ADSS Server v8.4.0 – Auto File Processor (AFP) Admin Guide

ASCERTIA LTD

FEBRUARY 2026

Document Version- 1.0.0

CONTENTS

FIGURES

TABLES

1 Introduction

1.1 Scope

This manual describes how to install and configure the Ascertia ADSS Auto File Processor (AFP) software.

1.2 Intended Readership

This manual is intended for ADSS AFP administrators responsible for its installation and configuration. It is assumed that the reader has a basic knowledge of standard protocols, PKI and IT security.

1.3 Conventions

The following typographical conventions are used in this guide to help locate and identify information:

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

1.4 Technical support

If Technical Support is required, Ascertia has a dedicated support team providing debugging assistance, integration assistance and general customer support. Ascertia Support can be accessed in the following ways:

Website	https://www.ascertia.com
Email	support@ascertia.com
Knowledge Base	Ascertia Community Portal

In addition to the free support service describe above, Ascertia provides formal support agreements with all product sales. Please contact sales@ascertia.com for more details.

A Product Support Questionnaire should be completed to provide Ascertia Support with further information about your system environment. When requesting help it is always important to confirm:

- System Platform details
- ADSS AFP and ADSS Server version numbers and build date
- Details of the specific issue and the relevant steps taken to reproduce it
- ADSS Server database version and patch level
- The product log files

2 ADSS AFP Concepts & Architecture

2.1 Overview

ADSS Auto File Processor (AFP) is flexible front-end application for ADSS Server. It provides intelligent watched folder automatic monitoring of one or more Windows folders or UNIX directories to process batches of documents in an unattended environment.

ADSS AFP delivers this by monitoring a set of input folders (sometimes call directories) for documents. Any document found in a specified input location is processed according to the operator defined ADSS AFP policy. Digital signature creation or verification requests are processed by making calls to ADSS Server. The processed documents are delivered to a defined output folder or, if a problem has been detected, an error folder.

ADSS Server Auto File Processor represents one of the simplest ways of utilising the power of the ADSS Server within organisations since it requires no programmatic integration.

2.2 Features & Benefits

- **Handles multiple watched folders**

  ADSS AFP can manage multiple watched folder profiles, each defines a set of input, output and error folders and the signing policy to be used. It is thus able to handle multiple different business document types.

- **Supports multiple signature formats**

  ADSS AFP signing policies can use the full power of ADSS Server and thus sign using basic and long-term signatures in various formats including: PDF/PAdES, PDF/A, XML /XAdES, PKCS#7, CMS/ CAdES and MS Office (Word and Excel). It is thus able to handle multiple business requirements.

- **Supports bulk signing and other services in future**

  ADSS AFP can currently request documents to be signed using ADSS Server Signing Service. In future, support for verification and LTANS archiving will be offered. At this point one ADSS AFP system will support multiple processing types.

- **Scalability and Resilience**

  ADSS AFP has been designed for high throughput and high availability, and it can load balance requests across multiple ADSS Servers to ensure that business systems are unaffected by single point failures.

- **Sophisticated Filtering**

  ADSS AFP allows watched folder profiles to include complex rules such as defining sub-folders, filename filters, batch sizes and operational timers. This allows system tuning to optimise throughput and resource utilisation.

- **Security**

  To ensure that only authorised requests are processed, ADSS AFP authenticates itself to ADSS Server using a pre-defined originator ID. For stronger authentication ADSS AFP can optionally use TLS client authentication. ADSS Server retains a secure log of all request and response transactions and can provide detailed reports from these.

- **Cost Effective**

  ADSS Server is multi-function product and only the modules your business requires need to be licensed. This provides a flexible yet cost-effective solution, with in built investment protection since other modules can be added later to support future business needs. There are no transactional costs - a server can be used to process any number of documents.

2.3 Deployment Scenarios

The scalable architecture of AFP Server lends itself to numerous deployment options. A typical deployment may need a single AFP and ADSS Server co-located on the same server or same network while complex deployments may require multiple load balanced AFP and ADSS Servers deployed on multiple servers on the same/different networks. The way you deploy AFP Server depends on what you want to do with it. If you are using the product for development or testing purposes, you don't need an extensive deployment, but if you are planning for the production system then you need to give extra consideration and resources to factors such as processing loads, single points of failure, and security.

As mentioned above, typically, AFP Server can be used in two possible ways:

1. Simple watched folder processing, where ADSS Server does all hashing and signature processing including signature appearances for PDFs. This is typically used when AFP and ADSS Server are co-located on the same server or same network.
2. More complex watched folder processing where ADSS Server operates remotely from AFP, possibly as a cloud service. As a consequence, AFP does all the hashing and signature appearance handling, signature embedding and creates a secure TLS v1.2 and TLS v1.3 connection to the ADSS Server.

The following deployment scenarios are presented as guides for you to think about as you prepare to build your automated watched folder monitoring and document signing system. Although you could deploy your system exactly as presented in one of the scenarios, you will probably want to use/look at these scenarios just to get ideas of what's possible with AFP Server, and then adjust your own deployment to fit your specialized needs and hardware resources.

Contact our solution consultant sales@ascertia.com if you need help to quickly understand how AFP should be used to meet your requirements.

The following diagram shows how ADSS AFP can be used with ADSS Server for automated document signing. Back end trust services such as Timestamp Authorities (TSAs) and OCSP Validation Authority servers may be required where long-term signatures are requested:

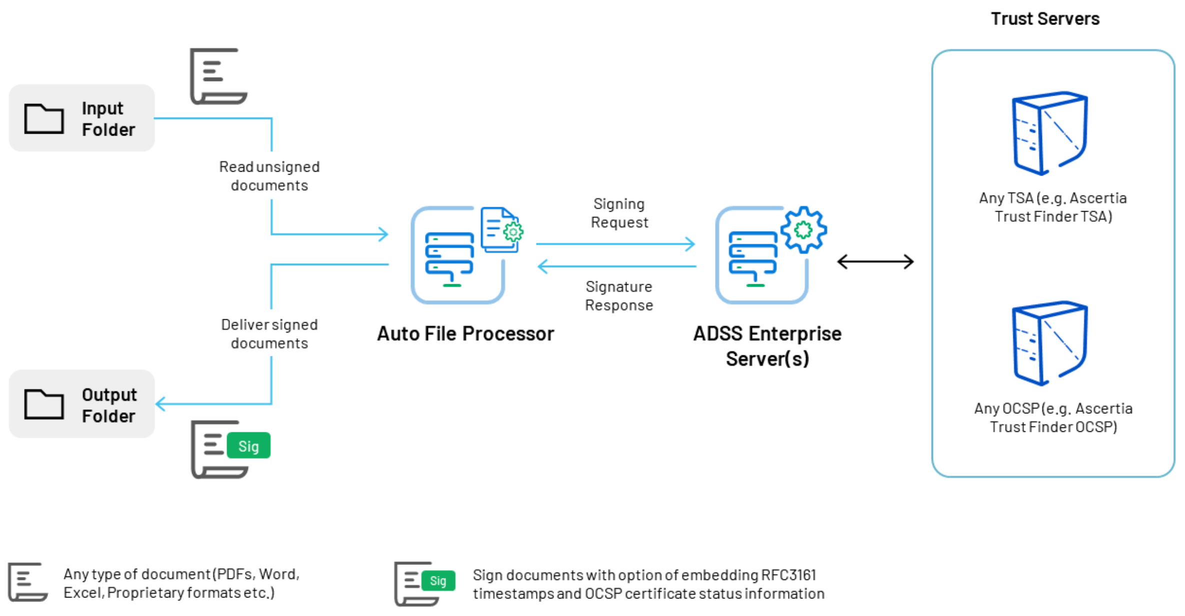

Trust Servers
Input Folder
Read unsigned documents
Signing Request
Signature Response
Auto File Processor
ADSS Enterprise Server(s)
Any TSA (e.g. Ascertia Trust Finder TSA)
Deliver signed documents
Output Folder
Sig
Any OCSP (e.g. Ascertia Trust Finder OCSP)
Any type of document (PDFs, Word, Excel, Proprietary formats etc.)
Sign documents with option of embedding RFC3161 timestamps and OCSP certificate status information

Figure 1 - Deployment Architecture

ADSS AFP uses an optimised HTTP communication protocol for communication with ADSS Server to ensure high performance operations.

2.4 Multiple Watched Folder Profiles

It is possible to set-up multiple ADSS AFP watched folder profiles each with its own set of input, output and error folders as illustrated below. Each AFP watched folder profile can be associated with its own Signing Profile configured on ADSS Server:

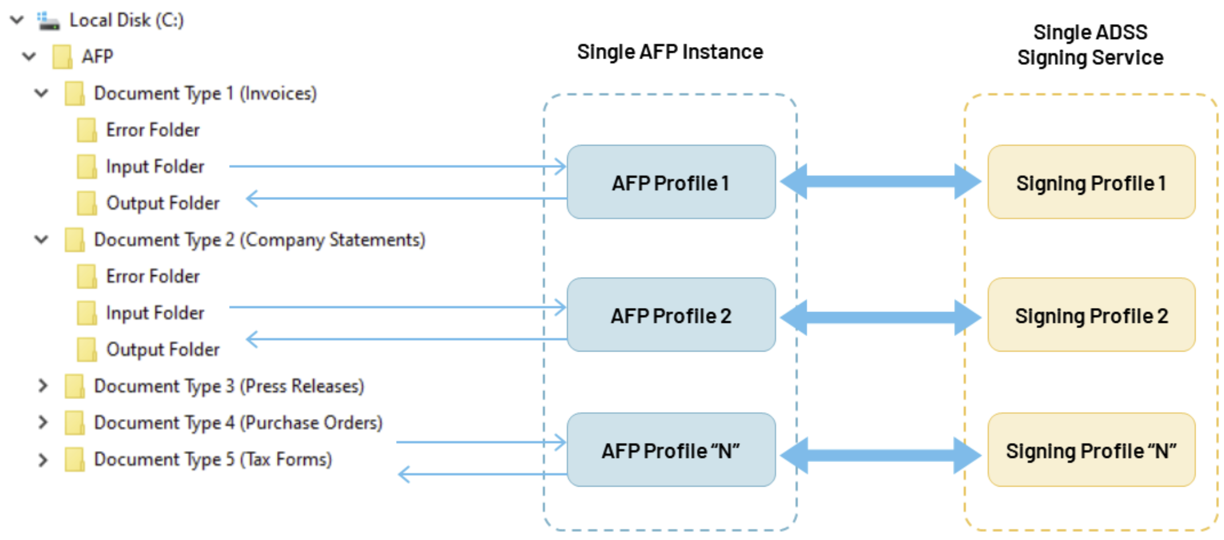


Figure 2 - Watched Folder Profiles

The above diagram illustrates multiple ADSS AFP profiles processing various document types located in different input folders. The documents are then signed on ADSS Server using unique Signing Profiles and the resultant file deposited in the appropriate output folder.

Signing Profiles are configurations on ADSS Server which define the type of signature to produce, including:

- Type of signature to create (PDF/PAdES, PDF/A, XML/XAdES, CMS/CAdES, PKCS#7, MS Office.
- Which signing key to use
- Which info to include in the signature (e.g. signing reason, signer's location, signing policy etc.).
- In case of PDF signatures which signature appearance to stamp on the document (e.g. which logos to include, their size and positioning, etc).
- Plus, many additional low-level configurations

2.5 High Availability Multi-Site Configuration

Multiple ADSS Servers can be listed for each ADSS AFP profile to provide fault tolerance. This ensures that if the primary ADSS Server is unavailable a back-up secondary ADSS Server will be automatically invoked:

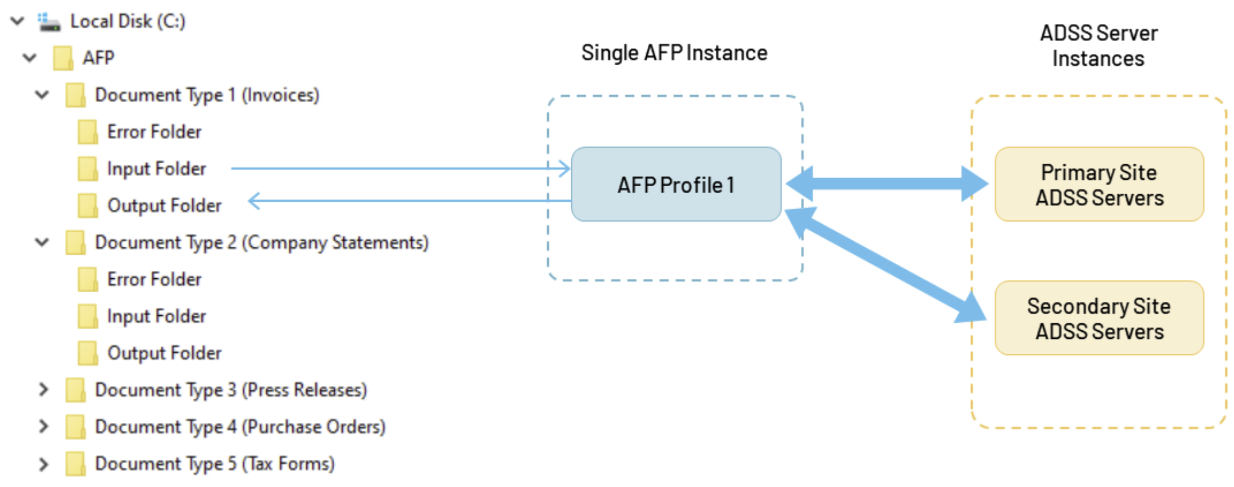

Local Disk (C:)
AFP
Document Type 1 (Invoices)
Error Folder
Input Folder
Output Folder
Document Type 2 (Company Statements)
Error Folder
Input Folder
Output Folder
Document Type 3 (Press Releases)
Document Type 4 (Purchase Orders)
Document Type 5 (Tax Forms)
Single AFP Instance
AFP Profile 1
ADSS Server Instances
Primary Site ADSS Servers
Secondary Site ADSS Servers

Figure 3 - High Availability to ADSS Server

2.6 ADSS Server High Availability Load Balanced Configurations

A load balancer can be deployed to ensure that two or more primary servers are used to provide greater throughput and resilience:

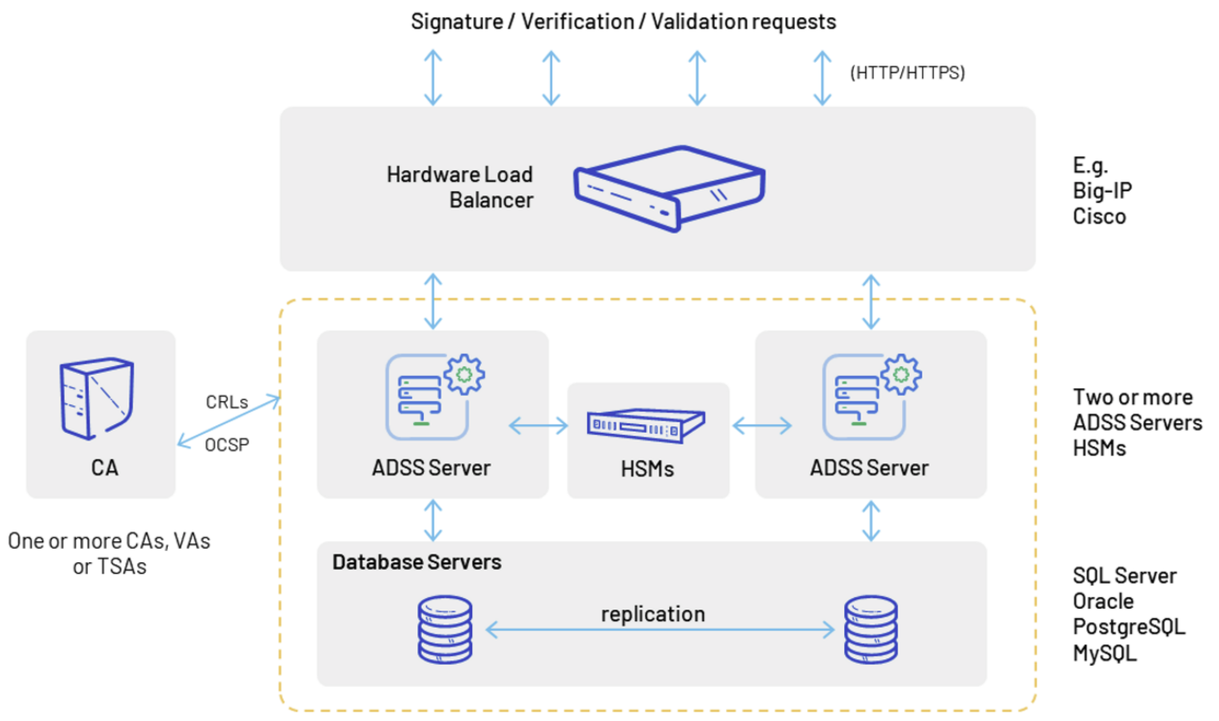


Figure 4 - High Availability Architecture

2.7 High Availability Load Balanced Configuration

Multiple ADSS AFP installations can share a single input folder to provide greater throughput and resilience. High availability and load balancing is achieved as shown below:

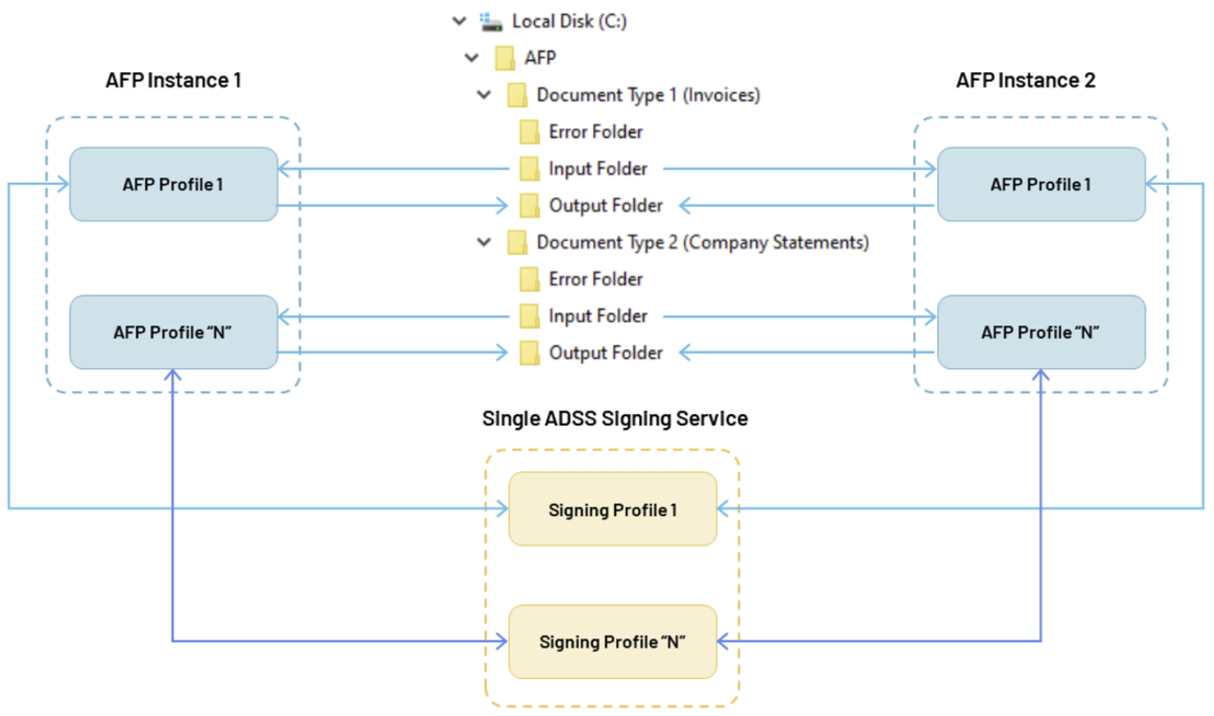


Figure 5 - High Availability Architecture with Load Balancing

3 System Requirements

The following table summarizes the minimum requirements to support ADSS AFP deployments:

Component	Minimum Requirements
Operating System	The following 64-bit operating systems are supported (32-bit on request): • Windows Server 2022, 2019, 2016 • Linux (RedHat v7.x, v8.x, CentOS v7.x, v8.x, SUSE)
CPU/RAM	A modern fast CPU with 4GB RAM (6GB RAM in case of local hashing) and 2GB disk space is recommended. Additional RAM may be required to handle more concurrency. Typically, 0.5GB to 1GB of disk space will be required depending on usage and transactional data / log retention requirements.

Table 1 - System Requirements

4 ADSS AFP Installation

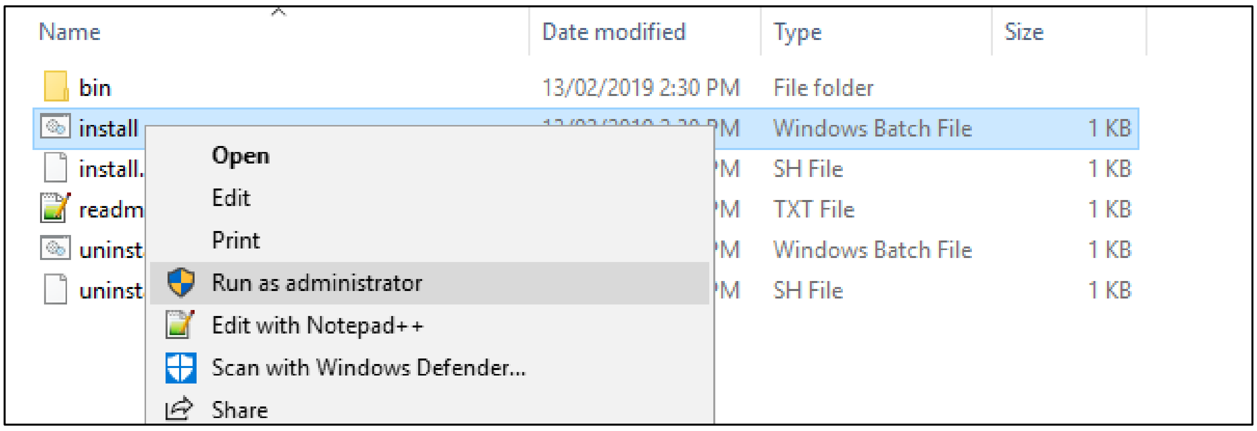
ADSS AFP setup is not like other installations which create desktop icons, shortcuts, asks for target folder where the product is to be installed, etc. Instead it is provided as a zipped set of files and running **install.bat/sh** merely performs the basic configuration before it can be used. After installation you will still need to perform further configuration as described in the next section.

4.1 Installation Instructions

First extract ADSS AFP installation zip to any directory where you want to install ADSS AFP. Navigate to location **[ADSS AFP Installation Directory]\setup** and follow these steps to install ADSS AFP on windows/Unix platforms:

4.1.1 Windows

Run the **install.bat** file under administrative privileges, as shown below, (otherwise ADSS AFP services will not be registered in Windows Services Panel) to install ADSS AFP Service as a Windows Service (Windows Control Panel/Administrative Tools/Services) under the name **Ascertia-AFP**:

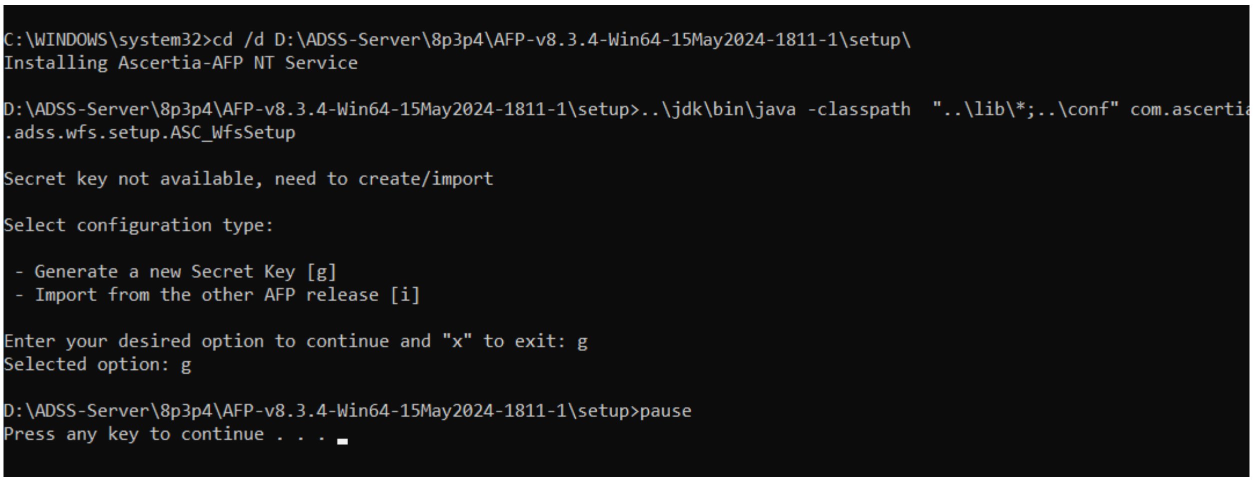


During the installation of the AFP service, a ***secret key*** will be generated by the ADSS Server using a secure algorithm such as AES or DES. This key will be used for encrypting and decrypting passwords. It is generated once during installation and utilized for various operations thereafter. For more details, navigate to this section.

If the key is not available at the time of installation, the user will be prompted to either generate a new secret key or import one from an older AFP version. Both the scenarios are displayed below:

Generate a new Secret Key

Clicking on the **install.bat** file will display the user the following screen:

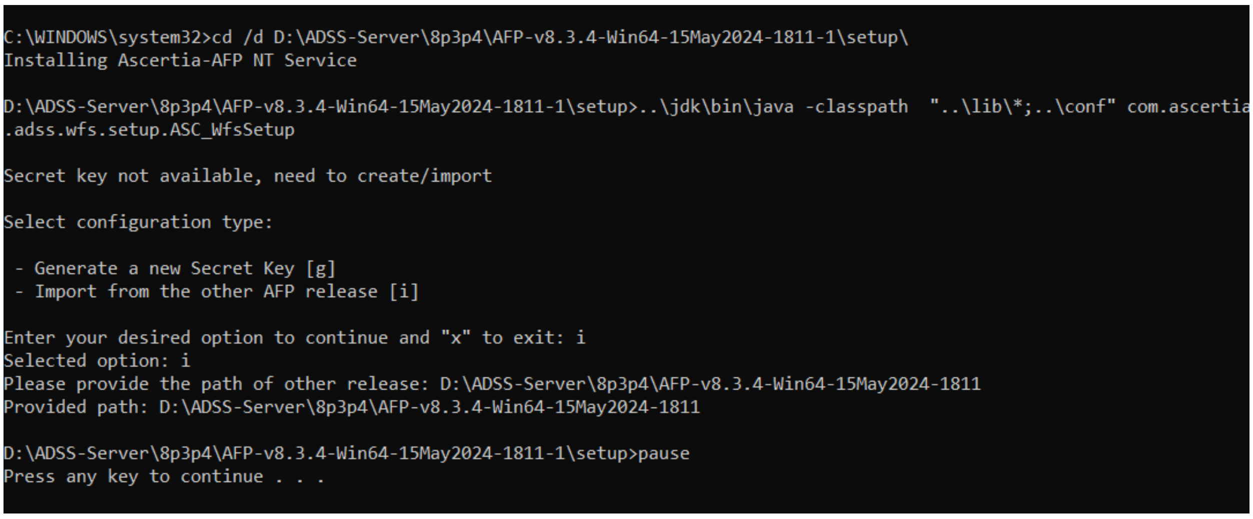


```
C:\WINDOWS\system32>cd /d D:\ADSS-Server\8p3p4\AFP-v8.3.4-Win64-15May2024-1811-1\setup\
Installing Ascertia-AFP NT Service

D:\ADSS-Server\8p3p4\AFP-v8.3.4-Win64-15May2024-1811-1\setup>..\jdk\bin\java -classpath  "..\lib\*;..\conf" com.ascertia
.adss.wfs.setup.ASC_WfsSetup

Secret key not available, need to create/import

Select configuration type:

 - Generate a new Secret Key [g]
 - Import from the other AFP release [i]

Enter your desired option to continue and "x" to exit: g
Selected option: g

D:\ADSS-Server\8p3p4\AFP-v8.3.4-Win64-15May2024-1811-1\setup>pause
Press any key to continue . . .
```

Enter the **'g'** option, a new secret key will be generated for user by the ADSS Server.

Import from the other AFP release

Clicking on the **install.bat** file will display the user the following screen:



```
C:\WINDOWS\system32>cd /d D:\ADSS-Server\8p3p4\AFP-v8.3.4-Win64-15May2024-1811-1\setup\
Installing Ascertia-AFP NT Service

D:\ADSS-Server\8p3p4\AFP-v8.3.4-Win64-15May2024-1811-1\setup>..\jdk\bin\java -classpath  "..\lib\*;..\conf" com.ascertia
.adss.wfs.setup.ASC_WfsSetup

Secret key not available, need to create/import

Select configuration type:

 - Generate a new Secret Key [g]
 - Import from the other AFP release [i]

Enter your desired option to continue and "x" to exit: i
Selected option: i
Please provide the path of other release: D:\ADSS-Server\8p3p4\AFP-v8.3.4-Win64-15May2024-1811
Provided path: D:\ADSS-Server\8p3p4\AFP-v8.3.4-Win64-15May2024-1811

D:\ADSS-Server\8p3p4\AFP-v8.3.4-Win64-15May2024-1811-1\setup>pause
Press any key to continue . . .
```

Select the **'i'** option, which will prompt you to enter the file path of the previous AFP release. Once you provide the file path, the secret key from the older AFP release will be imported.

To start, stop and restart the ADSS AFP service, launch the Windows services panel and select the **Ascertia-AFP** service and click on the **start/stop/restart** button accordingly.

*The ADSS AFP service can also be run by clicking on **afp.bat** file located at:*

[ADSS AFP Installation Directory]\bin

4.1.2 UNIX

To install ADSS AFP on UNIX systems the installer must be launched under root user privileges (otherwise ADSS AFP daemons will not be registered in **/etc/init.d**). Click here to read how to change the owner and group once the installation has completed. Use the following command to mark **install.sh** file as executable before launching:

```
# sh chmod +x install.sh
```

Following command will install the ADSS AFP as UNIX daemon under the name **wrapper.Ascertia-AFP**:

```
# sh install.sh
```

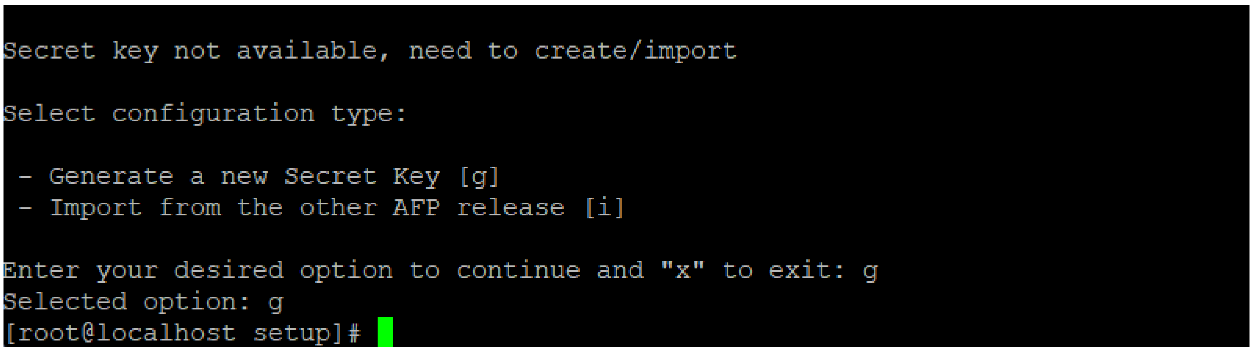
Use the following command to start, stop, restart or get the status of the ADSS AFP service accordingly:

```
# sh service wrapper.Ascertia-AFP start
```

As with Windows, the secret key mechanism is also applicable in UNIX. Users will be prompted to either generate a new secret key or import one from an older AFP version. Both scenarios are illustrated below:

Generate a new Secret Key

Clicking on the **install.sh** file will display the user the following screen:

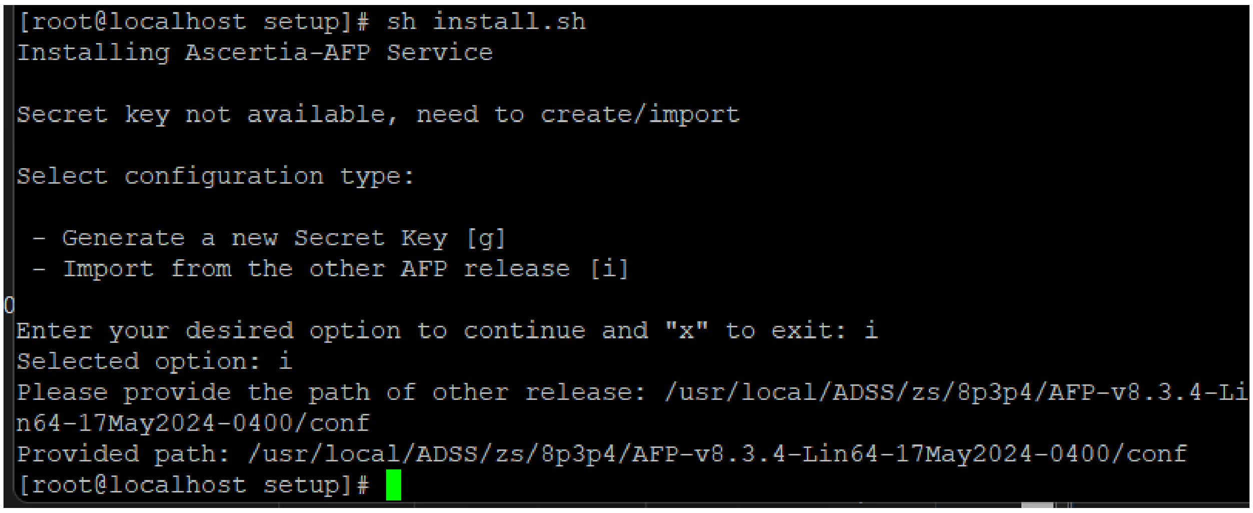


```
Secret key not available, need to create/import

Select configuration type:

 - Generate a new Secret Key [g]
 - Import from the other AFP release [i]

Enter your desired option to continue and "x" to exit: g
Selected option: g
[root@localhost setup]#
```

Enter the **'g'** option, a new secret key will be generated for user by the ADSS Server.

Import from the other AFP release

Clicking on the **install.sh** file will display the user the following screen:



```
[root@localhost setup]# sh install.sh
Installing Ascertia-AFP Service

Secret key not available, need to create/import

Select configuration type:

 - Generate a new Secret Key [g]
 - Import from the other AFP release [i]
0
Enter your desired option to continue and "x" to exit: i
Selected option: i
Please provide the path of other release: /usr/local/ADSS/zs/8p3p4/AFP-v8.3.4-Li
n64-17May2024-0400/conf
Provided path: /usr/local/ADSS/zs/8p3p4/AFP-v8.3.4-Lin64-17May2024-0400/conf
[root@localhost setup]#
```

Select the **'i'** option, which will prompt you to enter the file path of the previous AFP release. Once you provide the file path, the secret key from the older AFP release will be imported.

*The ADSS AFP service can also be run by executing **afp.sh** file located at:*

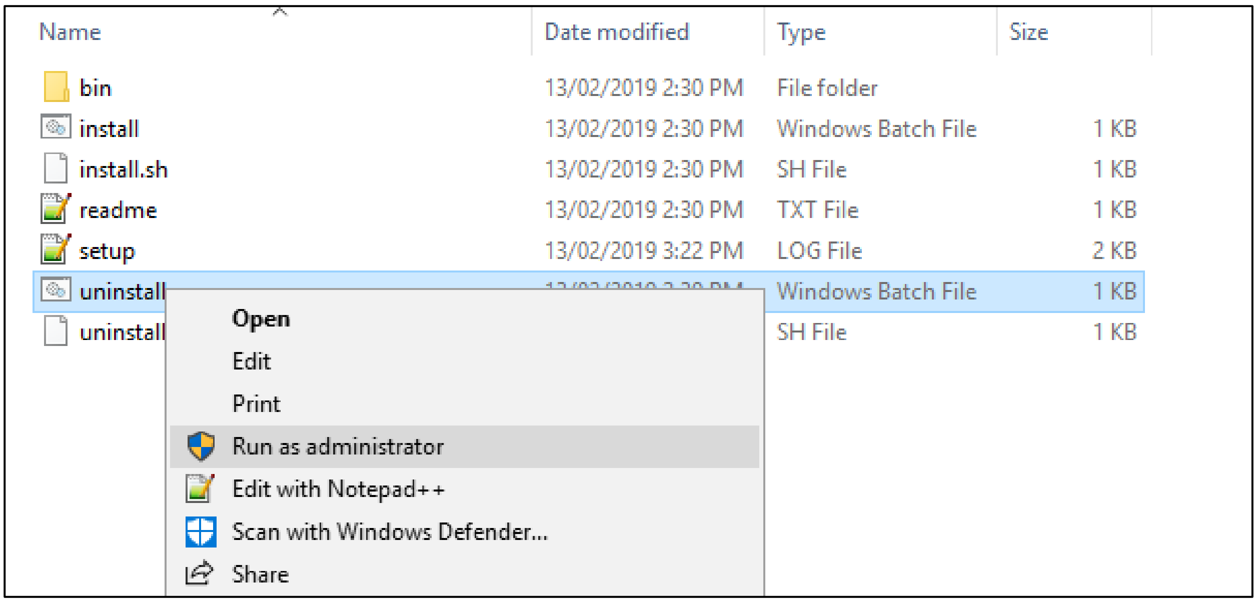
[ADSS AFP Installation Directory]/bin

4.2 Uninstall Instructions

Navigate to location **[ADSS AFP Installation Directory]\setup** and follow these steps to uninstall ADSS AFP on windows/Unix platforms:

Windows

Run the **uninstall.bat** file under administrative privileges, as shown below, (otherwise ADSS AFP services will not be removed from Windows Services Panel) to uninstall ADSS AFP service (Ascertia-AFP) from Windows Service panel.



Name	Date modified	Type	Size
bin	13/02/2019 2:30 PM	File folder	
install	13/02/2019 2:30 PM	Windows Batch File	1 KB
install.sh	13/02/2019 2:30 PM	SH File	1 KB
readme	13/02/2019 2:30 PM	TXT File	1 KB
setup	13/02/2019 3:22 PM	LOG File	2 KB
uninstall	13/02/2019 2:30 PM	Windows Batch File	1 KB
uninstall		SH File	1 KB

UNIX

Use the following command to mark **uninstall.sh** file as executable before launching:

```
# sh chmod +x uninstall.sh
```

Run the following command to uninstall the ADSS AFP service (wrapper.Ascertia-AFP) from UNIX daemon:

```
# sh uninstall.sh
```

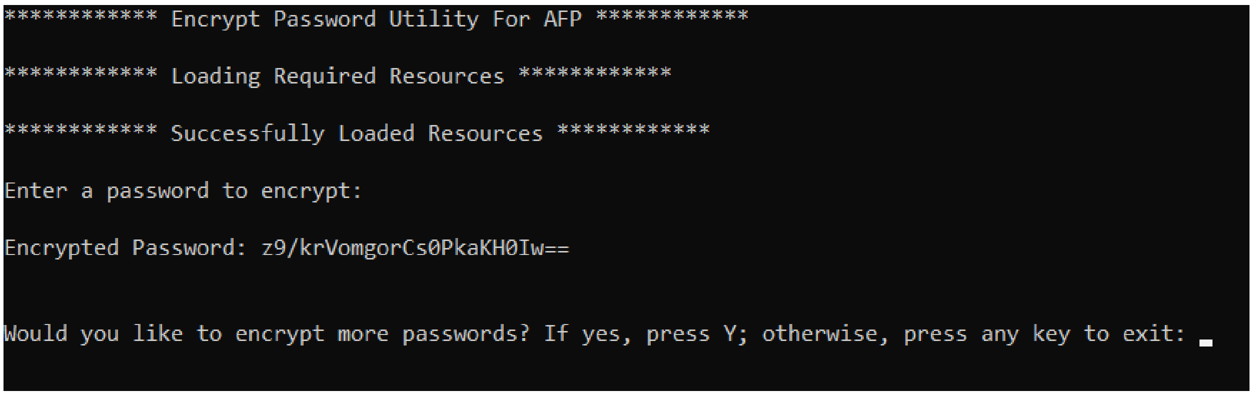
On all platforms the ADSS AFP folder structure must be deleted manually.

5 Encrypt Password in ADSS AFP

To encrypt the password in ADSS AFP, use the **encrypt_password** utility. The password is encrypted with the secret key generated or imported during the ADSS AFP installation. This utility is located in the [ADSS AFP Installation Directory]\bin folder.

Windows

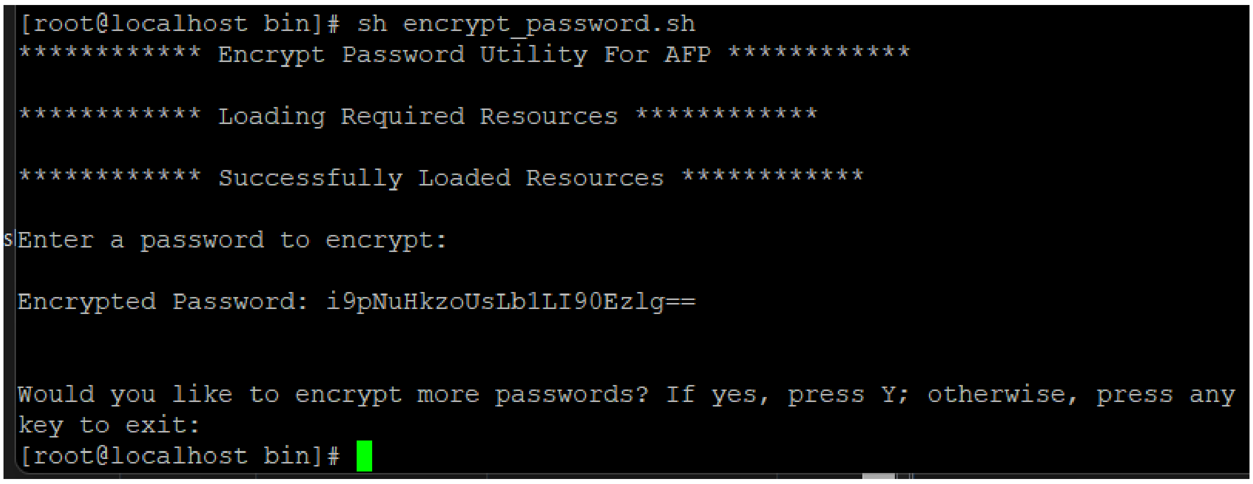
To run the utility on Windows, go to the [ADSS AFP Installation Directory]\bin folder and execute the encrypt_password.bat file. This will display the following screen:



```
************ Encrypt Password Utility For AFP ************

************ Loading Required Resources ************

************ Successfully Loaded Resources ************

Enter a password to encrypt:

Encrypted Password: z9/krVomgorCs0PkaKH0Iw==


Would you like to encrypt more passwords? If yes, press Y; otherwise, press any key to exit: _
```

UNIX

To run the utility on UNIX, go to the [ADSS AFP Installation Directory]\bin folder and execute the encrypt_password.sh file. This will display the following screen:



```
[root@localhost bin]# sh encrypt_password.sh
************ Encrypt Password Utility For AFP ************

************ Loading Required Resources ************

************ Successfully Loaded Resources ************

Enter a password to encrypt:

Encrypted Password: i9pNuHkzoUsLb1LI90Ezlg==


Would you like to encrypt more passwords? If yes, press Y; otherwise, press any
key to exit:
[root@localhost bin]#
```

The user enters its required password and receives an encrypted password generated by the utility. Once received, the user needs to store the encrypted password at a secure location.

The password will be decrypted while performing signing operation at the start of the AFP using the secret key generated at the time of ADSS AFP installation. If the password is decrypted successfully, then the request will be forwarded, else an error will be shown.

ADSS Auto File Processor (AFP) v8.3.5 can handle both encrypted and plain passwords. Therefore, when upgrading to ADSS AFP v8.3.5 from an earlier version, if the user prefers not to use the encrypted passwords, it is recommended that the encrypt_password utility must NOT be used.

5.1 List of Secured Tags

Below is the list of secured tag/fields where the user can provide plain or encrypted passwords:

- SigningCertificatePassword
- PdfProtectedPassword
- SigningHub > Password
- ProxyPassword
- PfxPassword
- PermissionsPassphrase

6 ADSS AFP Configuration

This section describes the ADSS AFP configurations. All the configurations are created using an XML based configuration file. This **afp.xml** configuration file is located at location: **[ADSS AFP Installation Directory]/conf**

*The ADSS AFP service only reads the **afp.xml** configuration file when it starts. ADSS AFP must therefore be restarted whenever a change is made to the afp.xml file*

6.1 Sample Profiles

Following sample ADSS AFP profiles are available in the ADSS AFP package at location **[AFP Installation Directory]/conf/samples/**:

1. **PDF Signing**
   For PDF signing there are three sample profiles available:
   - **PDF Signing with Server Hashing**
     Default profile to create PDF signatures by sending PDF documents to ADSS Server for signing. At location **[AFP Installation Directory]/conf/**, sample default profile already configured with name `afp.xml`
   - **PAdES Part 2 LTV signatures with Local Hashing**
     For use when you wish the PDF document to be hashed locally and the hash alone sent to ADSS Server for signing. Once the signed object is returned this is embedded into the PDF document by ADSS AFP to generate PAdES Part 2 LTV signatures. Use sample profile: `afp-LocalHashing_PAdES_Part2_LTV.xml`
   - **PAdES Part 4 LTV signatures with Local Hashing**
     For use when you wish the PDF document to be hashed locally and the hash alone sent to ADSS Server for signing. Once the signed object is returned this is embedded into the PDF document by ADSS AFP to generate PAdES Part 4 LTV signatures. Use sample profile: `afp-LocalHashing_PAdES_Part4_LTV.xml`
   - **PAdES Baseline-LT signatures with Local Hashing**
     For use when you wish the PDF document to be hashed locally and the hash alone sent to ADSS Server for signing. Once the signed object is returned this is embedded into the PDF document by ADSS AFP to generate PAdES Baseline Long Term signatures. Use sample profile: `afp-LocalHashing_PAdES_B_LT.xml`
   - **PAdES Baseline-LTA signatures with Local Hashing**
     For use when you wish the PDF document to be hashed locally and the hash alone sent to ADSS Server for signing. Once the signed object is returned this is embedded into the PDF document by ADSS AFP to generate PAdES Baseline Long Term Archive signatures. Use sample profile: `afp-LocalHashing_PAdES_B_LTA.xml`
2. **PDF Signature Field Creation**
   For creating blank signature fields inside a PDF document. Use the sample profile: `afp-PDFSignatureFieldCreation.xml`
3. **Document Conversion**
   For converting documents to PDF format. Use the sample profile: `afp-DocumentConversion.xml`
4. **XML Signing**
   For XML signing there are two sample profiles available:
   - **XML Signing with Server Hashing**
     Use for creating XML signatures by sending XML documents to ADSS Server for signing. Use sample profile: `afp-ServerHashing_XML.xml`
   - **XML Signing with Local Hashing**
     For use when you wish the XML document to be hashed locally and the hash alone sent to the ADSS Server for signing. Once the signed object is returned AFP embeds this into the XML document. Use sample profile: `afp-LocalHashing_XML.xml`

5. **File Signing**
   For file signing there are two sample profiles available:
   - **File Signing with Server Hashing**
     Use for creating file signatures by sending documents to ADSS Server for signing. Use sample profile: `afp-ServerHashing_FILE.xml`
   - **File Signing with Local Hashing**
     For use when you wish the File to be hashed locally and the hash alone sent to the ADSS Server for signing. Once the signed object is returned AFP embeds and creates the file signature. Use sample profile: `afp-LocalHashing_FILE.xml`

6. **MS Office Signing**
   For MS Office document native signing there are two sample profiles available:
   - **MSOffice Native Signing with Server Hashing**
     For sending MS Office documents to ADSS Server for native signing. Use sample profile: `afp-ServerHashing_MSOffice.xml`
   - **MSOffice Native Signing with Local Hashing**
     For use when you wish the MS Office document to be hashed locally and the hash alone sent to the ADSS Server for signing. Once the signed object is returned this is embedded into the MS Office document by AFP. Use sample profile: `afp-LocalHashing_MSOffice.xml`
7. **Document Sharing at SigningHub**
   Use for sharing/uploading documents to SigningHub Server. Use sample profile: `afp-DocumentSharing_SigningHub.xml`
8. **MIME Signing**
   Use for creating S/MIME by sending MIME messages to ADSS Server for signing (S/MIME). Use sample profile: `afp-ServerHashing_MIME.xml`

6.2 Running the Sample Profiles

In order to run a sample profile, other than default PDF signing, i.e. afp.xml, follow these steps:

- Stop ADSS AFP Service from Windows service or UNIX daemon.
- Go to location **[AFP Installation Directory]/conf/** and rename the default **afp.xml** file to another name of your choice.
- Go to location **[AFP Installation Directory]/conf/samples/** and rename the relevant sample profile to afp.xml, e.g. if it is required to run the **afp-LocalHashing_PAdES_Part2_LTV.xml** then rename this file to **afp.xml** and place it folder **[AFP Installation Directory]/conf/**.
- Start ADSS AFP Service from Windows service or UNIX daemon.

*For generating PAdES Part 4 LTV signatures using local hashing, please read the instructions inside the sample **afp-LocalHashing_PAdES_Part4_LTV.xml**.*

6.3 Profile Configurations

The following sections explain each part of **afp.xml** file. When reviewing these settings, it is important to note that some XML elements are required and some are optional. If an element is not required then you must (a) omit the tag, or (b) comment the tag. Note addresses cannot be left empty.

*Whenever the **afp.xml** file is changed the Ascertia-AFP service must be restarted to accept the changes.*

6.3.1 Global Configuration Settings

Global Settings deal with settings for authentication on ADSS Server and local network:

```
<AFP>
     <Settings>
          <OriginatorId>samples_test_client</OriginatorId>
          <ServiceAddress>
               <PrimaryAddress>http://localhost:8777/adss/signing/hdsi</Pri
               maryAddress>
               <SecondaryAddress>http://localhost:8777/adss/signing/hdsi</S
               econdaryAddress>
         </ServiceAddress>
         <ProxySettings status="disable">
               <ProxyHost>192.168.0.1</ProxyHost>
               <ProxyPort>8090</ProxyPort>
               <Credentials status="disable">
                    <ProxyUserName>user1</ProxyUserName>
                    <ProxyPassword>password1</ProxyPassword>
                    <AuthenticationScheme>BASIC</AuthenticationScheme>
               </Credentials>
         </ProxySettings>
         <ClientAuthPfxSettings status="disable">
                 <PfxFilePath>client.pfx</PfxFilePath>
                 <PfxPassword>password2</PfxPassword>
         </ClientAuthPfxSettings>
    </Settings>
```

If an element is not required, then the relevant element tag must either be omitted or commented. An empty value must not be used e.g.

```
<!-- <AuthenticationScheme>BASIC</AuthenticationScheme> -->
```

XML Tags	Description
AFP	AFP (Auto File Processor) is the root element.
Settings	The elements in between <Settings> and </Settings> tag are used for global configurations.
OriginatorId	Defines the client ID for this ADSS AFP and is included in the request messages which are sent to ADSS server. This client ID needs to be registered within the ADSS Client Manager module. Follow this link for more details: Client Manager
ServiceAddress	The elements in between <ServiceAddress> and </ServiceAddress> tag are used for defining ADSS Server primary and secondary addresses.
PrimaryAddress	Defines the primary ADSS Server address that will be used by default for all profiles unless a profile defines an alternative.
SecondaryAddress	Defines the secondary ADSS Server address that will be used by default for all profiles unless a profile defines an alternative.
ProxySettings	The elements in between <ProxySettings> and </ProxySettings> tag are used for defining the proxy settings. **Note**: It works only when status="enable".
ProxyHost	Defines the IP address or machine name of the proxy server.

XML Tags	Description
ProxyPort	Defines the port number for communication with the proxy host.
Credentials	Defines the user credentials for proxy settings. **Note:** It works only when status="enable".
ProxyUserName	Defines the user name for the proxy user authentication.
ProxyPassword	Defines the password for proxy user authentication.
AuthenticationScheme	Defines the authentication scheme. Possible values are: • BASIC • DIGEST
ClientAuthPfxSettings	The elements in between <ClientAuthPfxSettings> and </ ClientAuthPfxSettings> tag are used for defining the client authentication settings. **Note:** It works only when status="enable".
PfxFilePath	Defines the path of the private key to be used for client TLS authentication when communicating with ADSS Server.
PfxPassword	Defines the password for the client TLS authentication private key.

Table 2 - Global Configuration

If globally defined primary and secondary addresses are not required, then comment the element “<ServiceAddress>” instead of commenting the primary and secondary addresses individually. ***In this case ADSS Server address must be configured in each ADSS AFP profile.***

6.3.2 Common Configuration Settings

These are the Common Settings that apply to all ADSS AFP profiles:

- General settings
- Local hashing settings
- Concurrent request handling and file locking
- Connection settings
- Configuring file postfix
- Configuring file filters
- Configuring Profile based ADSS Server address

6.3.3 General Configuration Settings

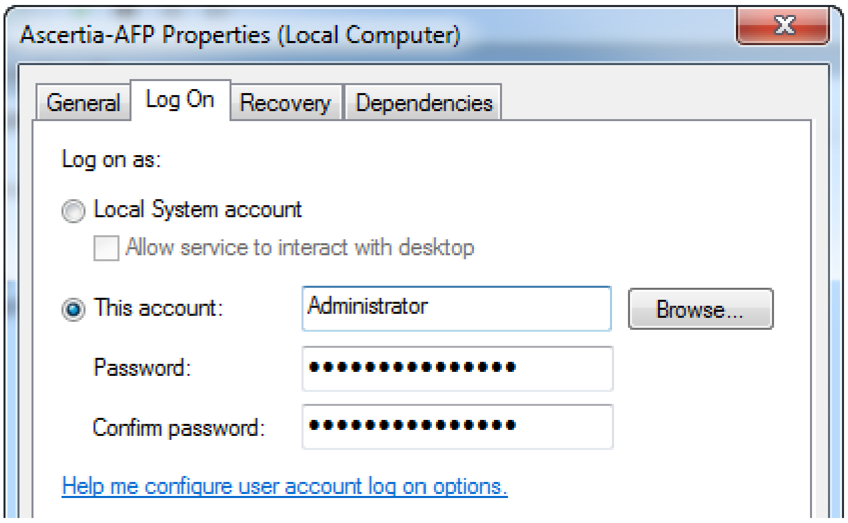
This example shows all possible General Settings that an ADSS AFP profile can have:

```
<Profile status="enable" localHash="false" type="signing"
name="ServerHashing1">
     <InputFolderPath>../data/input_folder</InputFolderPath>
     <OutputFolderPath>../data/output_folder</OutputFolderPath>
     <ErrorFolderPath>../data/error_folder</ErrorFolderPath>
     <ScanSubDirectories delete="false">FALSE</ScanSubDirectories>
     <SigningProfile>adss:signing:profile:001</SigningProfile>
     <SigningCertificate>samples_test_signing_certificate</SigningCertificat
     e>
     <SigningCertificatePassword>password12</SigningCertificatePassword>
```

If an element is not required, then the relevant element tag must either be omitted or commented. An empty value must not be used e.g.

```
<!--
<SigningCertificate>samples_test_signing_certificate</SigningCertif
icate> -->
```

XML Tags	Description
Profiles	The elements between <Profiles> and </Profiles> are used to define multiple profile configurations.
Profile	Defines a profile configuration that can include the following child nodes: • **status:** To enable or disable a profile • **localhash:** Set to true to enable local hashing • **Type:** To configure signing or PDF conversion or SigningHub • **name:** To identify the profile in debug logs (optional) **Note:** Multiple profile configurations can be defined here.

XML Tags	Description
InputFolderPath	Defines the path of the input folder. Files can be read and deleted from here. The folder path could either be from the local file system or from a shared network directory. If your files are on the network folder path, then note that the services on Windows OS are always installed with "Log On As" type "Local System", which doesn't necessarily have the access rights to the UNC (network) path. So you are required to change the "Log On As" user of Ascertia-AFP service to your user e.g. ".\Administrator" as shown in the following screenshot:  Ascertia-AFP Properties (Local Computer) Log On: Log on as: [✓ This account:] Administrator; Password: •••••••••••••••; Confirm password: ••••••••••••••• **Note:** Do not use "/" or "\" slashes at the end of the path as they will produce an error.
OutputFolderPath	This defines the path of the output folder. Successfully processed files are written here. Optionally a file locking extension can be specified i.e. <OutputFolderPath **lockExt**="ApplicationLock"> **Note:** Do not use "/" or "\" slashes at the end of the path as they will produce an error.
ErrorFolderPath	This defines the path of the error folder. Any files that cannot be processed successfully are placed here. Optionally a file locking extension can be specified i.e. <OutputFolderPath **lockExt**="ApplicationLock"> **Note:** Do not use "/" or "\" slashes at the end of the path as they will produce an error.
ScanSubDirectories	Set this flag to TRUE if you want to process the files in subdirectories of the Input Folder Path. Subdirectories will be created automatically in the Output/Error Folder Path. If this flag value is set to FALSE then files in the root folder will be processed only. By default, the value of this element is set to FALSE. **Note:** If the attribute **delete** value is set to TRUE e.g. delete="true" then subdirectories will get deleted once all the files get processed. By default, the value of this attribute is set to FALSE.

XML Tags	Description
SigningProfile {optional}	Identifies the Signing Profile Name/ID to be included in the request messages sent to ADSS Server. This Signing Profile Name/ID is defined within ADSS Signing Service module. Follow this link for more details: Step 4 - Configuring Signing Profile
SigningCertificate {optional}	Identifies the Signing Certificate alias to be included in the request messages that are sent to ADSS Server. If this alias is supplied, this overrides the default signing certificate configured in ADSS Server Signing Profile. Using Signing Certificate Alias, and using the same Alias every year is a good way of avoiding having to update the AFP file every time a signer certificate is updated. This parameter is distinct from the SignerCertificate parameter described in section 5.3.11 which identifies the specific signer certificate '.cer file' and thus supplies certificate specific information into the request. When the certificate is renewed, a new '.cer file' must be supplied to the client. **Note:** The signing certificate alias can be defined inside ADSS Server Key Manager or ADSS Server Certification Service. Follow these links for more details: • Service Keys • Certification Service
SigningCertificatePassword {optional}	Provide the signing certificate password if the key/certificate is generated in certification service using client password. By default, this element is commented. **Note:** This element is not required if the key/certificate is generated inside Certification Service without client password or defined in Key Manager (password not required in both cases).
PdfProtectedPassword {optional}	Specify the password in case of PDF is password protected

Table 3 - General Configuration

6.3.4 Local Hash Configuration Settings

The following example shows all possible common values that a local hashing profile can have:

```
      <HashAlgorithm>SHA256</HashAlgorithm>
      <SignaturePaddingScheme>PKCS1</SignaturePaddingScheme>
      <HashMode>Detached</HashMode>
      <SigDictionarySize>40</SigDictionarySize>
      <!—These settings are used only when generating PAdES/MS Office
      signatures-->
      <PAdESSignatureType>PAdES-LTV</PAdESSignatureType>
      <Verification>
           <VerificationProfile>adss:verification:profile:001</Verification
           Profile>
           <ServiceAddress>http://localhost:8777/adss/verification/dss</Ser
           viceAddress>
      </Verification>
      <TimeStamping>
           <TsaPolicy>1.2.3.4.5</TsaPolicy>
           <ServiceAddress>http://localhost:8777/adss/tsa</ServiceAddress>
      </TimeStamping>
```

If the Verification and TimeStamping elements are not mentioned, then the ADSS AFP uses the Primary/Secondary base URL for the Verification and TimeStamping Service.

If an element is not required, then the relevant element tag must either be omitted or commented. An empty value must not be used e.g.

```
<!-- <PAdESSignatureType>PAdES-LTV</PAdESSignatureType> -->
```

XML Tags	Description
HashAlgorithm	Identifies the hashing algorithm to be used for local hashing. Possible values are: • SHA1 • SHA256 • SHA384 • SHA512 • SHA3-224 • SHA3-256 • SHA3-384 • SHA3-512 **Note:** SHA3 hash algos are only applicable for XAdES, CAdES and Office Signatures with PSS Padding scheme and padding scheme not applicable with PAdES Signatures
SignaturePaddingScheme	Set the padding scheme for the signature to be used for local hashing. Possible values are: • PKCS1 • PSS **Note:** This is only applicable for XAdES, CAdES and Office Signatures
HashMode	Identifies the hashing mode to be used for local hashing. Possible values are: • ENVELOPING • ENVOLPED • DETACHED **Note:** In case of XAdES ETSI Baseline and Extended Signatures ADSS does not support for DETACHED signatures for local hashing.
SigDictionarySize	Identifies the signature dictionary size in KB to be allocated for local hashing. **Note:** A default value of 40 KB is recommended for most circumstances.

XML Tags	Description
PAdESSignatureType	Identifies the PAdES signature type to be created for local hashing purposes only. Possible values are: PAdES Baseline Signatures: • PAdES-B-B • PAdES-B-T • PAdES-B-LT • PAdES-B-LTA PAdES Extended Signatures: • PAdES-E-BES • PAdES-E-T • PAdES-E-LTV
Verification {optional}	The elements between <Verification> and </Verification> can be used to define ADSS Server Verification Service settings. Follow this link for more details: Verification Service **Note:** This element is required only when PAdES-LT, PAdES-LTV and PAdES-B-LTA values are configured in element PAdESSignatureType.
VerificationProfile {optional}	Identifies the Verification Profile Name/ID to be used for getting revocation information of the signing certificate in order to generate PAdES-LT, PAdES Part 4 LTV, PAdES-B-LTA or MS Office signatures other than XAdES EPES. This Verification Profile Name/ID is defined within the ADSS Verification Service module. Follow this link for more details: Step 4 - Configuring Verification Profile
ServiceAddress	The elements in between <ServiceAddress> and </ServiceAddress> tag are used for defining ADSS Server Verification Service address. **Note:** Only DSS Interface is supported for verification service.
TimeStamping {optional}	The elements between <TimeStamping> and </TimeStamping> can be used to define the TSA Service settings. **Note:** This element is required only when PAdES-B-LTA and PAdES-LTV values are configured in element PAdESSignatureType and also the Verification element is configured.
TsaPolicy {optional}	Identifies TSA Policy OID to be used for generating document timestamp in case of PAdES-B-LTA and PAdES Part 4 LTV signatures. This TSA policy OID is defined within ADSS Server TSA Service module. Follow this link for more details: Step 2 - Configuring TSA Profile
ServiceAddress	The elements in between <ServiceAddress> and </ServiceAddress> tag are used for defining the TSA Service address.

Table 4 - Local Hash Configuration

6.3.5 Concurrent Request Handling & File Locking Configuration Settings

ADSS AFP is able to read and process multiple files concurrently. The concurrent requests parameter defines how many files ADSS AFP reads and processes concurrently.

To ensure that other ADSS AFP instances cannot read and process the same set of files, a locking mechanism is used to reserve a batch of files for exclusive processing. Each of the selected filenames has the string "-afpinputlock" appended to it.

To ensure that other applications cannot start to process the signed files as they are written to the output folder a similar lock mechanism is used in this case the default string "-afplock" is added, although this can be changed using a parameter defined in a later section.

```
      <ConcurrentRequests>10</ConcurrentRequests>
      <BatchCount>30</BatchCount>
      <LockTimeout>120</LockTimeout>
```

XML Tags	Description
ConcurrentRequests	Defines the number of files that will be sent to ADSS Server in each 'batch' read operation. A default value of 10 is recommended for most circumstances. In order to achieve optimal performance, you can increase this value based on the average document size and type of signatures. **Note:** The concurrent request count should be less than the batch count otherwise concurrent request count will be ignored by the ADSS AFP and it will send all the locked files based on the configured value for batch count.
BatchCount	Defines number of files an ADSS AFP instance attempts to lock for processing. **Note:** A default value of 30 is recommended for most circumstances. In order to achieve optimal performance, you can increase this value base on the average document size.
LockTimeout	Defines the time interval (in seconds) for which ADSS AFP custom lock is considered valid by other ADSS AFP instances working in load-balanced mode. Each ADSS AFP instance attempts to remove this lock if lock timeout has elapsed. **Note:** A default value of 120 seconds is recommended for most circumstances.

Table 5 - Concurrent Request Handling & File Locking Configuration

6.3.6 ADSS Server Connection Configuration Settings

The next set of configurations deal with ADSS AFP to ADSS Server communication:

```
      <IdleSleepTime>30</IdleSleepTime>
      <ConnectionTimeout>30</ConnectionTimeout>
      <FailureRetries>3</FailureRetries>
```

XML Tags	Description
IdleSleepTime	When ADSS AFP is running it will continue to search for and process files until no more are available. At this time, it will sleep for the configured number of seconds set by this parameter in order to save client system CPU time. **Note:** A default value of 30 seconds is recommended for most circumstances.
ConnectionTimeout	If the connection between ADSS AFP and ADSS Server fails, it will timeout after this number of seconds.

XML Tags	Description
	Note: A default value of 30 seconds is recommended for most circumstances.
FailureRetries	If the connection between ADSS AFP and ADSS Server fails, then ADSS AFP will try to re-establish the connection this many times. **Note:** A default value of 3 is recommended for most circumstances.

Table 6 - ADSS Server Connection Configuration

6.3.7 File Postfix Configuration Settings

The next set of configurations deal with file postfix:

```
<FilePostfix>-processed</FilePostfix>
```

XML Tags	Description
FilePostfix	Defines an optional postfix string to be added to the processed filename e.g. • To add "-s" to a file name use: <FilePostfix>-s</FilePostfix> • To leave the filename unchanged use: <FilePostfix></FilePostfix>

Table 7 - File Postfix Configuration

6.3.8 Signed attributes Configuration Settings

The next set of configurations deal with configuring the signed attributes for the signatures to be produced:

```
<AddSigningTime>TRUE</AddSigningTime>
<DocumentFormat>1.2.3.4</DocumentFormat>
```

XML Tags	Description
AddSigningTime	Set this flag to TRUE if you want to add signing time in case of Extended Signature. **Note:** signingTime will be implicitly added in the signature dictionary in case of Baseline Signature either this flag is set to TRUE or FALSE.
DocumentFormat	Defines an optional document format OID/String to be added as a signed attribute in the signature. **Note:** For CAdES signatures the document format should be an OID value and for XAdES signatures the document format should be a string value according to CAdES/XAdES specifications. Document format cannot be included in PAdES signatures as per PAdES specifications.
CommitmentTypeIdentifier	It is an optional identifier that provides additional information for signatures where possible values includes: - `ProofOfOrigin` - `ProofOfReceipt`

XML Tags	Description
	- `ProofOfDelivery` - `ProofOfSender` - `ProofOfApproval` - `ProofOfCreation`
CommitmentTypeQualifier	It is an optional attribute where the user can define its own value according to its need. If the attribute is set blank, then, the value set in the CommitmentTypeIdentifier attribute will be copied here automatically.
MimeType	It is a signing attribute. If we create a XAdES Baseline Signature of the type XAdES-B-B, then, it is necessary to set the value of MimeType attribute in XML file. Else, If MimeType is set empty, the value set in DocumentFormat attribute will be automatically copied here. Possible Value: - `text/xml`

Table 8 - Signed Attributes Configuration

6.3.9 File Filter Configuration Settings

The next set of configurations deal with file filters:

```
      <InputFileFilter>
            <FileExtension>pdf</FileExtension>
            <FileNameContains>UK</FileNameContains>
      </InputFileFilter>
```

XML Tags	Description
InputFileFilter	Defines Profile filter criteria.
FileExtension	Defines the type of file that ADSS AFP will process based on the file extension/content type e.g. pdf, doc, xml, mime etc. If mime is specified in this element, then AFP picks the MIME messages from the input folder and get S/MIME from ADSS Signing Server. **Note:** If it is required to process multiple file extensions then user should create multiple AFP profiles to handle each extension separately.
FileNameContains	Defines the type of file that ADSS AFP will process based on the text contained in the file name.

Table 9 - File Filter Configuration

6.3.10 Configuring Profile Based ADSS Server Address Configuration Settings

It is possible to define profile specific ADSS Server addresses overriding ADSS Server address defined in ADSS AFP global settings.

Profile base ADSS Server address is not required when creating signature field only, converting non-PDF documents to PDF or sharing document to SigningHub.

```
     <ServiceAddress>
          <PrimaryAddress>http://localhost:8777/adss/signing/hdsi</Primary
          Address>
          <SecondaryAddress>http://localhost:8777/adss/signing/hdsi</Secon
          daryAddress>
     </ServiceAddress>
```

XML Tags	Description
ServiceAddress	The elements in between <ServiceAddress> and </ServiceAddress> tag are used for defining ADSS Server primary and secondary addresses.
PrimaryAddress	Defines the primary ADSS Server address for this profile only.
SecondaryAddress	Defines the secondary ADSS Server address for this profile only.

Table 10 - Profile Based ADSS Server Address Configuration

*If primary and secondary addresses are not required to be defined in a profile (because the globally defined values are to be used) then remove or comment the root element **<ServiceAddress>** instead of commenting the primary and secondary addresses individually.*

6.3.11 PDF Visible Signature Configuration Settings

The next set of configurations deal with PDF visible signature settings:

```
<PdfSignatureSettings>
     <!--SigningPosition and SigningArea must not be used simultaneously
     in a   profile-->
     <SigningPosition>X1=0,Y1=0,X2=200,Y2=100</SigningPosition>
     <SigningArea>TOP_LEFT</SigningArea>
     <SignatureField>Signature1</SignatureField>
     <SignatureAppearanceID>default_sig_appearance</SignatureAppearanceID
     >
     <SigningPage>1</SigningPage>
      <SigningReason>I am author of this document</SigningReason>
      <SigningLocation>London</SigningLocation>
      <ContactInfo>support@ascertia.com</ContactInfo>
      <CompanyLogo>../data/resources/company_logo.png</CompanyLogo>
      <HandSignature>../data/resources/hand_signature.jpg</HandSignature>
      <!— These settings are only used when local hashing is needed and
     enabled -->
     <SignedBy>John Doe</SignedBy>
     <CertifyPermission>CERTIFIED_NO_CHANGES_ALLOWED</CertifyPermission>
      <FontRepository>../data/resources/fonts</FontRepository>
     <SignerCertificate>../data/resources/samples_test_signing_certificat
     e.cer</SignerCertificate>
     <SignatureAppearance>../data/resources/default_sig_appearance.xml</S
     ignatureAppearance>
     <LockPdfAfterSigning>TRUE</LockPdfAfterSigning>
</PdfSignatureSettings>
```

If an element is not required, then the relevant element tag must either be omitted or commented. An empty value must not be used e.g.

`<!-- <ContactInfo>support@ascertia.com</ContactInfo> -->`

XML Tags	Description
PdfSignatureSettings	The elements between < PdfSignatureSettings > and </ PdfSignatureSettings > can be used to define PDF signature appearance settings that override the settings configured in ADSS Server Signing Profile.
SigningPosition {optional}	Defines the location (in terms of XY-Coordinates) on the PDF document where signature field is to be placed. **Note:** SigningPosition and SigningArea must not be used simultaneously in a profile.
SigningArea {optional}	Defines the location on the PDF document where a signature is to be placed. Possible values are: • TOP_LEFT • TOP_RIGHT • CENTER • BOTTOM_LEFT • BOTTOM_RIGHT **Note:** SigningPosition and SigningArea must not be used simultaneously in a profile.
SignatureField {optional}	Defines the name of the target (empty) signature field to be signed. **Note:** Mandatory if SigningPosition is used.
SignatureAppearanceID {used for Server hashing only}	Defines the signature appearance id configured in ADSS Server Signing Service. **Note:** Mandatory if SigningPosition is used.
SignatureAppearance {used for local hashing only}	Defines the full path of the signature appearance file. The appearance of the signature is controlled by the configurations in the signature appearance file. **Note:** Mandatory if creating visible PDF signatures.
SigningPage {optional}	Defines the page number of the PDF document where signatures are placed. Permitted values are 1 to the maximum page number of the document being processed. **Note:** Mandatory if SigningPosition is used.
SigningReason {optional}	Defines the “signing reason” text added within the visible signature.
SigningLocation {optional}	Defines the “signing location” text added to the visible signature.
ContactInfo {optional}	Defines the “contact” text added to the visible signature.
CompanyLogo {optional}	Defines the full path of the company logo image to be added to the visible signature.
HandSignature {optional}	Defines the full path of the hand signature image to be added to the visible signature.

XML Tags	Description
SignedBy {used for local hashing only}	Optionally defines the text to be inserted in the "Signed By" field of the visible signature.
CertifyPermission {used for local hashing only}	Optionally defines the permissions for Certify signing. The permitted values are: • CERTIFIED_NO_CHANGES_ALLOWED • CERTIFIED_FORM_FILLING • CERTIFIED_FORM_FILLING_AND_ANNOTATIONS **Note:** When generating PAdES_LT, PAdES_LTV and PAdES_B_LTA signatures in case of local hashing, then certify permission `CERTIFIED_NO_CHANGES_ALLOWED` will be ignored.
FontRepository {used for local hashing only}	Optionally defines the full path for the font to be used if the font needs to be embedded in the target document.
SignerCertificate	The XML tag covers the below mentioned scenarios: • **Local Hashing** Defines the full path of the signing certificate. By default, the common name of this certificate would be used in the "Signed By" field of the visible signature. • **Absence of Signing Certificate on Signing Service instance** If the user keys and certificates are not available at the Signing Service instance, then we must include a signer certificate parameter in the AFP.xml file to provide the certificate with the request.
LockPdfAfterSigning {optional}	This tag enables the user to lock the entire PDF document after the final signature.

Table 11 - PDF Visible Signature Configuration

If it is required to generate invisible PDF signatures, then for client side hashing, remove all child elements of ***PdfSignatureSettings*** *except the* ***SignerCertificate*** *element*

6.3.12 PDF Protection Configuration Settings

The next set of configurations deal with PDF protection settings:

```
<PdfProtection>
     <!-- Optionally specify PDF data protection settings in the case of
     local hashing only. Note when sending the complete document to ADSS
     Server these permissions can be applied via signature profile settings
     on the server. -->
     <PermissionsPassphrase> password12</PermissionsPassphrase>
     <AllowPrinting>
          <Resolution>LOW</Resolution>
     </AllowPrinting>
     <AllowModification>TRUE</AllowModification>
     <AllowCopyingAndExtraction>TRUE</AllowCopyingAndExtraction>
     <AllowDocumentAssembly>TRUE</AllowDocumentAssembly>
     <AllowTextAccess>TRUE</AllowTextAccess>
     <AllowFormFilling>TRUE</AllowFormFilling>
     <AllowCommenting>TRUE</AllowCommenting>
</PdfProtection>
```

If an element is not required, then the relevant element tag must either be omitted or commented. An empty value must not be used e.g.

```
<!-- <AllowModification>TRUE</AllowModification> -->
```

XML Tags	Description
PdfProtection {optional}	The elements between <PdfProtection> and </PdfProtection> can be used to define the document level permissions on PDF files as defined in ISO 32000 specification. Note: Currently these permission settings are not supported for PAdES part 4 signatures.
PermissionsPassphrase {optional}	Defines the passphrase that will be set on the PDF document to change the permissions of this document. e.g. <PermissionsPassphrase>password12</PermissionsPassphrase>
AllowPrinting {optional}	Defines whether user is permitted to print the document or not. The allowed values are TRUE or FALSE.
Resolution {optional}	Defines the resolution level to be allowed for document printing. Possible values are: • LOW • HIGH
AllowModification {optional}	Defines whether the user is permitted to modify the contents e.g. to change the content of a page, or insert or remove a page. The allowed values are TRUE or FALSE.
AllowCopyingAndExtraction {optional}	Defines whether the user is permitted to insert, remove, and rotate pages and add bookmarks. The allowed values are TRUE or FALSE. Note: The content of a page can't be changed unless the permission Allow content to be modified is granted too.
AllowDocumentAssembly {optional}	Defines whether the user is permitted to copy or otherwise extract text and graphics from the document, including using assistive technologies i.e. screen readers or other accessibility devices. The allowed values are TRUE or FALSE.

XML Tags	Description
AllowTextAccess {optional}	Defines whether the user is permitted to extract text and graphics for use by accessibility devices. The allowed values are TRUE or FALSE.
AllowFormFilling {optional}	Defines whether the user is permitted to fill form fields (for 128-bit encryption only). The allowed values are TRUE or FALSE.
AllowCommenting {optional}	Defines whether the user is permitted to add or modify text annotations and interactive form fields. The allowed values are TRUE or FALSE.

Table 12 - PDF Protection Configuration

6.3.13 MS Office Visible Signature Configuration Settings

The next set of configurations deal with MS Office document signatures. Currently only MS Word document native signing is supported. For more details about the MS office signing follow this link:

Microsoft Office Signing Attributes

```
<OfficeSignatureSettings>
     <SignatureLine>info@ascertia.com</SignatureLine>
     <HandSignature>../data/resources/hand_signature.jpg</HandSignature>
     <!— These settings are only used when local hashing is needed and
     enabled -->
     <SignatureFormat>XAdES-T</SignatureFormat>
     <SignHash>FALSE</SignHash>
     <SignerCertificate>../data/resources/samples_test_signing_certificat
     e.cer</SignerCertificate>
     <!-- SignerCertificateChain and IssuerCertificate must not be used
     simultaneously in a profile-->
     <SignerCertificateChain>../data/resources/samples_test_signing_certi
     ficate.p7b</SignerCertificateChain>
     <IssuerCertificate>../data/resources/samples_test_ca.cer</IssuerCert
     ificate>
</OfficeSignatureSettings>
```

XML Tags	Description
OfficeSignatureSettings	The elements between < OfficeSignatureSettings > and </ OfficeSignatureSettings > can be used to define settings for MS Office native signature creation.
SignatureLine	Defines the signature line in the MS Office document. It holds either: • Signature Line Id e.g. {39F43AA2-8A4B-45D2-BA1D-8A5A2C6706BB} • Or suggested signer email e.g. **JohnDoe@ascertia.com**
HandSignature {Optional}	Defines the full path of the hand signature image to be added to the signature.
SignatureFormat {used for local hashing only}	This defines the format of the Office document XAdES signature. Following are the possible values. • XAdES-EPES • XAdES-T

XML Tags	Description
	• XAdES-C • XAdES-X-TYPE-1 • XAdES-X-L-TYPE-1 **Note:** Mandatory for local hashing only.
SignHash {used for local hashing only}	Defines whether ADSS AFP performs local hashing of the data to be signed. The allowed values are TRUE or FALSE. **Note:** Mandatory for local hashing only.
SignerCertificate	The XML tag covers the below mentioned scenarios: • **Local Hashing** Defines the full path of the signing certificate. By default, the common name of this certificate would be used in the "Signed By" field of the visible signature. • **Absence of Signing Certificate on Signing Service instance** If the user keys and certificates are not available at the Signing Service instance, then we must include a signer certificate parameter in the AFP.xml file to provide the certificate with the request.
SignerCertificateChain {used for local hashing only}	Defines the full path of the signing certificate chain. **Note:** Mandatory for local hashing only. SignerCertificateChain and IssuerCertificate must not be used simultaneously in a profile
IssuerCertificate {used for local hashing only}	Defines the full path of the signing certificate issuer. **Note:** Mandatory for local hashing only. SignerCertificateChain and IssuerCertificate must not be used simultaneously in a profile

Table 13 - MS Office Visible Signature Configuration

6.3.14 PDF Signature Field Creation Configuration Settings

ADSS AFP can create one or more empty signature fields within a PDF document. The location of each field and its size is defined using the page number and XY coordinates on this.

```
<PdfEmptySignatureFieldSettings>
     <EmptySignatureField>X1=100,Y1=100,X2=300,Y2=200,page=1,name=Signatu
     re1</EmptySignatureField>
</PdfEmptySignatureFieldSettings>
```

XML Tags	Description
PdfEmptySignatureFieldSettings	The elements in between <PdfEmptySignatureFieldSettings> and </PdfEmptySignatureFieldSettings> tag are used to define empty signature fields to be created in the input PDF document.
EmptySignatureField	Defines the empty signature field. Can be repeated as per need e.g. <EmptySignatureField>X1=100,Y1=100,X2=300,Y2=200,page=1,name=Signature1</EmptySignatureField>. Note bottom left of the PDF page is considered origin of XY-Coordinates. Repeat <EmptySignatureField> to create multiple fields but value for 'name' must be unique per profile.

Table 14 - PDF Signature Field Creation Configuration

6.3.15 XML Signature Configuration Settings

The next set of configurations deal with XML document signatures.

```
<XmlSignatureSettings>
     <UriAttributeIdentifier>Book-1</UriAttributeIdentifier>
</XmlSignatureSettings>
```

XML Tags	Description
XmlSignatureSettings	The elements between <XmlSignatureSettings> and </XmlSignatureSettings> can be used to define settings for XML signature creation.
UriAttributeIdentifier	The URI attribute identifies a data object using a URI-Reference. This setting Defines Reference URI attribute in XML signatures e.g. book-1.
SignatureFormat {used for local hashing only}	This defines the format of the XML document XAdES signature. Following are the possible values. XAdES Signatures (ETSI TS 101 903): • `XAdES-BES` • `XAdES-T` • `XAdES-C` • `XAdES-X` • `XAdES-X-L` • `XAdES-A` XAdES Baseline Signatures (EN 319 132-1): • `XAdES-B-B` • `XAdES-B-T` • `XAdES-B-LT` • `XAdES-B-LTA` XAdES Extended Signatures (EN 319 132-2): • `XAdES-E-BES` • `XAdES-E-T` • `XAdES-E-C` • `XAdES-E-X` • `XAdES-E-X-L` • `XAdES-E-A` **Note:** Mandatory for local hashing only.

Table 15 - XML Signature Field Configuration

6.3.16 Document Conversion Configuration Settings

For conversion of documents to PDF, the profile type should be set to "**conversion**" and the input file filter should specify one of the following types.

- .doc
- .docx
- .xls
- .xlsx
- .ppt
- .pptx
- .odt
- .rtf
- .txt
- .ods
- .csv
- .tsv
- .tif

ADSS AFP also supports the PDF/A compliant document generation. Use this xml tag for conversion profile:

```
<PdfAFormat>1a</PdfAFormat>
```

XML Tags	Description
PdfAFormat	This is used for defining the PDF/A format for document conversion. The supported PDF/A formats are: • 1a • 1b • 2a • 2b • 3b

Table 16 - Document Conversion Configuration

6.3.17 SigningHub Document Upload Configuration Settings

The next set of configurations deals with document uploading to SigningHub and associating a template to it and then trigger a workflow.

For information on how to configure SigningHub follow this link:
https://docs.ascertia.com/signinghub-admin/integration/thirdpartyappintegrationkeys

```
<SigningHubSettings>
     <ApiURL>https://api.signinghub.com</ApiURL>
     <ClientID>ClientID</ClientID>
     <ClientSecret>YexTeEx7drf84RT4fgtGHT987qpzs349erfr67</ClientSecret>
     <UserName>JohnDoe@ascertia.com</UserName>
     <Password>password123</Password>
     <Scope>bob@ascertia.com</Scope>
     <ConvertDocument>false</ConvertDocument>
     <Source>AFP</Source>
     <TemplateName>Sample Template</TemplateName>
</SigningHubSettings>
```

XML Tags	Description
SigningHubSettings	The elements between <SigningHubSettings> and </SigningHubSettings> can be used to define settings for sharing/uploading documents to SigningHub Server.
ApiURL	Defines API URL for SigningHub.
ClientID	Defines Application Name that has been configured in Enterprise Account.
ClientSecret	Defines API Key that has been generated in Enterprise Account.
UserName	Defines SigningHub email address identifying the target user account. **Note:** This can be an Enterprise Admin or an Enterprise User account.
Password	Defines the password for the user account.
Scope {Optional}	Define this element when Enterprise Admin user is identified in ‘username’ and it is required to identify an Enterprise child user to be owner of the document being shared. This can be needed when Enterprise Admin configuring ADSS AFP does not have credentials of Enterprise child user. **Note:** If the business application directly identifies Enterprise child user and their password is required then the Scope parameter is not required.
ConvertDocument {Optional}	Defines whether to convert the document to a PDF or if it should be retained in its original format.
Source	Defines the source of the document from where the document is being uploaded e.g. “ADSS AFP”. The user is free to define any ‘string’ value for this. It is only used as a reference.
TemplateName	Defines the name of the template to be applied e.g. User_Template.

Table 17 - SigningHub Document Upload Configuration

7 Running ADSS AFP Over TLS Client Authentication

ADSS AFP supports communication with ADSS Server over TLS client authentication. This ensures ADSS AFP instance is uniquely identified within ADSS Server, and the instance must authenticate itself to ADSS Server before using services. The configuration requires set-up on the target ADSS Server and once complete, adding the necessary configuration to afp.xml file. Follow this KB article for configurations on both ADSS AFP and ADSS Server:

https://ascertia.my.site.com/partners/s/article/How-to-set-up-TLS-Client-Authentication-communication-with-ADSS-Services

Once configured ensure the ClientAuthPfxSettings is enabled and correctly configured. These settings are found under the Global Settings section 6.3.1.

ADSS AFP service must be restarted once the configuration changes have been made.

8 Upgrading ADSS AFP

Follow these instructions to upgrade any older version of ADSS AFP to the latest one:

1. Stop and uninstall the old ADSS AFP (section uninstalling ADSS AFP Server).
2. Download and extract the latest package of ADSS AFP. Note do not overwrite the current ADSS AFP file set with the latest version.
3. Copy the **afp.xml** file from location: **[Old ADSS AFP Installation Directory]/conf** and overwrite it at location: **[New ADSS AFP Installation Directory]/conf.**
4. Install the new ADSS AFP (section Installation Instructions).
5. If the ADSS Server is secured by TLS (standard in production systems) then it is required to copy the **jssecacerts** file from location: **[Old ADSS AFP Installation Directory] /jre/lib/security** and overwrite it at the corresponding new location: **[New ADSS AFP Installation Directory]/jre/lib/security**.
6. Start the ADSS AFP from the Windows NT Services Panel / UNIX Daemon. Note the new service will point at the new directory structure, and ultimately, new afp.sh or afp.bat file.

9 Effective Memory Management for Best Performance

The better your JVM performs, the better your installation of AFP will perform. It's as simple as that. Getting the most out of your JVM is a matter of configuring its settings to match your real-world performance needs as closely as possible. Establish some accurate benchmarks so you have a way of quantifying any changes you make, and then get down to business.

The main thing to consider when tuning your AFP Server JVM for best performance is how to avoid wasting memory and draining your server's power to process requests. Certain automatic JVM processes, such as garbage collection and memory reallocation, can chew through memory if they occur more frequently than necessary. AFP Server make sure these processes only occur when they need to by using the -Xmx and -Xms switches to control how JVM handles its heap memory.

Our recommendation is to always set -Xms512m and then set -Xmx to 1024m if you are not hashing locally and not using large documents or data. If hashing locally or using very large documents or data then set this to 2048m (you may increase it as per your requirement) but check your system has this memory available.

Follow these instructions to configure the memory parameters for optimal performance:

1. Go to location **[AFP Installation Directory]/conf/**
2. Edit the **wrapper.conf** file in a text editor
3. Change the **wrapper.java.initmemory** and **wrapper.java.maxmemory** switches values accordingly (depends upon available RAM, average document size and though put requirement)
4. Restart the AFP from Windows Services panel or UNIX daemon.

*If the ADSS AFP service is not installed or registered in Windows service panel or as UNIX daemon and it is being run directly by executing the **afp.bat/sh** file then follow these instructions to configure the memory parameters for optimal performance:*

1. *Stop the AFP*
2. *Go to location [AFP Installation Directory]/bin/*
3. *Edit the afp.bat/sh file in a text editor*
4. *Change the –Xms and –Xmx switches values accordingly (depends upon available RAM, average document size and though put requirement)*
5. *Run the AFP.bat/sh file again*

*** End of document ***