ADSS Signing Service

Table of Contents

How can I embed a font when signing PDF documents?

In order to embed a font used in the PDF signature appearance within the signed document, modify the signing profile in use by going to the **Signature Settings** tab and turn on the option **Embed font to be used for PDF signature appearance text objects**.

Click here for more details.

How to prevent the signing service from database bloating?

A small increase in the database size is expected from each transaction because a record of every request and response is stored within the Signing Service transaction logs. Go to the Signing Service > **Service Manager** module to check that the functionality to store a copy of the input and output documents is not selected. **Click here** for more details.

To minimize database bloating by the ADSS Server Tomcat logs, refer to the **Managing Disk Space** section.

What are authorisation profiles and how do they work?

Authorisation profiles are used to specify the list of authorisers (i.e. registered end-users) who can provide authorisation to sign one or more documents using a specific document signing key held within the ADSS Server database or HSM.

Authorised signing is especially effective when used to protect or provide a wilful act when signing with a high trust qualified certificate or Adobe rooted certificate held within an HSM connected to ADSS Server. This also provides strong internal audit evidence of sign-off and approval for signing of important documents and assures the documents have not changed from the first to the last authorising signature.

For details on how to configure an Authorisation Profile, **click here**

How can I configure the PDF signature dictionary size?

When signing PDF documents, some space is reserved within the document to embed the signature and revocation information. If it is required to adjust the PDF signature dictionary size, then modify the signing profile in use by going to the **Advanced Settings** tab and configure the desired value against the option **PDF Signature Space Allocation**. The default value is set to 40 KB. **Click here** for more details.

If computed signature size is larger than the configured signature dictionary size then the user may face the error “signature dictionary size 39062” bytes are smaller than the expected size “e.g. 45221” bytes. If this error is faced then adjust the PDF signature dictionary size for the signing profile in use.

The space allocation algorithm works as follows:

For PDF Signatures:

Standard PDF signature – The PDF document size increases by the space allocated after signing. e.g. if the space allocated is 20KB then PDF document size is increased by 20KB.

PDF signature with embedded timestamp – The PDF document size increases by the space allocated after signing. e.g. if the space allocated is 20KB then PDF document size is increased by 20KB.

PDF signature with embedded timestamp and revocation information – The size of the PDF document increases based on the following factors:

- Allocated space
- Size of the revocation information of the signer's complete certificate chain

e.g. if the space allocated is 20KB and size of the revocation information of the signer's certificate chain is 40KB then the PDF document size is increased by 60KB after signing.

For PAdES Signatures:

PAdES-BES (Basic PAdES Part 3 Signature) – The PDF document size increases by the space allocated after signing. e.g. if the space allocated is 20KB then PDF document size is increased by 20KB.

PAdES-T (Basic PAdES Part 3 Signature with embedded timestamp) – The PDF document size increases by the spaced allocated after signing. e.g. if the space allocated is 20KB then PDF document size is increased by 20KB.

PAdES-LTV (PAdES Part 4, PAdES-T Signature with embedded validation information) – The size of the PDF document increases based on the following factors:

- Allocated space
- Size of the revocation information of the signer's complete certificate chain

e.g. if the space allocated is 20KB and size of the revocation information of the signer's certificate chain is 40KB then the PDF document size is increased by 60KB after signing.

Note:- For PDF/PAdES Hash Profiles, the business application should manage the signature dictionary size while generating the respective signature type. If business application is using ADSS Client-SDK then **PdfSigningRequest class** provides a method **setSignatureDictionarySize** to set the signature dictionary size.

Changing an existing PDF signature appearance aspect ratio and size

PDF signature appearance image size and aspect ratio can be changed by following the below given instructions:

1. Launch the ADSS Server Console
2. Go to location **Signing Service > PDF Signature Appearances**
3. Edit the target signature appearance
4. The signature appearance aspect ratio and size can be changed by following the below given instructions:
    - Move the mouse over the signature appearance lower left corner (or edge)
    - Click on a corner or edge and re-size pointers will be shown
    - Hold the mouse and drag it to increase or decrease the aspect ratio of the signature appearance
5. The hand-signature and logo images within the signature appearance can be re-sized by selecting the image left mouse click on the image and
    - Use Alt + Right to expand the image size to the right
    - Use Alt + Down to expand the image size to the bottom
    - Use Alt + Left - shrink the image size to the left
    - Use Alt + Up - shrink the image size towards the top

Why embedded OCSP revocation information in PDF documents may not be verifiable by Adobe Reader?

One good reason why the PDF signatures created by ADSS Server with OCSP based embedded revocation information may not be verifiable by Adobe Reader is that the OCSP Responder certificate may not be authorised to respond for the relevant target CA. Adobe Reader expects the OCSP Responder certificate to be issued by the same CA as the issuer of the target certificate. The OCSP Responder certificate should also contain the ocspSigning bit set in its Extended KeyUsage. This aspect is discussed in more detail in the Digital Signature guide by Adobe, see section 5.3.1.1 of **this guide**.

Why am I unable to use signing key that resides in Azure Key Vault?

If you are unable to use the signing key that resides in Azure Key Vault, then check the **keyvault.log** file for error details at location: **[ADSS Server Installation Directory]/logs/service/pkcs11/keyvault.log**

Configuring ADSS Server to sign large files (100MB+)

Large files are supported by the Signing and Go>Sign Service, however:

1) Tomcat must be configured to allow large files to be used - **click here** to see how to do that.

2) The Java Memory Management parameters must be set to be quite large on the services service to allow a large file to be processed. Expect to provide at least around 25x the size of the file as RAM and sightly less for the ADSS-Services service JVM. Ensure that proper Java heap size

is configured for both a Java based business application and also for ADSS Server (only when the whole file is sent to the ADSS Server). To change the heap sizes for the ADSS Server **Click Here**. The JVM size for the business application can be changed in a similar way. Normally the ADSS Services JVM memory should be ~4 GB for a 100 MB file. For a 200MB file allow 5-6GB, for 500MB allow 12GB, for 700MB allow 16GB, for 1GB allow 24GB. To process multiple files at once allow more memory.
If the business application is still failing after increasing the JVM size, check the errors/exceptions generated inside the Java business application and ADSS Server (for logs check: ***ADSS Server Installation/ logs/service/signing***) and if an OutofMemory exception is shown then increase the value for the -JvmMx parameter, providing more physical RAM as needed.

If your business application is using ADSS Client-SDK or Auto File Processor to hash the file locally then there is no need to increase the JVM size for ADSS Server. A Java based business application will still need a large JVM to handle such large files. It can take a few seconds for the signing process to complete.

For rapid application development, the ADSS Client-SDK can be used which comes with sample code to assist developers. Signing supports all of the formats mentioned here. Check these demos in the ADSS Client-SDK package:

- **SignPDF** - send whole PDF to ADSS Server
- **SignFile**- send whole File to ADSS Server
- **SignXML**- send whole XML to ADSS Server
- **SignPdfWithLocalHash** - send just the hash of the PDF to ADSS Server.
- **SignFileWithLocalHash*** - send just the hash of the File to ADSS Server. **Note: Only CAdES-A is not supported via the Signing Service**
- **SignXmlWithLocalHash*** - send just the hash of the XML to ADSS Server. **Note: Only XAdES-A is not supported via the Signing Service**

*These signatures can be later upgraded to PAdES Part 4 LTV, CAdES-A and XAdES-A respective by using the ADSS Verification Service.