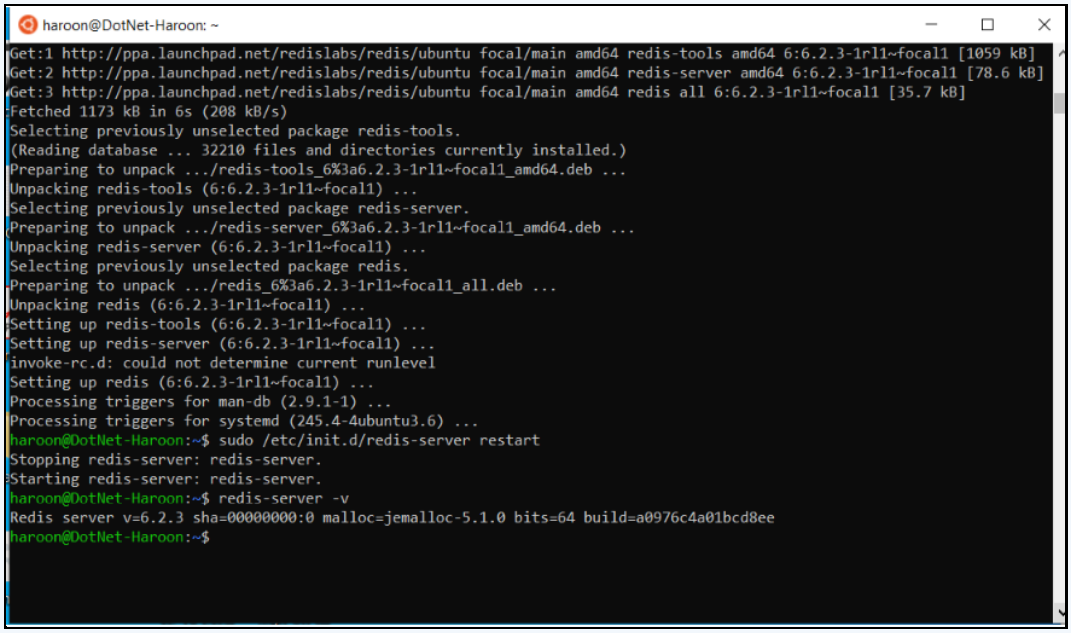
- After update, type upgrade command to upgrade the ubuntu instance. **sudo apt-get upgrade**



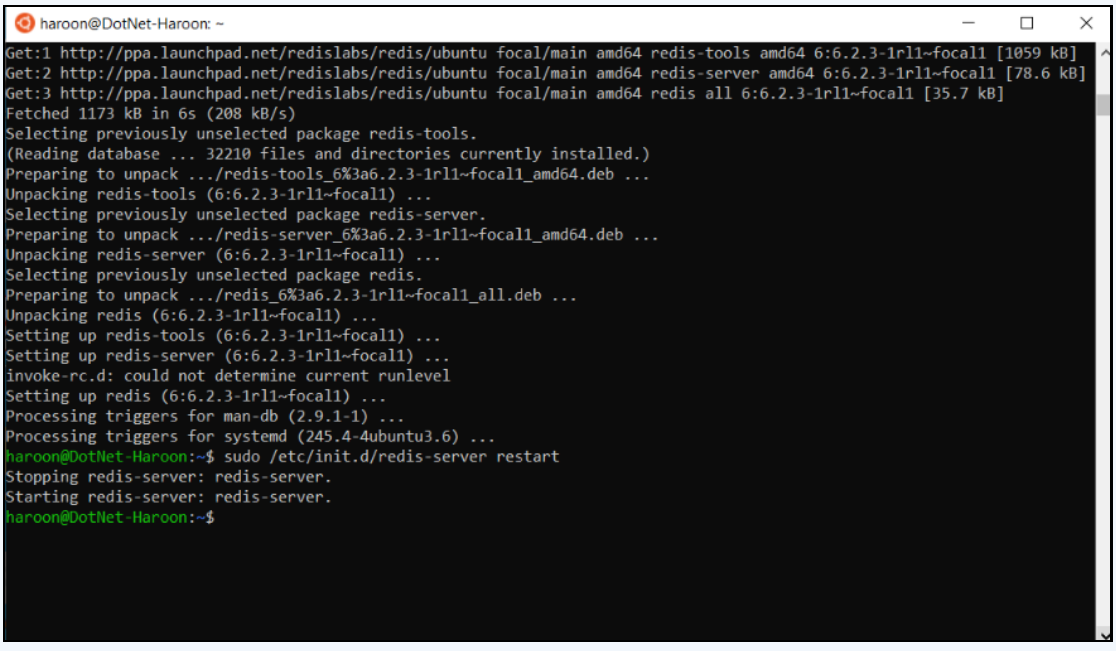
- Now we are ready to install Redis server using the following command. **sudo apt-get install redis-server**



- After installation Redis server can be reached with the following command to check if installation is properly done. **redis-cli –v**



- To ‘restart’ the Redis Server and to ensure it is running, type following command. **sudo service redis-server restart**



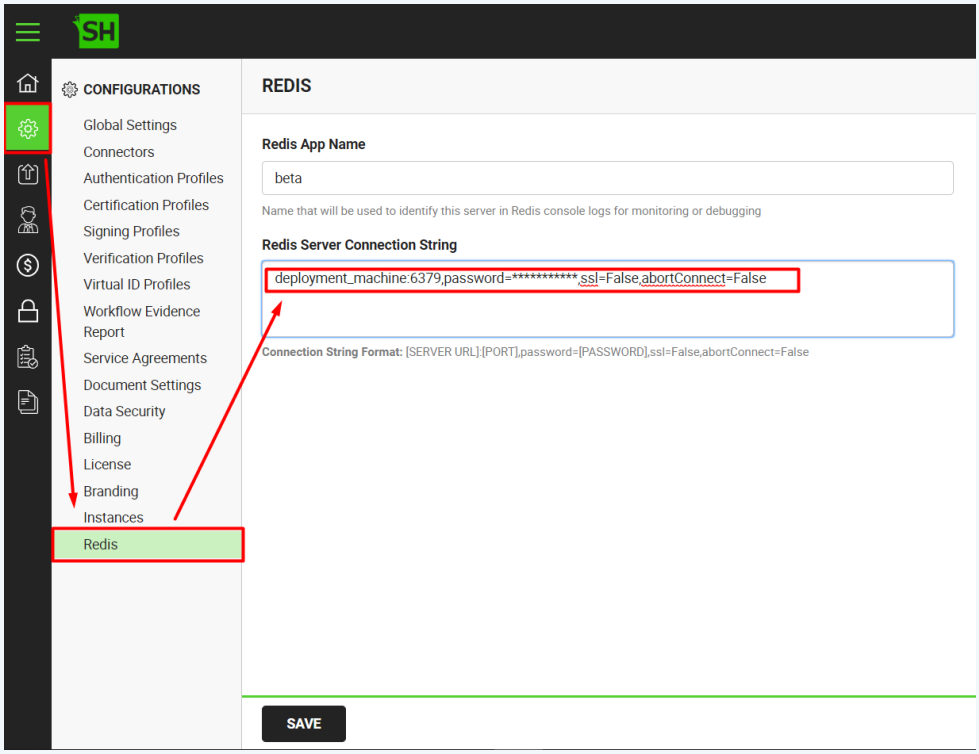
- After restart, ping Redis service using following command. **redis-cli and ping.**

To Install Redis on Linux Distribution using Windows Server environment same commands are applicable but 'Windows Subsystem for Linux' not supported in all versions of Windows Server, see Limitations section at the end.

Update Redis Server Configurations for SigningHub

To update Redis server connection string in SigningHub, login to SigningHub admin and navigate to 'Configurations -> Redis'. Redis Server connection string can be updated, as shown in below screenshot.

Click on '**Save'** to keep changes and restart IIS where SigningHub is deployed.



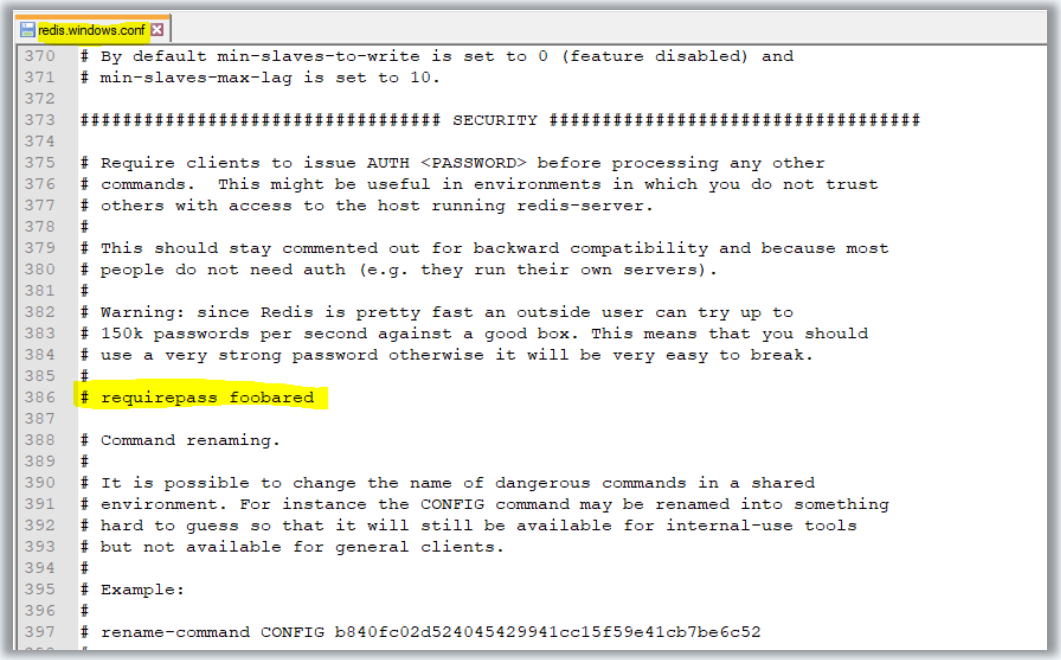
Security

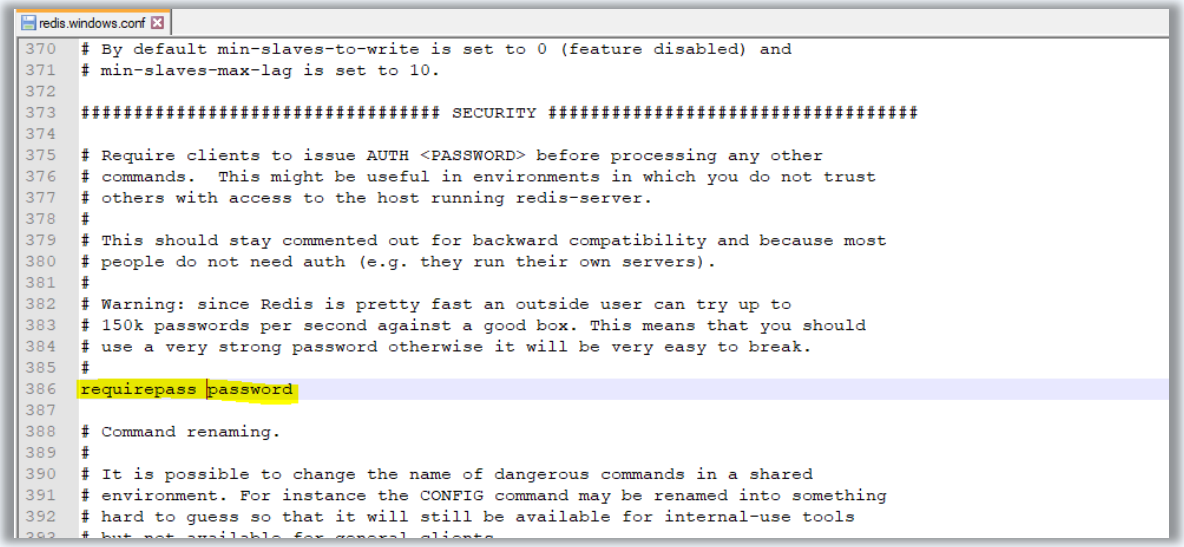
It is recommended to set a strong password for the Redis Service. Redis password can be set using following steps:

How to Set Redis Service Password?

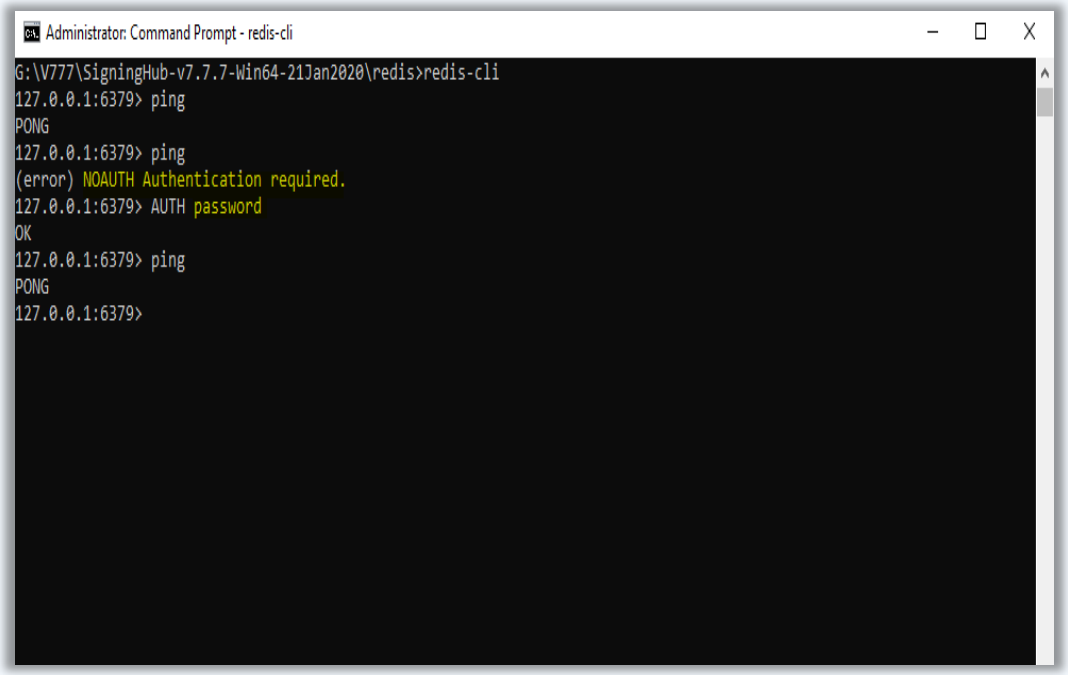
- By default Redis password will not be set
- Open Redis windows config file from package and edit '**requirepass foobared**' by omitting # and set custom password e.g. **requirepass password**

- After updating the file, restart Redis service from windows services



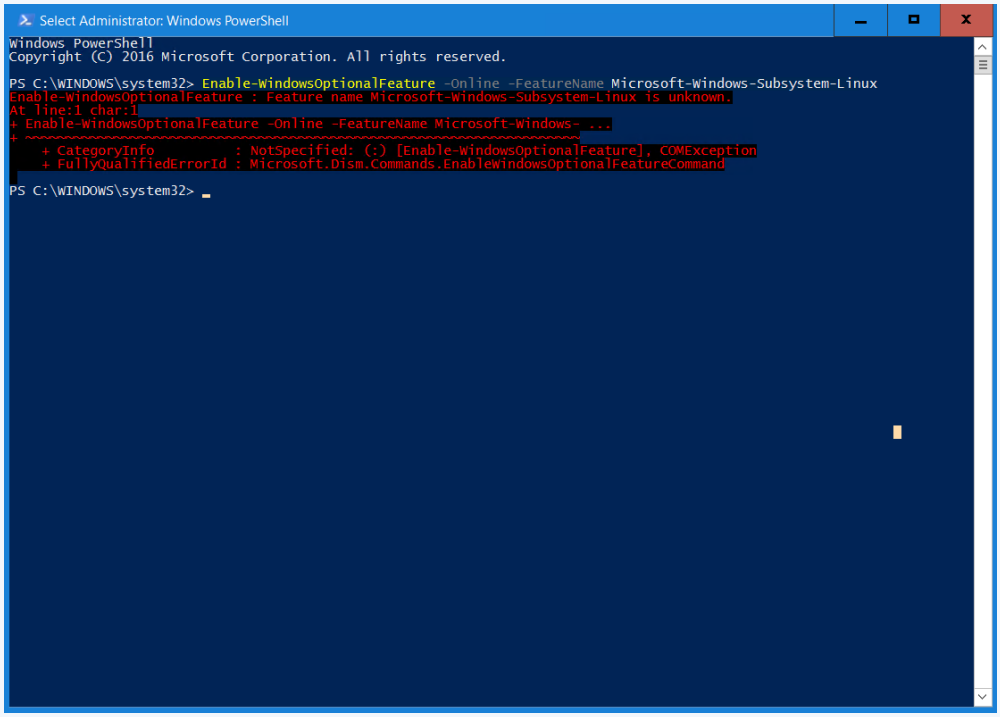


- Now again Access command prompt and ping Redis. System will ask for the password
- Enter set password by typing **AUTH {set password}**



Limitations

Windows sub system for Linux is supported from Windows Server 2016 (build 1709) and onwards. All previous versions do not support “Windows sub system for Linux”



For More Information about Redis go to the following link:
https://redis.io/topics/quickstart

For further details contact us on sales@ascertia.com or visit www.ascertia.com

Appendix I - Enable Transparent Data Encryption (TDE)

I.1 1 Introduction

Transparent Data Encryption can be configured for SigningHub, so that the data will be encrypted at rest for SigningHub database, which means the data and log files are protected from any potential threat of hacking or misusing of data.

Transparent Data Encryption (TDE) encrypts SQL Server data files, which are commonly known as encrypting data at rest. The user can take several precautions to help secure the database such as designing a secure system, encrypting confidential assets, and building a firewall around the database servers. However, in a scenario where the physical media (such as drives or backup tapes) are stolen, a malicious party can just restore or attach the database and browse the data. One solution is to encrypt the sensitive data in the database and protect the keys that are used to encrypt the data with a certificate. This prevents anyone without the keys from using the data, but this kind of protection must be planned.

TDE performs real-time I/O encryption and decryption of the data and log files. The encryption uses a database encryption key (DEK), which is stored in the database boot record for availability during recovery. The DEK is a symmetric key secured by using a certificate stored in the master database of the server or an asymmetric key protected by an EKM module.

It provides the ability to comply with many laws, regulations, and guidelines established in various industries. This enables software developers to encrypt data by using AES and 3DES encryption algorithms without changing existing applications.

I.2 How It Works?

To configure TDE, following are the high level steps that need to done.

- **Create master key**. The Server Master Key is created at the time of the initial SQL Server instance setup. The Service Master Key encrypts the database Master Key for the master database.
- **Create certificate protected by master key**. The database master key creates a certificate in the master database. Keep in mind that you must create a backup of this certificate. Not only for environmental refreshes but disaster recovery purposes.
- **Enable database encryption**. Once Transparent Data Encryption is enabled on the database, you won't be able to restore or move it to another server unless this same certificate has been installed.
- **Take a backup of certificate**. Keep good (and secure records) of the certificate and password.
- **Verify the TLS configurations** by executing queries on SQL Server, which are mentioned in upcoming sections.

I.3 Create Master Key

We must first create the master key. It must be created in the master database, so as a precautionary measure, begin this statement with the USE MASTER command. The syntax of the command is:

```
USE Master;
GO
CREATE MASTER KEY ENCRYPTION
BY PASSWORD='ProvideStringPassword';
GO
```

I.4 Create Certificate Protected by Master Key

Once the master key is created along with the strong password (that you should remember or save in a secure location), we will go ahead and create the actual certificate. The syntax of command is:

```
CREATE CERTIFICATE TDE_Cert
WITH
SUBJECT='Database_Encryption';
GO
```

I.5 Create Database Encryption Key

Now, we must utilize our USE command to switch to the database that we wish to encrypt. So, that we can create a connection or association between the certificate that we just created and the actual database. Then, we indicate the type of encryption algorithm we are going to use. In this case it will be AES_256 encryption. The syntax of command is:

```
USE TDE_TestDB
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES 256
ENCRYPTION BY SERVER CERTIFICATE TDE_Cert;
GO
```

I.6 Enable Encryption

Finally, we can enable encryption on our database by using the ALTER DATABASE command. The syntax of command is:

```
ALTER DATABASE TDE_TestDB
SET ENCRYPTION ON;
GO
```

Once the encryption is turned on, depending on the size of the database, it may take some time to complete. You can monitor the status by querying the sys.dm_database_encryption_keys

I.7 Backup Certificate

It's important to backup the certificate you created and store it in a secure location. If the server ever goes down and you need to restore it elsewhere, you will have to import the certificate to the server. In certain environments, the DR servers are already stood

up and on warm/hot standby, so it's a good idea to just preemptively import the saved certificate to these servers. The syntax of command is:

```
BACKUP CERTIFICATE TDE_Cert
TO FILE = 'C:\temp\TDE_Cert'
WITH PRIVATE KEY (file='C:\temp\TDE_CertKey.pvk',
ENCRYPTION BY PASSWORD='ProvideStrongPasswordHere')
```

Backup files of databases that have TDE enabled are also encrypted by using the database encryption key. As a result, when you restore these backups, the certificate protecting the database encryption key must be available. This means that in addition to backing up the database, you have to make sure that you maintain backups of the server certificates to prevent data loss. Data loss will result if the certificate is no longer available.

I.8 Restoring Certificate

In order to restore the certificate, you will once again have to create a service master key on the secondary server.

```
USE Master;
GO
CREATE MASTER KEY ENCRYPTION
BY PASSWORD='ProvideStrongPasswordHere';
GO
```

Once that is done, you must remember where you backed up the certificate and the encryption/decryption password.

```
USE MASTER
GO
CREATE CERTIFICATE TDECert
FROM FILE = 'C:\Temp\TDE_Cert'
WITH PRIVATE KEY (FILE = 'C:\TDECert_Key.pvk',
DECRYPTION BY PASSWORD = 'InsertStrongPasswordHere' );
```

Something to note before applying TDE are its drawbacks which Transparent Data Encryption encrypts the underlying database files including the backups. You can't just take the files and dump them onto another SQL Server without the appropriate encryption keys and certificates. It does NOT allow for granular user level encryption.

I.9 Restoring Certificate

In order to verify if the TDE has been implemented on a specific database, use following queries:

1. Following query will return the databases list and ‘encryption_state =3 shows the database is actually TDE enabled.

```
Select * from sys.databases
```

2. Following query will return the list of certificates and the one that was created above can be looked in the table.

```
Select * from sys.certificates
```

3. Following query will return the databases encryption keys details.

```
Select * from sys.dm_database_encryption_keys
```

I.10 Limitations and Restrictions on TDE

The following operations are not allowed during initial database encryption, key change, or database decryption:

- Dropping a file from a filegroup in the database
- Dropping the database
- Taking the database offline
- Detaching a database
- Transitioning a database or filegroup into a READ ONLY state

The operations which are not allowed while performing the processes like `CREATE DATABASE ENCRYPTION KEY, ALTER DATABASE ENCRYPTION KEY, DROP DATABASE ENCRYPTION KEY, or ALTER DATABASE...SET ENCRYPTION` statements are:

- Dropping a file from a filegroup in the database
- Dropping the database
- Taking the database offline
- Detaching a database
- Transitioning a database or filegroup into a READ ONLY state
- Using an ALTER DATABASE command
- Starting a database or database file backup
- Starting a database or database file restore
- Creating a snapshot

The following operations or conditions will prevent the `CREATE DATABASE ENCRYPTION KEY, ALTER DATABASE ENCRYPTION KEY, DROP DATABASE ENCRYPTION KEY, or ALTER DATABASE...SET ENCRYPTION` statements:

- The database is read-only or has any read-only file groups.
- An ALTER DATABASE command is executing.
- Any data backup is running.
- The database is in an offline or restore condition.
- A snapshot is in progress.
- Database maintenance tasks.

- When creating database files, instant file initialization is not available when TDE is enabled.

In order to encrypt the database encryption key with an asymmetric key, the asymmetric key must reside on an extensible key management provider.

All these configurations are verified on SQL Server 2019 Standard Edition for now. TDE is only supported to specific SQL Server versions.

Appendix J - Enable Transport Security Layer (TLS)

J.1 Introduction

SigningHub application must be configured over TLS (Transport Layer Security) to connect it with the SQL Server database over the secure connection.

The reason behind incorporation of TLS into the SigningHub is because it provides the data encryption while using the SQL Server. The SQL Server can use Transport Layer Security (TLS) in order to encrypt the data which will be transmitted across the network between an instance of SQL Server and the client application. The basic reason to configure TLS into the SQL server is:

- Enabling TLS encryption increases the security of data transmitted across networks between instances of SQL Server and applications.
- TLS performs server validation, for instance, when a client connection will request the encryption, then following procedure will run: If the instance of SQL Server is running on a computer that has been assigned a certificate from a public certification authority, identity of the computer and the instance of SQL Server is guaranteed by the chain of certificates that lead to the trusted root authority. Such server validation will be going to require that the computer on which the client application is running should be configured to trust the root authority of the certificate that is used by the server.

J.2 How It Works?

To enable TLS configurations for SigningHub following are high level steps that need to be done.

- **Configure a certificate** (where Common Name (CN) must have Fully Qualified Domain Name (FQDN), Enhanced Key Usage must have Server authentication, Subject Alternative Name (SAN) property must have FQDN under DNS)
- **Import this certificate under Microsoft Management Console (MMC) > Trusted Certificates**, on machine where SQL Server is running.
- **Set Force Encryption** to true under SQL Configuration Manager.
- **Select Certificate** from certificate tab under SQL Server Configuration Manager.
- **Restart SQL Service**
- Add required parameters under **connection string**.

- Run the application after **restarting IIS**.

J.3 Configuration for Certificate Enrollment

To setup TLS configuration Certificate, following are requirements which must be done.

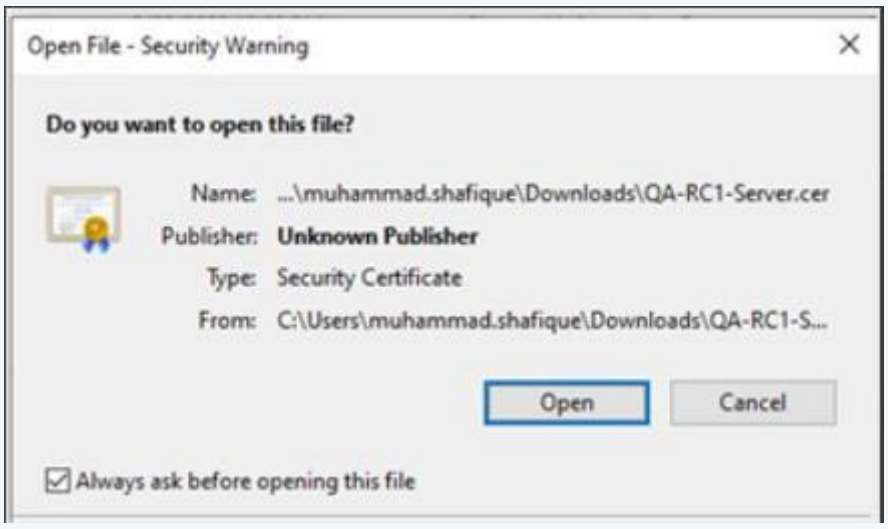
A Certificate is required initially (which can be created under ADSS or any third party CA). There are certain requirements that a certificate needs to have in order to work seamlessly with the SQL Server. Some important information which must be known while creating a certificate is as follows:

CN (Common Name) under Subject property of certificate must have Fully Qualified Domain Name (FQDN) as a parameter, of your SQL Server. (For instance SQLDB.ascertia.com.pk).

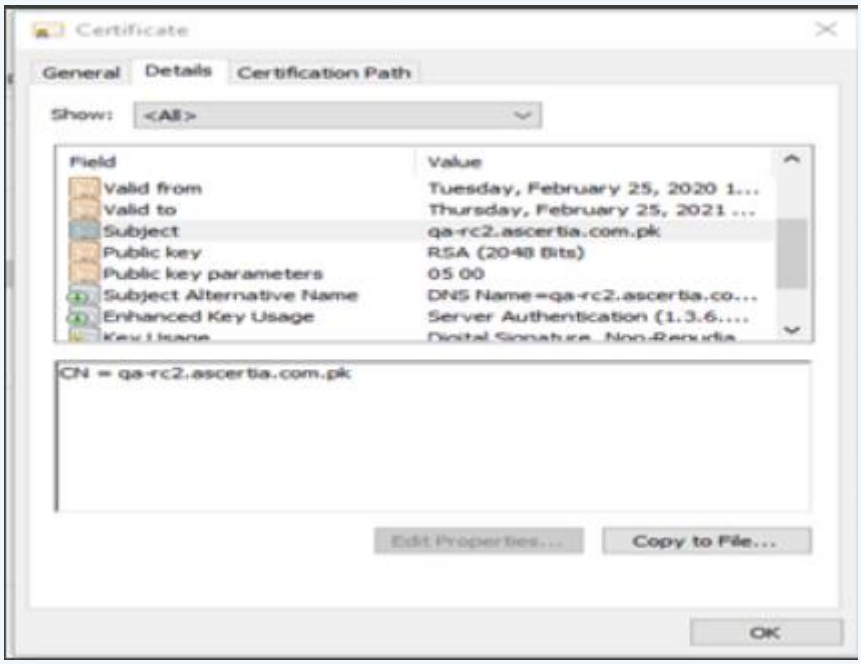
CN can be defined as the Subject property on the certificate which must specify a Common Name (CN) that should be same as the host name or fully qualified domain name (FQDN) of the SQL Server.

In order to perform **Server Authentication** without any error, FQDN for SQL Server must be known.

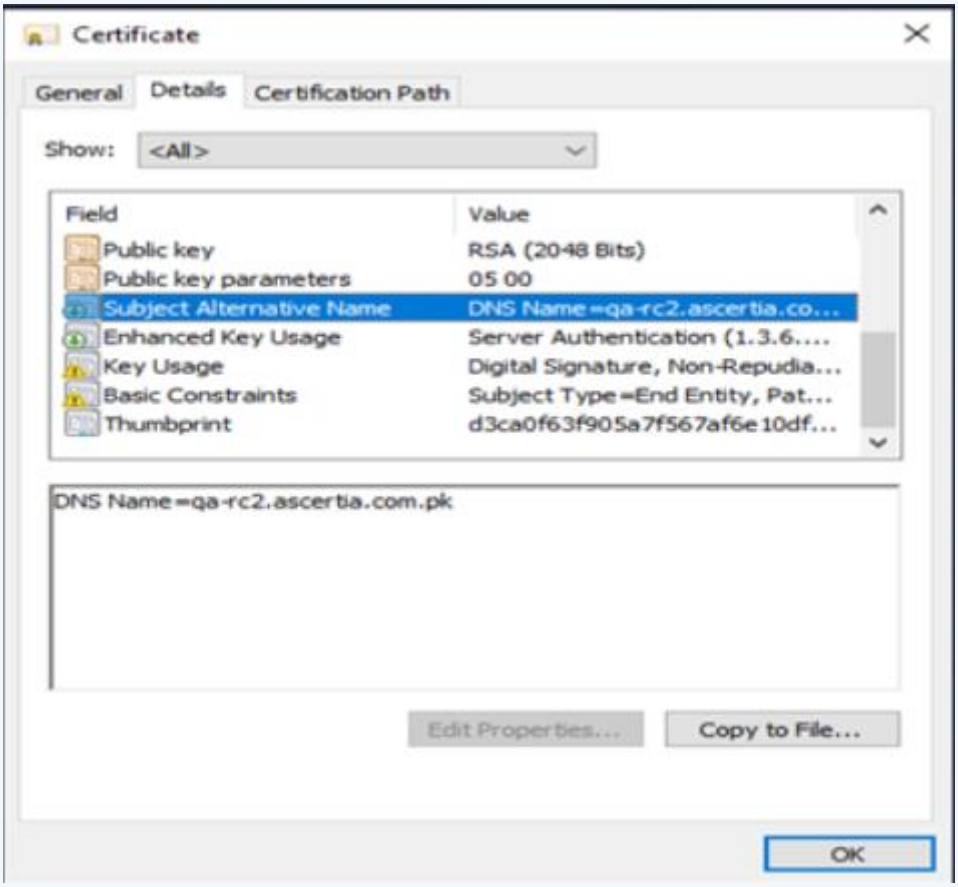
To verify these parameters, double click on certificate and following dialog will appear.



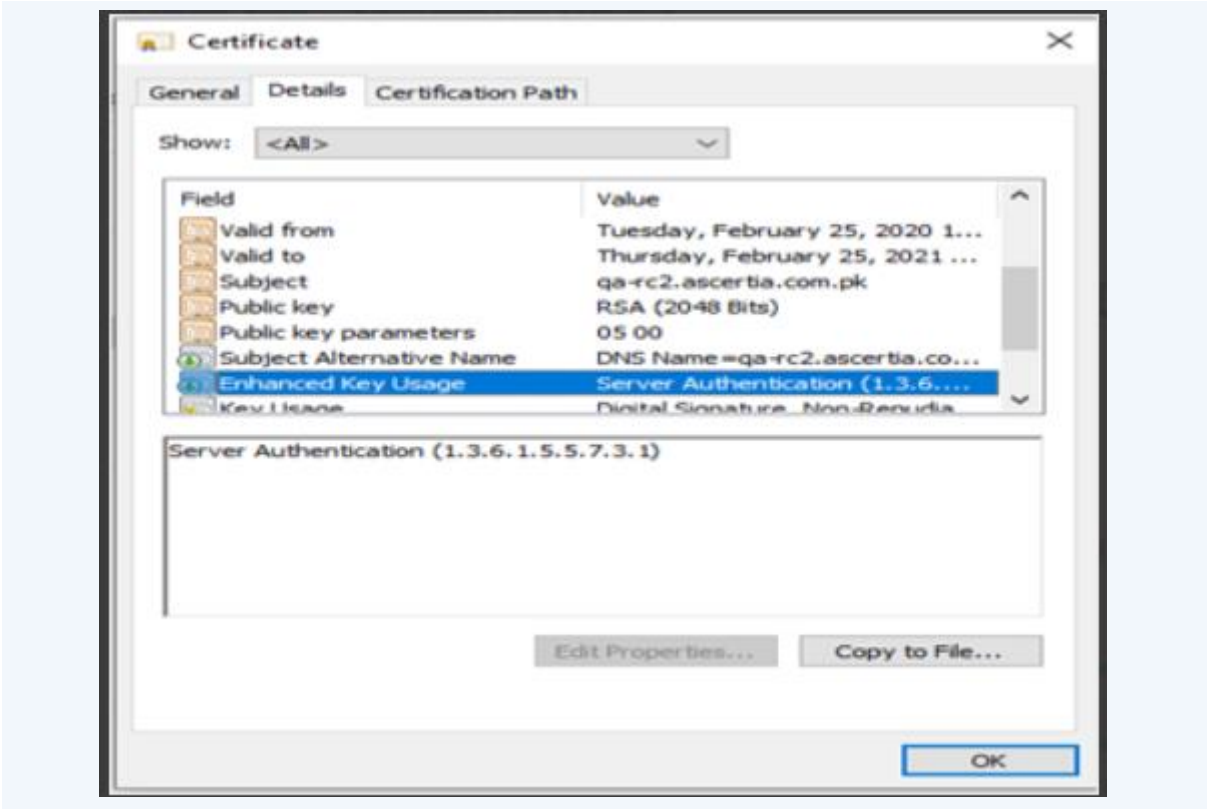
Click ‘Open’ and go to the ‘Details’ Tab and select Subject field. CN must contains FQDN as a parameter value.



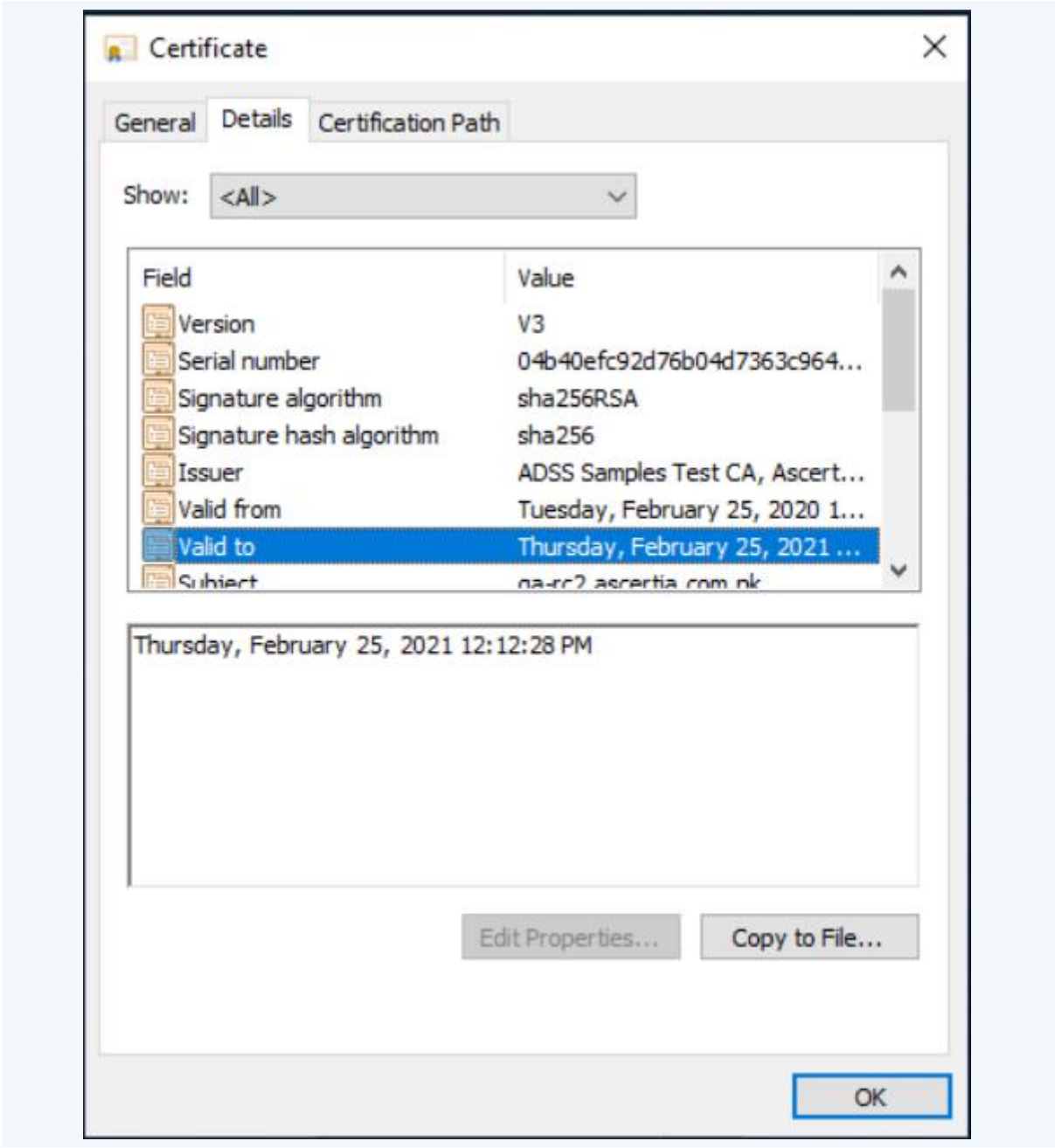
DNS (Domain Name System) under the **Subject Alternative Name** (SAN) property must contain **FQDN**. As shown in the image below.



The certificate must be meant for server authentication. Select **Enhanced Key Usage** field. The value under this property must contain **Server Authentication (1.3.6.1.5.5.7.3.1)**.



The current system time must be after the **Valid from** property of the certificate and before the **Valid to** property of the certificate, to make sure that the Certificate must be valid i.e. valid from and valid to values are within current date.

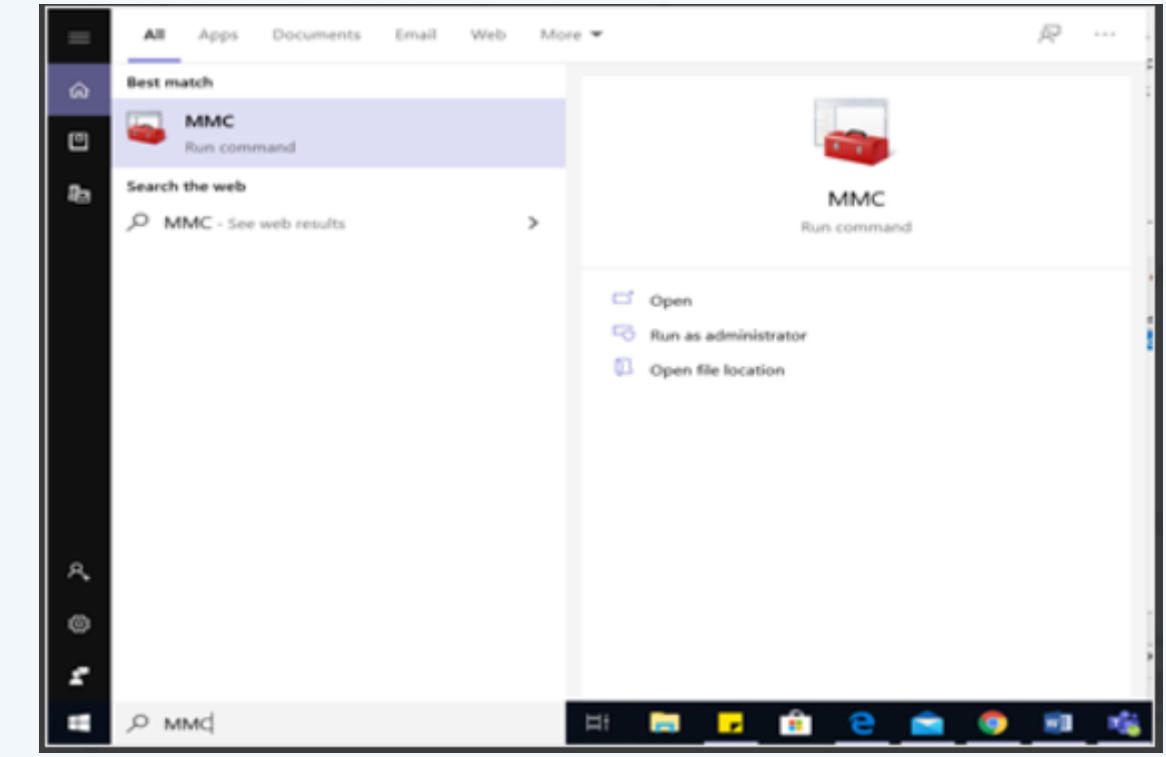


Once a certificate with above pre-requisites is generated, then certificate needs to import under certificate store of machine where the SQL Server is running.

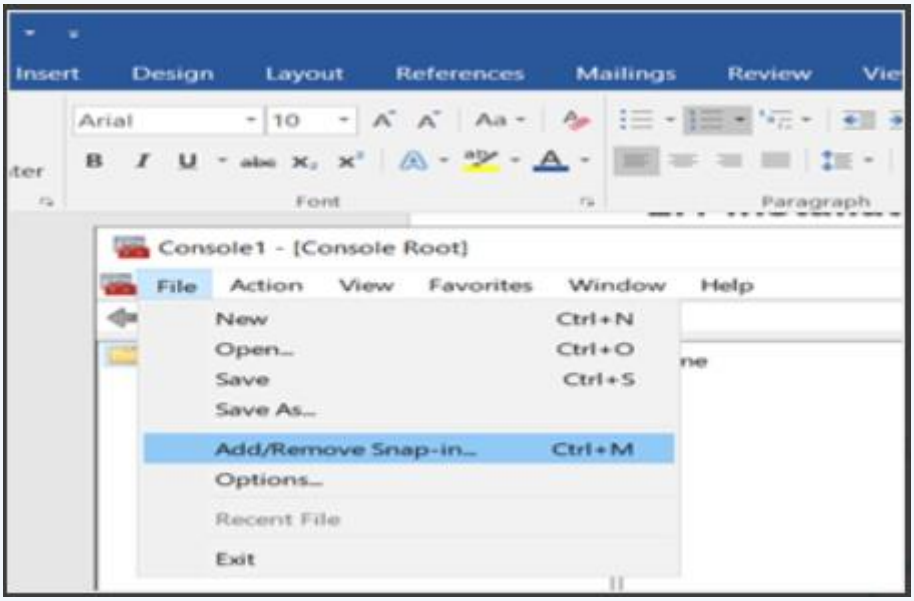
J.4 Installation of SQL Server Certificate Using Microsoft Management Console

If you obtained a certificate which fulfils the above requirements then, you are now able to import it to the certificate store on your database server, while following the steps mentioned below.

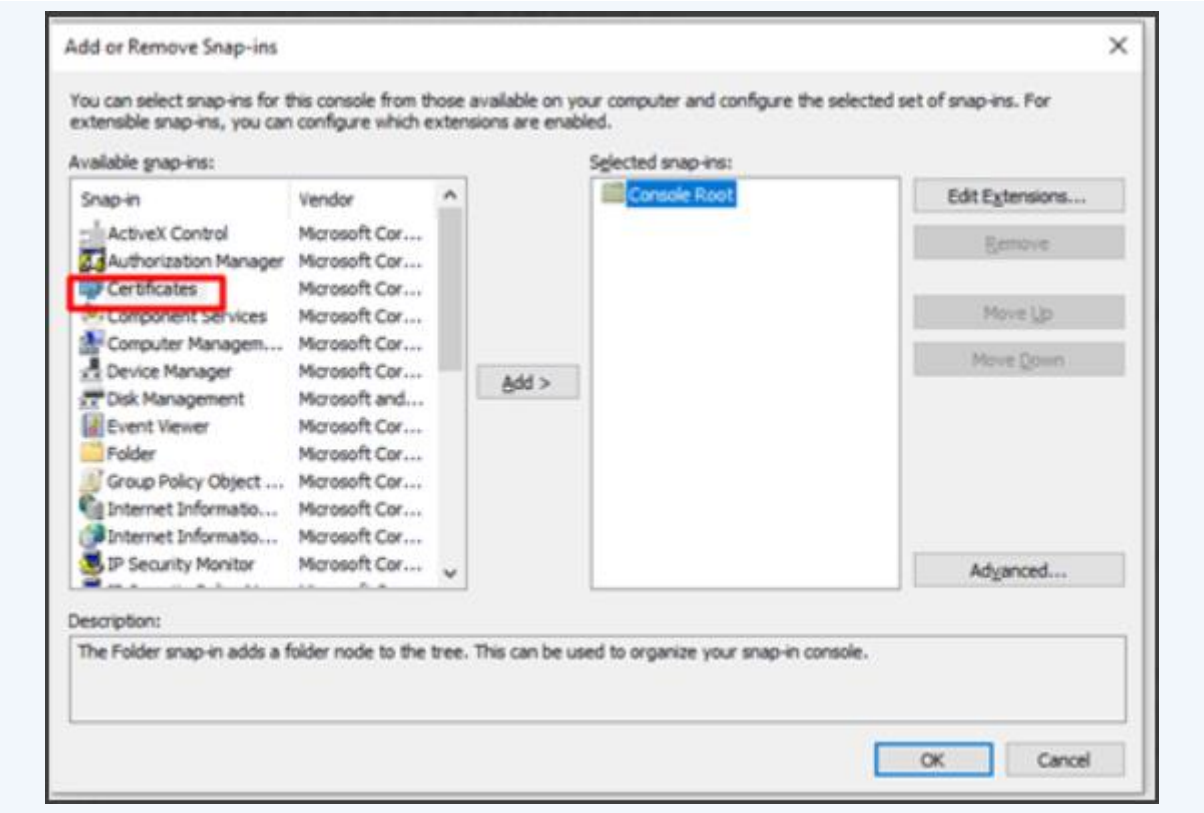
First click on the Start menu and write MMC in the search box and then Open the Microsoft Management Console (MMC). Press Open button, as shown below.



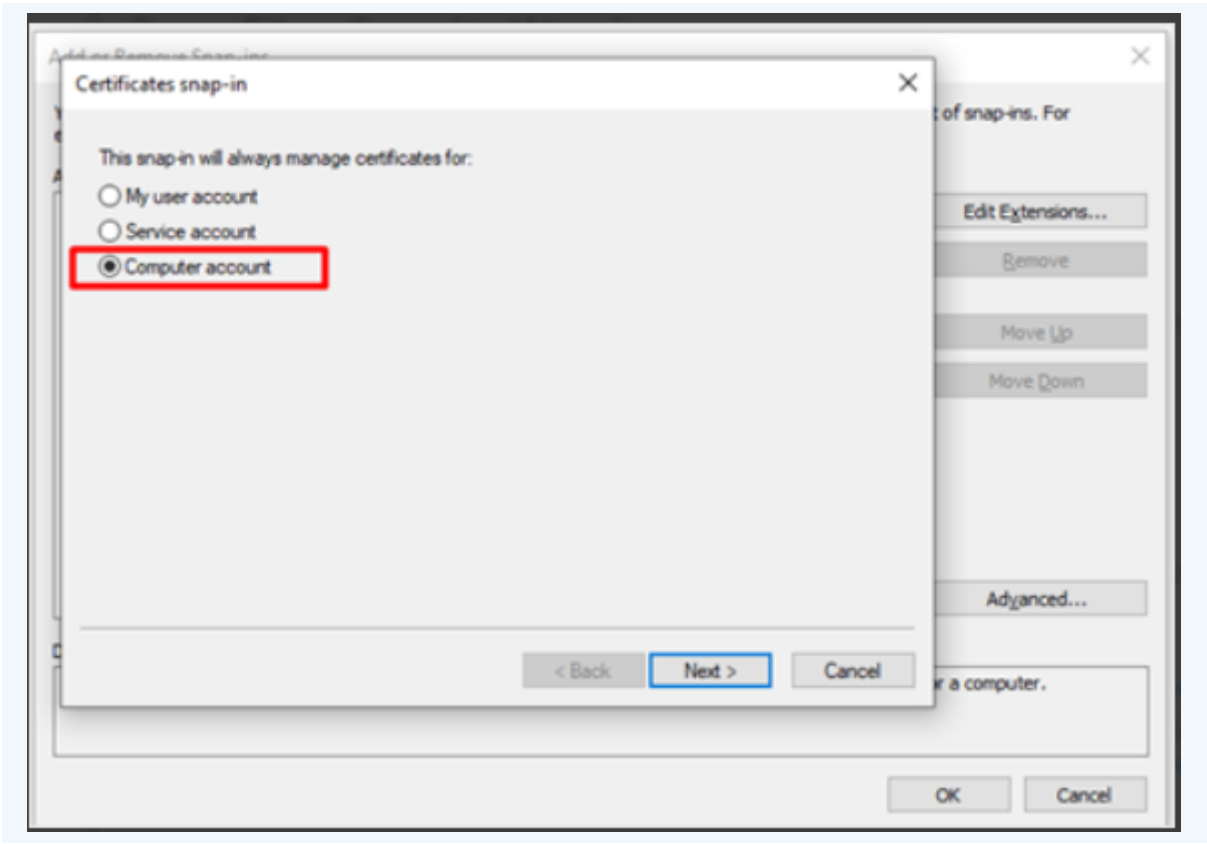
On the File menu, click Add/Remove Snap-in, as shown below.



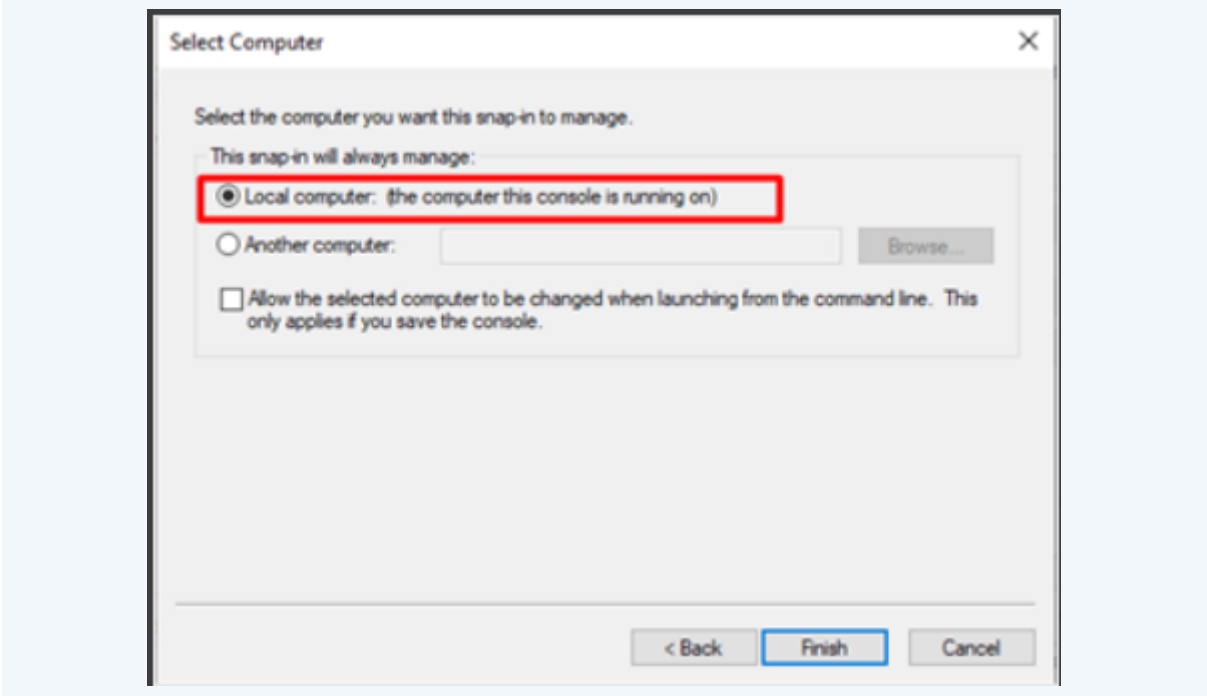
Select Certificates, click on Add button.



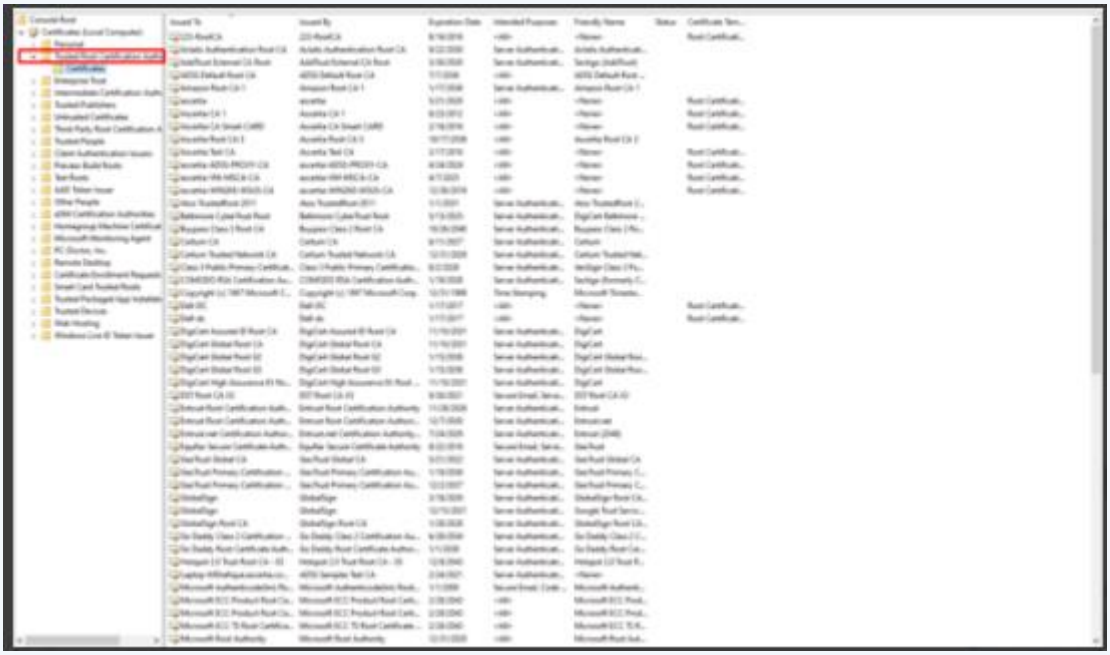
Now you are prompted to open the certificates snap-in, select computer account and click Next.



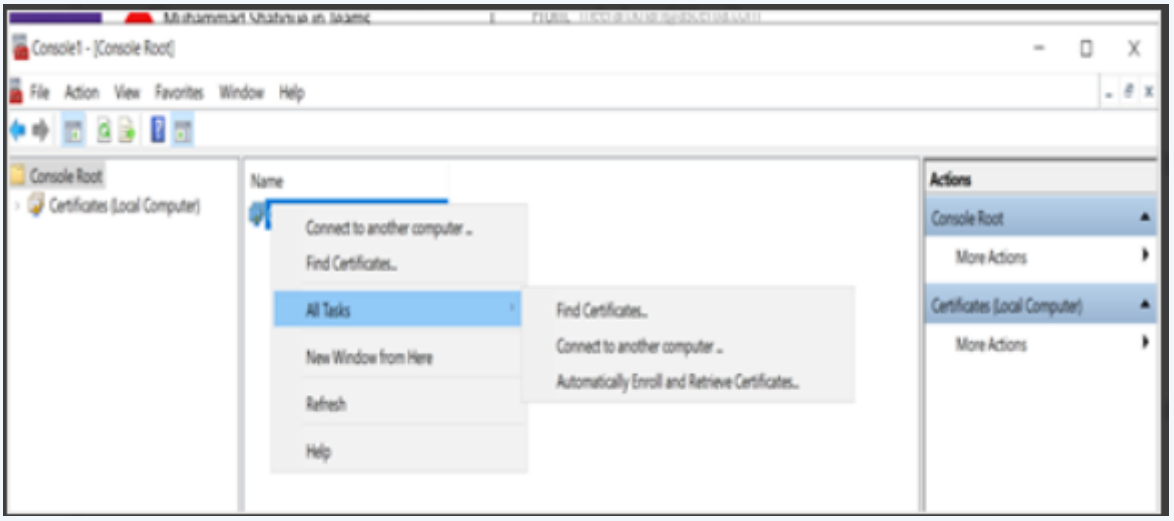
The certificate must be located in either the local computer certificate store or the current user certificate store, thus, select Local computer, and then click Finish, as shown below.



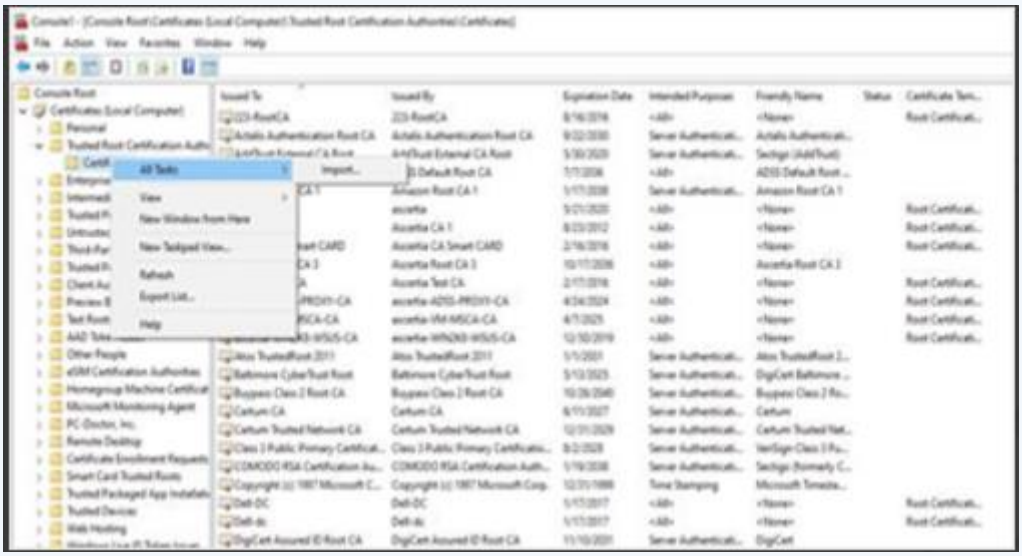
Click OK in the Add/Remove Snap-in dialog box. After that click, select the **Trusted Root Certification Authorities** folder in the left pane.



Click on **Certificates** and right click to select **All Tasks->Import**.



Import required certificate, that was created using above steps.



Now, you must be able to see the certificate in the folder with the fully qualified domain name of your SQL Server.

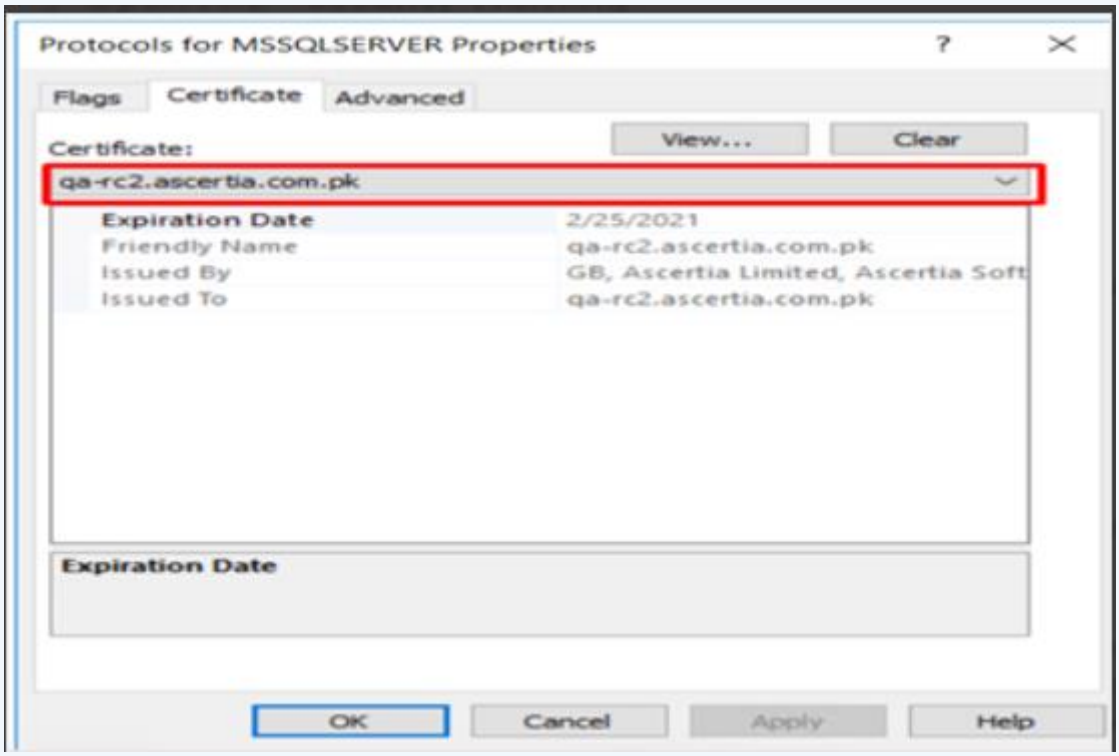
J.5 Configuration of SQL Server to Use Encrypted Connections

Once you've successfully installed the certificate, it needs binding to the database engine service in SQL Server Configuration Manager.

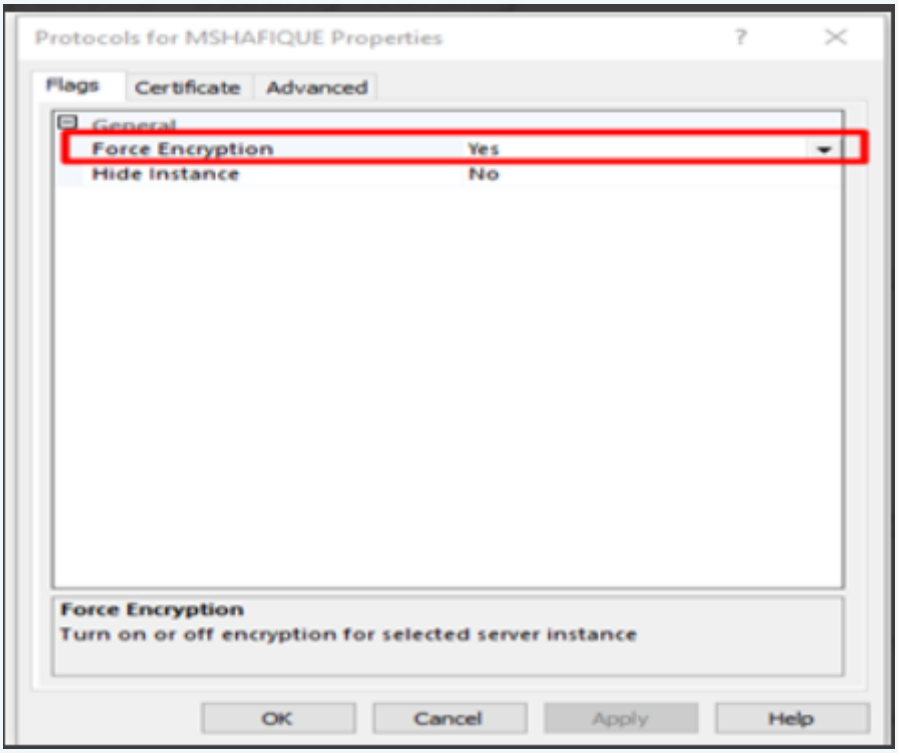
To configure TLS using the SQL Server Configuration Manager. First run SQL Server Configuration Manager under the SQL Server service account. The only exception is if the service is running as Local System, Network Service, or Local Service, in this case you can use an administrative account.

Expand SQL Server Network Configuration and right-click on **Protocols for <YourMSSQLServer>**, select Properties.

On the Certificate tab, select the certificate you would like to use and click ‘OK’.

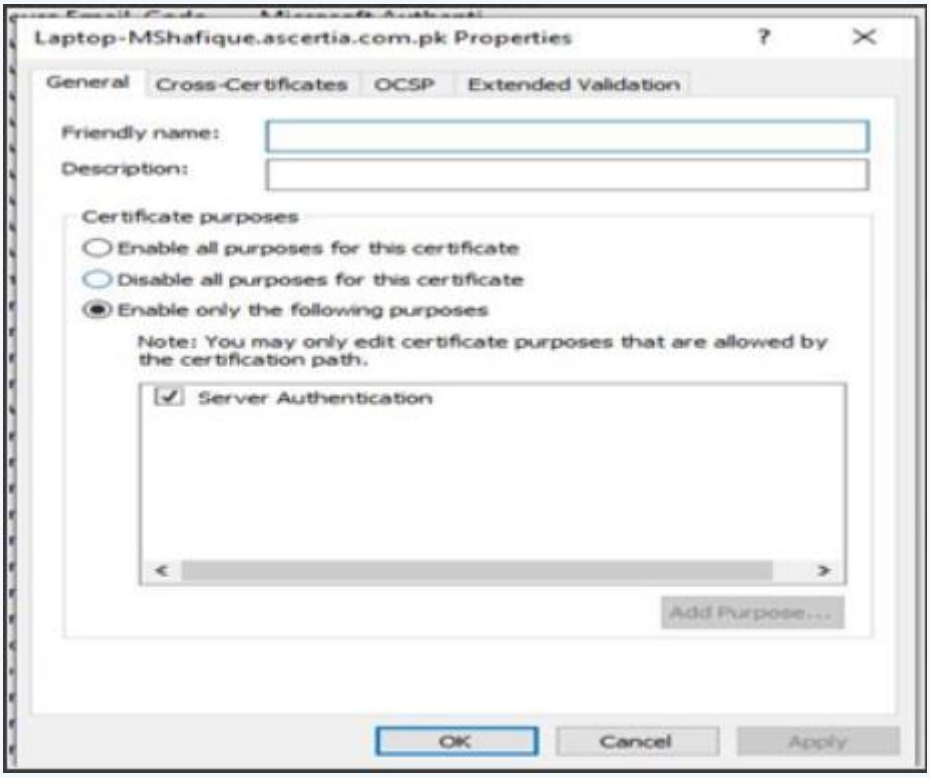


On the Flags tab, select 'Yes' in the **Force Encryption** box, then click OK.



Restart the SQL Server service.

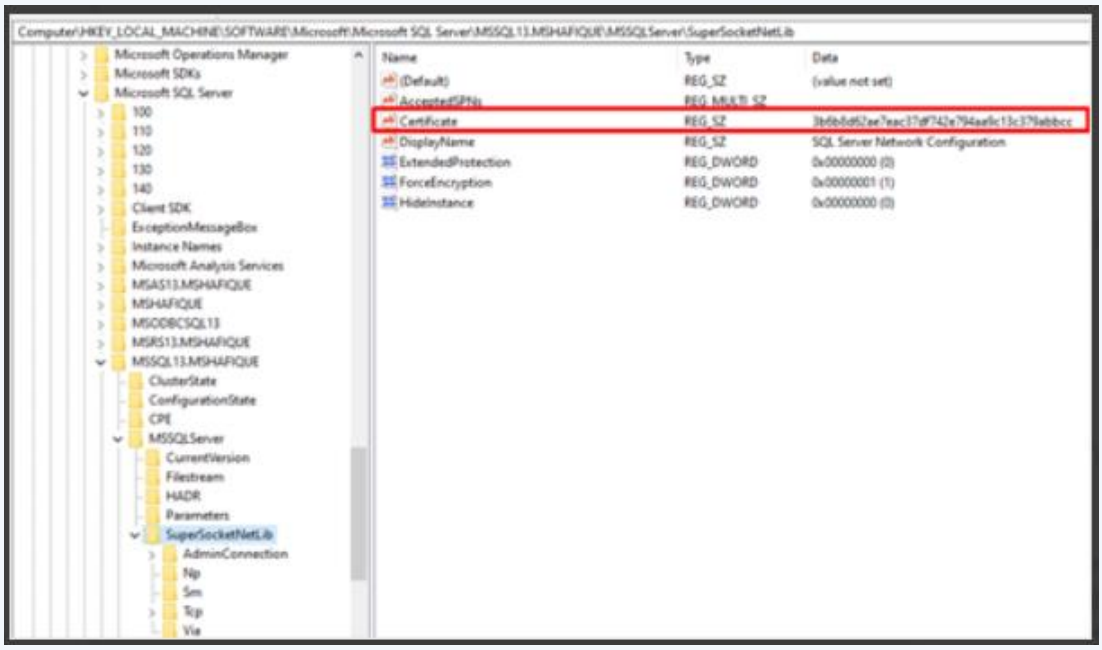
Right click to certificate under MMC, and under properties select **Enable only the following purposes** option. Select **Server Authentication** option, as shown below.



J.6 Post Configurations Steps

If certificate is not available in the list, then following steps are required.

Click Start and write Run and now write **Regedit** in the field and then open it.



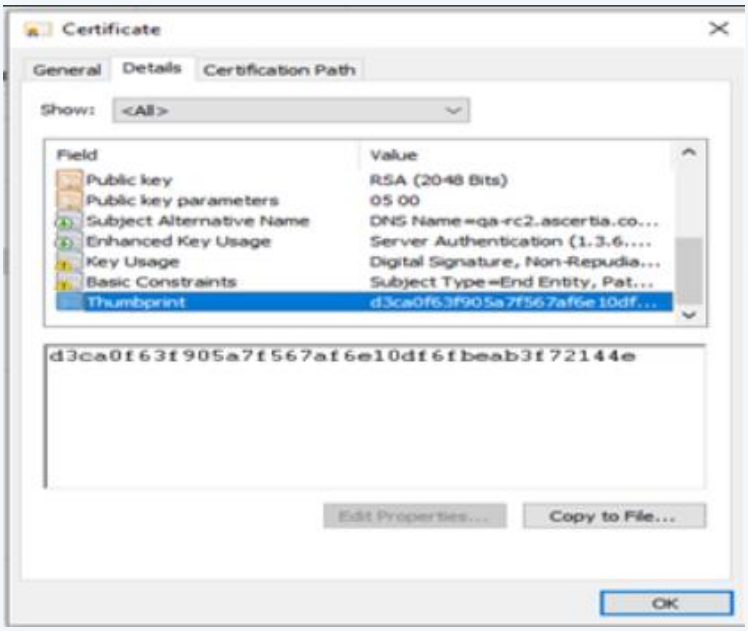
Add **Thumbprint** property value under registry (Thumbprint value can be obtained from the certificate details.

Following path which will be used to find the **Certificate Key.**

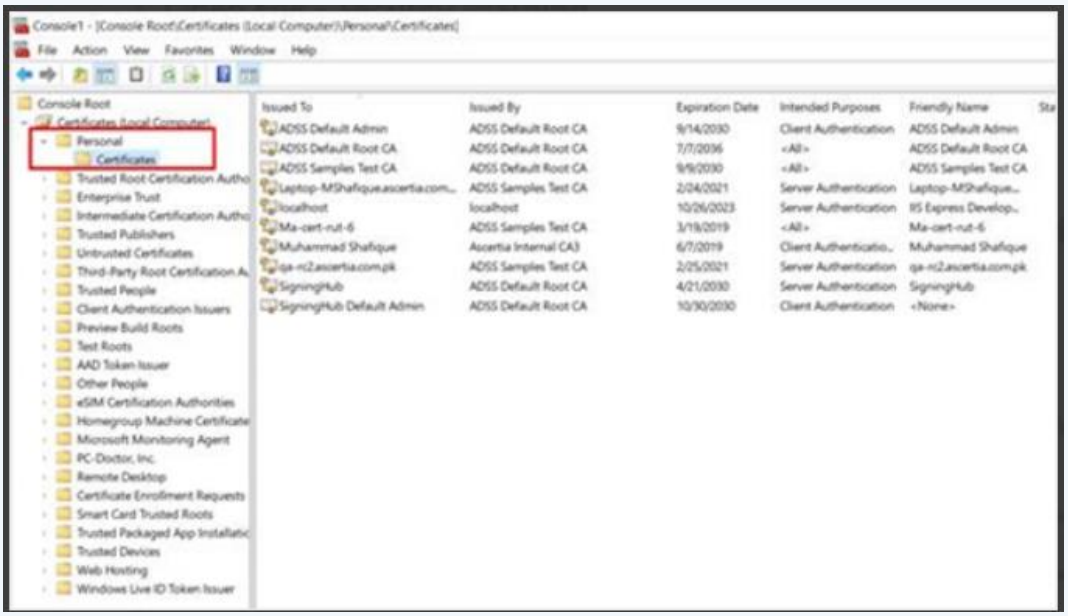
```
HKEY_LOCAL_MACHINE -> Software -> Microsoft -> Microsoft SQL Server -
> MSSQL13.MSSQLSERVER->MSSQLServer->SueperSocketNetLib->Certificate
(Key)
```

Example Value for Thumbprint:

`d3 ca 0f 63 f9 05 a7 f5 67 af 6e 10 df 6f be ab 3f 72 14 4e` (It must be without spaces).



Also add the PFX under personal store under MMC of SQL DB Server, as illustrated below.



SQL Server Service Account must have the Read permissions to access the TLS certificate by SQL Server Configuration Manager.

J.7 Configurations Required for SigningHub Connection String

To setup SigningHub application to run over TLS connection with SQL Server, connections string for SigningHub must be set as per following recommendations.

1. Two parameters that are required to add under connection string to setup TLS connection for SigningHub with SQL Server.

```
Encrypt=true;TrustServerCertificate=false
```

2. Add these parameters under web config of application in following format:

```
connectionString="data source={%FQDN_SQLSERVER%},1433;initial
catalog={%DB_NAME%};user
id=sa;password={%PASSWORD%};MultipleActiveResultSets=True;Pooling=true
;Encrypt=true;TrustServerCertificate=false"
```

Remember to use FQDN of your SQL Server in data source under connection string in order to make TLS work.

3. Restart IIS after changes in web configuration.

J.8 Verification of TLS Configurations

In order to verify if the connections are being encrypted with SQL Server DB, execute following query on SQL Server.

```
SELECT session_id, connect_time, net_transport, encrypt_option,
auth_scheme, client_net_address
FROM sys.dm_exec_connections where encrypt_option = 'true'
```

It will return all the encrypted connections for the DB Server.

To verify whether all the connections are encrypted or not, execute following query. This has set three levels i.e. RED means no encrypted connections, GREEN means all the connections are encrypted and AMBER shows there were some connections that are not encrypted.

Here is the SQL Query:

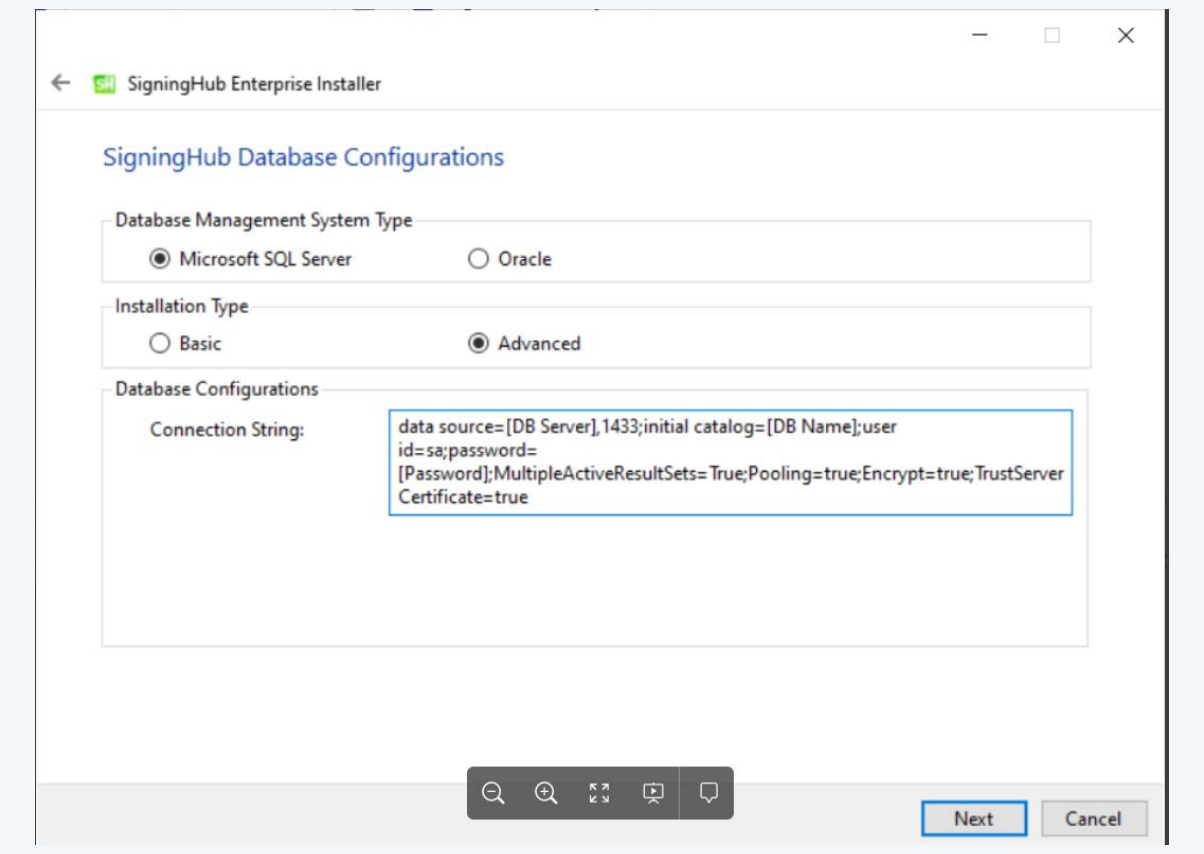
```
WITH CTE Summary AS
(
SELECT    encrypt_option, COUNT(*) cnt
FROM      sys.dm_exec_connections
GROUP BY  encrypt_option
)
SELECT @@ServerName AS [Server Name],
CASE WHEN COUNT(*)=1 AND MIN(encrypt_option) = 'FALSE' THEN 'RED - no
connections are encrypted'
WHEN COUNT(*)=1 AND MIN(encrypt_option) = 'TRUE' THEN 'GREEN - all
connections are encrypted'
ELSE 'AMBER - some connections are encrypted'
END AS [Connection Encryption RAG Status]
FROM  CTE_Summary
```

J.9 Update Existing SigningHub Instance Connection String Over TLS

If an existing SigningHub instance has to be configured over the TLS then connection string has to be updated, and it can be done using SigningHub installer using update database credentials option.

Just simply execute **install.bat** file (Run as Administrator), which is present in 'Setup' folder under the package installation directory. Select **Change DB Credentials** option and provide connection string as follows.

Click **Next** and the database credentials will update. Click on finish to complete the process.



For further details contact us at sales@ascertia.com or visit www.ascertia.com

*** End of Document ***