ADSS Server v8.4.0 – Test Tool User Guide

ASCERTIA LTD

FEBRUARY 2026

Document Version- 1.0.0

CONTENTS

1 Introduction

1.1 Scope

This document provides information on how the ADSS Test Tool utility can be used to confirm the correct configuration and operation of an ADSS Server instance.

ADSS Server provides a comprehensive set of security services via its high level APIs and web-services interfaces. When the ADSS Server is first installed it is reassuring to be able to confirm the correct operation of these services.

The ADSS Test Tool has been designed for this purpose. It is a command line utility that tests the ADSS Server's Signing, Verification, XKMS, SCVP, Certification, OCSP, OCSP Repeater, TSA, LTANS and RA Services.

Ascertia additionally provides separate test tools for testing OCSP and TSA services. These tools include OCSP Client Tool, OCSP Crusher and TSA Crusher.

1.2 Intended Readership

This document is intended for ADSS Server administrators who need to test its correct configuration and operation. It is assumed that the reader has a basic knowledge of digital signatures, certificates 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 New` font identifies code and text that appears on the command line.
- **`Bold Courier New`** identifies commands that you are required to type in.

1.4 Technical Support

If Technical Support is required, Ascertia has a dedicated support team. Ascertia Support can be reached/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 described above, Ascertia provides formal support agreements with all product sales. Please contact sales@ascertia.com for more details.

When sending support queries to Ascertia Support team, also send ADSS Server logs. Use the Ascertia's trace log export utility to collect logs for last two days or from the date since you are facing the problem. It will help support team to diagnose the issue in detail.

2 System Requirements

The following table summarizes the minimum requirements to support ADSS Server Test Tool deployment:

Component	**Minimum Requirements**
Operating System	• Windows 7 - Windows 10 • 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 and 1GB 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.

3 Running the ADSS Server Test Tool

The ADSS Server Test Tool must be unzipped in a suitable directory. ADSS Server Test Tool is available in two versions:

- Java based - for all types of platforms e.g. UNIX, Windows etc.
- .Net based - for only Windows platform

3.1 Java Based

If you are using the Java based ADSS Server Test Tool, Test-Tool Package contains Windows and Linux JREs, browse to the ADSS Server Test Tool home directory and follow these instructions to run it:

- **On Windows:**
  i. Execute the file **adss-testtool.bat** to launch the ADSS Server Test Tool
- **On UNIX:**
  i. Browse to the ADSS Server Test Tool home directory e.g. **/export/home/ADSS-Server-Test-Tool** and execute the following command:

     ```
     # sh chmod +x adss-testtool.sh
     ```

  ii. Execute the file **adss-testtool.sh** to launch the ADSS Server Test Tool by using the flowing command:

     ```
     # sh adss-testtool.sh
     ```

3.2 .Net Based

If you are using the .Net based ADSS Server Test Tool, then simply browse to the ADSS Server Test Tool home directory e.g. **C:/ADSS-Server-Test-Tool/** and execute the file **adss-testtool.bat** to launch the ADSS Server Test Tool.

4 Testing ADSS Signing Service

ADSS Signing Service is used to apply digital signatures on electronic documents. The ADSS Server Test Tool has an in-built Help feature, use the following command to get help on ADSS Test Tool usage for the Signing Service:

```
-service signing -help
```

The usage information will be shown on the console. The detail of each attribute is shown in the below table

Many of the attributes mentioned below are optional as they can be pre-defined in the signing profile configured on the ADSS Server. However, these signing profile attributes can be over-ridden by including new values for these attributes through the request message.

If a request parameter value contains space character(s) then, enclose such values within double quotation marks. To avoid this, ensure spaces are not used, e.g. in the file paths.

Action	Notes
-service	Specifies the ADSS Server service name. e.g. signing
-server	Specifies the ADSS Server URL. e.g. http://localhost:8777/adss/signing/hdsi
-client	Specifies the client originator ID to identify this client application to the ADSS Server. e.g. samples_test_client **Note:** See the ADSS Server Admin Guide for further details on managing client applications within ADSS Server.
-type	Specifies the type of document to be signed. Possible values are: • PDF • XML • XADES (For XAdES Legacy Signatures) • XADES_E (For XAdES Extended Signatures) • XADES_B (For XAdES Baseline Signatures) • FILE (For other type of files) • HASH • MS_WORD • MS_EXCEL • STATUS • EPASS_LDS
-in	Specifies the path of the input file for signing. e.g. “c:/input.pdf”
-out {optional}	Specifies the path for storing the signed file. e.g. “c:/output.pdf”

-requestId {optional}	Specifies an ID to be associated with the signing transaction. e.g. REQ-001
-mode {optional}	Specifies the mode to be used by the client to send and receive the request/response from the ADSS Server. Possible values are: • HTTP (default) • DSS If this parameter is not provided in the request, then default mode (HTTP) will be used. **Note:** HTTP refers to the use of plain HTTP requests using Ascertia defined HTTP protocol. DSS refers to the standard OASIS DSS protocol.
-userId {optional}	Specify the user id to be used for the authorised remote signing request. **Note:** It is mandatory in case of authorised remote signing request.
-transactionId {optional}	Specifies an ID to be used for fetching the status of the signature for the authorised remote signing request. e.g. transaction-001 **Note:** It is mandatory in case of authorised remote signing status request
-documentID {optional}	Specify a unique id to be associated with document being signed during authorised remote signing. e.g. 56398547 **Note:** It is mandatory in case of authorised remote signing request.
-profile {optional}	Specifies a document signing profile ID or Name. e.g. adss:signing:profile:002 **Note:** If this is not specified then the default signing profile configured in Client Manager will be used. Of course the profile name used in the command must already exist within the ADSS Signing Service.
-alias {optional}	Specifies a document signing certificate alias. e.g. test-certificate **Note:** If this is not specified than the default certificate identified in the ADSS Signing Profile will be used. The certificate alias name must exist within the ADSS Server and its purpose must be document signing. It must also be authorised for use by this application within the ADSS Server Client Manager in case generated via the ADSS Key Manager or previously have been requested by this client application in case generated via the ADSS Certification Service.
-password {optional}	Specifies the password for the Signing certificate. e.g. password123 **Note:** This is mandatory field only in cases where the certificate being used was generated automatically through the ADSS Certificate Service with user password. In case of keys manually generated using the ADSS Key Manager then there is no need to supply the password as it is automatically managed by the ADSS Server
-samlAssertion {optional}	Specifies the SAML2 assertion to authenticate and authorize the signer on RAS/SAM Service.

-localHash {optional}	Use this flag if the client needs to send only the hash of the document for signing. The hash will be computed locally by the client and sent to the ADSS Signing Service. The signed hash value will then be embedded back into the original document by the client e.g. -localHash
-localHashAlgo {optional}	Specifies hashing algorithm to be used for computing hash of the document locally. Possible values are: • SHA1 • SHA256 • SHA384 • SHA512 • SHA3-224 • SHA3-256 • SHA3-384 • SHA3-512
-sigPadding {optional}	Specify the signature padding scheme to be used for local signing. e.g. PKCS1/PSS
-docHashEnabled {optional}	Use this flag to send only document hash to signing service instead of full document. If this flag is used the Client-API will computes the hash **-in** file with given algorithm in **-localHashAlgo**. **Note:** - localHashAlgo is mandatory if this flag is set. Currently used for CAdES Detached Signatures
-base64DocumentHash {optional}	Specifies the base64 format for the unsigned content which is already hashed in case of CMS/CAdES detached signature signing or when document hash is required to send to signing service in case user does not wants to send full document to signing server. e.g. pd348U6+qXh7y1CkWzek/zEIyjgAfSQhUi/N/OYfLtg= **Note:** - localHashAlgo is mandatory if this flag is set. Currently used for CAdES Detached Signatures.
-lockDocument {optional}	Specify this attribute to lock the entire PDF document while doing local hash signing. e.g. -lockDocument
-lockFields {optional}	Specify this attribute to lock the specific form fields while doing local hash signing. e.g. -lockFields FormField1, FormField2
-hashMode {optional}	Specify the signature mode to be used for local hash signing. Possible values are: • Enveloping • Enveloped • Detached
-signHash {optional}	Specify this attribute to compute hash at signing time

	e.g. -signHash
-pdfSigType {optional}	Specify PAdES Signature type to be used for local hash signing. Possible values are: Extended Signatures: • PAdES-BES • PAdES-T • PAdES-LTV Baseline Signatures: • PAdES-B-B • PAdES-B-T • PAdES-LT • PAdES-B-LTA **Note:** Following parameters must be used with this option: • -localHash • -localHshAlgo • -hashMode (Its value must be detached) • -sigSize (Minimum value should be 40960)
-verificationAddress {optional}	Specify the ADSS Verification Service address to be used for generating PAdES-LTV, PAdES-LT and PAdES-B-LTA signatures using local hash. e.g. http://localhost:8777/adss/verification/hsvi **Note:** Not required if the ADSS Signing and Verification Services are running on the same URL, ADSS Server Test Tool will construct the Verification Service URL automatically. Only HTTP interface is supported.
-verificationProfile {optional}	Specify the ADSS Verification Profile Name/ID to be used for generating PAdES-LTV, PAdES-LT and PAdES-B-LTA signatures using local hash. e.g. adss:verification:profile:001 **Note:** If this is not specified then the default verification profile configured in Client Manager will be used. Of course the profile name used in the command must already exist within the ADSS Verification Service.
-timeStampAddress {optional}	Specify the ADSS TSA Service address to be used for generating PAdES-LTV and PAdES-B-LTA signatures using local hash. e.g. http://localhost:8777/adss/tsa **Note:** Not required if the ADSS Signing and TSA Services are running on the same URL, ADSS Server Test Tool will construct the TSA Service URL automatically.
-policyId {optional}	Specify the ADSS TSA Policy ID to be used for generating PAdES-LT, PAdES-LTV and PAdES-B-LTA signatures using local hash e.g. 1.2.3.4.5 **Note:** If this is not specified then the default TSA Policy configured in TSA Service will be used. Of course the Policy ID used in the command must already exist within the ADSS TSA Service.

-sigSize {optional}	Specify the signature dictionary size in bytes to be used for generating PAdES signatures using local hash. e.g. 40960
-sigPage {optional}	Specify the single/multiple comma separated page number of the PDF document for visible signature placement. e.g. 4,7 **Note:** The page number provided here should not be greater than the maximum number of pages in the PDF document.
-sigField {optional}	Specify the single already existing empty signature field name in the PDF document needs to be signed or multiple comma separated new signature field names need to be created and signed e.g. Signature1 **Note:** If this is specified in the request then the signature field name configured in the ADSS Signing Profile will be overridden by it. If the signing profile mentions that the PDF document will be signed on a pre-defined location (e.g. Top_Left) or at a precise signing location specified using PDF renderer then this attribute will be ignored.
-sigArea {optional}	Specify the Signing area in the PDF document where the signature needs to be placed. Possible values are: • top_left • top_right • center • bottom_left • bottom_right
-fieldsPos {optional}	Specify the single/multiple comma separated signature field position (X/Y coordinates) where the signature needs to be placed e.g. x1=308&y1=695&x2=508&y2=795 **Note:** • If this is specified in the request then the signature appearance location section configured inside the ADSS Signing Profile will be ignored. • For iText version 7 or later, the X2 coordinate is interpreted as the width value and Y2 coordinate is interpreted as the height value.
-certify {optional}	Specify the certified signing permission to be used for local hash signing. Possible values are: • noChanges • formFilling • formFilling&annotations
-reason {optional}	Specify the reason for signing to appear in the signature appearance. e.g. I am author of this document.
-location {optional}	Specify the location string to appear in the signature appearance. e.g. Egham, England

-contactInfo {optional}	Specify the contact info string to appear in the signature appearance. e.g. sales@ascertia.com
-companyLogo {optional}	Specify the path of the company logo image file on the client machine to appear in the signature appearance. e.g. C:/Images/CompanyLogo.jpg
-handSig {optional}	Specify the path of the hand signature image file on the client machine to appear in the signature appearance. e.g. C:/Images/HandSignature.jpg
-appearance {optional}	Specify the name of the PDF signature appearance to be used for signing. e.g. default_sig_appearance **Note:** Specify this parameter to override the signature appearance configured inside the ADSS Signing Profile. Of course the appearance profile name used in the command must already exist within the ADSS Server.
-sigApp {optional}	Specify the path of the signature appearance file on the client machine in case of local hash signing. e.g. C:\default_sig_appearance.app **Note:** If user does not provide this attribute in the hash signing request then an invisible signature will be produced on the PDF document.
-sigBy {optional}	Specify the signer name to appear in the "Signed By" attribute of the signature appearance upon hash signing. e.g. "John Doe"
-sigCert {optional}	Specify the path for the signing certificate to: • Get common name to be used for "Signed By" attribute in signature appearance upon hash signing. • Set the signer certificate as signed attribute upon local hashing. • When routing request to RAS/SAM Gateway as user key is not available at the relevant gateway to generate the signature structure. e.g. C:\sampleCertificate.cer
-sigCertChain {optional}	Specifies the path of PKCS7/p7b (certification chain) of the signing certificate. It is required to send signing certificate chain to the server for verification purpose.
-fontRep {optional}	Specify the path of the font directory to be used while generating visible signatures inside PDF document in case of local hash signing. e.g. C:\WINDOWS\Fonts **Note:** This is required if the fonts need to be embedded in the PDF signature to retain PDF/A compliance for the PDF documents.
-signingTime {optional}	Use this flag to add the signing time in the CAdES/XAdES signature. e.g. -signingTime
-signerRole {optional}	Specify the signer role to be added in the signed properties of the signature. e.g. "Admin"

-city {optional}	Specify the city name to be added in the signed properties of the signature. e.g. "Egham"
-postalCode {optional}	Specify the postal code to be added in the signed properties of the signature. e.g. "1784"
-stateOrProvince {optional}	Specify the name of the state or province to be added in the signed properties of the signature. e.g. "surrey"
-countryName {optional}	Specify the name of the country to be added in the signed properties of the signature. e.g. "England"
-docFormat {optional}	Specify the format of the document to be added in the signed properties of the signature in case of CAdES, XAdES and MS Office Signatures only. e.g. 1.2.3.4.5 **Note:** In case of CAdES signatures the document format must be an OID e.g. "1.3.5.4.7" and in case of XAdES it could be any string value e.g. "image file" as per CAdES/XAdES specifications.
-mimeTypeAttribute {optional}	Specify the Mime type to be added in the signature signed attribute for XAdES / CAdES signatures. This parameter is mandatory when `-docFormat` is specified e.g. text/xml
-mimetype {optional}	Specifies Input document mime type (file extension) e.g. pdf for test_input_unsigned.pdf. Applicable for `-type pdf` only. A mandatory parameter when document conversion is required.
-contentTimeStamp {optional}	Use this flag if the client needs to timestamp the content hash before generating the signature. e.g. - contentTimeStamp
-commitmentTypeIdentifier {optional}	Specify the commitment type indicator to be added in the signed properties of the CAdES and XAdES signatures. Possible values are: • ProofOfOrigin • ProofOfReceipt • ProofOfDelivery • ProofOfSender • ProofOfApproval • ProofOfCreation
-commitmentTypeQualifier {optional}	Specify the commitment type qualifier to include additional qualifying information on the commitment made by the signer. **Note:** `-commitmentTypeIdentifier` value is used if no value is given for this parameter.
-sigPolicyOID {optional}	Specify the Signature policy OID to be added in the signed properties of the AdES signature.

	e.g. "1.2.3.4.5"
-sigPolicyURI {optional}	Specify the Signature policy document URI to be extracted and added in the signed properties of the AdES signature. e.g. "http://www.ascertia.com/policy/policy.doc"
-sigPolicy {optional}	Specify the path to the Signature policy document. e.g. "c:/policy/policy.doc"
-sigPolicyUserNotice {optional}	Specify the Signature policy user notice to be added in the signed properties of the AdES signature. e.g. "Use this policy for test purpose only"
-counterSignature {optional}	Use this flag if the client needs to create a counter signature upon an already signed document. e.g. -counterSignature
-signedAuth {optional}	Specifies the single/multiple comma separated file paths for the signed authorisation files. e.g. C:/input_folder/AuthorisationFile.xml **Note:** This attribute becomes Mandatory if the client is using a signing profile that has an associated authorisation profile. An M of N scheme is used for signature based authorisation such that if M out of total N number of configured authorisers provides their approval then a high-trust signing key can be used to sign important documents.
-sigForm {optional}	Specify the AdES signature type in which signed document get converted. Signature type defined in the ADSS Signing Profile get overridden by this attribute value. Possible values are:

For Office and XAdES (ETSI TS 101 903) Signatures	For Extended Signatures (PAdES /CAdES/XAdES)	For Baseline Signatures (PAdES/ CAdES/XAdES)
• T • C • X • X-Type1 • X-Type2 • XL • XL-Type1 • XL-Type2 • A • A-Type1 • A-Type2	• E-T • E-C • E-X • E-X-Type1 • E-X-Type2 • E-XL • E-XL-Type1 • E-XL-Type2 • E-A • E-A-Type1 • E-A-Type2	• B-T • B-LT • B-LTA

-pdfProtection {optional}	Specify the single/multiple comma separated attributes to set the document level permissions on PDF document in case of local hashing. Possible values are: • allowPrintingLow/allowPrintingHigh • allowModification • allowCopying • allowDocAssembly • allowTextAccess • allowFormFilling • allowCommenting **Note:** Currently these permission settings are not supported for PAdES Part 4 signatures
-permissionsPassphrase {optional}	Specify the permission passphrase that will be set on the PDF document to change the permissions of this document in case of local hashing. e.g. password12
-reqTimeOut {optional}	Specify the time period in seconds that test tool should wait for a response from the ADSS Signing Service before closing the connection with it. e.g. 30
-soapVer {optional}	Specify the soap version to be used for composing the XML based request. Possible values are: • 1.1 • 1.2
-sigMode {optional}	Specify the signature type to be used for signing the XML based request. Possible values are: • Enveloped • Enveloping
-signingElementName {optional}	Specify the XML element to be signed based on Element Name or XPath expression. e.g. 'ContractName' or 'ContractName,ContractDate' or '//ContractName' or '//ContractName, //ContractDate'
-sigLocation {optional}	Specify the XML signature placement location in XML signing document. e.g. 'ContractName' or '//ContractName'. The generated signature will be placed within the specified element.
-addSigProperty {optional}	Add a signature property for additional information attached with signature.
-sigPropertyValue {optional}	Specify the signature property value as an additional information attached with signature e.g. 4654646465. It's time in milliseconds.
-clientAuthPfx {optional}	Specify the path of the Client Authentication PFX e.g. C:/Certs/ClientAuthentication.pfx **Note:** This client authentication certificate must also be configured in ADSS Server Client Manager. Issuer CA of this client Authentication certificate

	must be registered inside Trust Manager with purpose CA for Verifying TLS Client certificates (After registering the Issuer CA, ADSS Server Core, Console and Service instances must be restarted from Windows Service panel or Unix Daemon). For detailed instructions on setting up TLS Client Authentication with ADSS Services, refer to the technical note titled ***'How to set up TLS Client Authentication communication with ADSS Services'*** in the Ascertia Community Portal knowledge base.
-clientAuthPfxPass {optional}	Specify the password for the Client Authentication PFX. e.g. password12
-reqSignPFX {optional}	Specify the path of the Request Signing PFX. e.g. C:/Certs/RequestSigning.pfx **Note:** This request signing certificate must also be configured in ADSS Server Client Manager.
-reqSignPfxPass {optional}	Specify the password for the Request Signing PFX. e.g. password12
-proxyHost {optional}	Specify the IP address or machine name of the proxy host. e.g. ProxyHostName or 192.168.102.12
-proxyPort {optional}	Specify the port no. for communication with proxy host. e.g. 8080
-proxyUser {optional}	Specify the user name for proxy authentication. e.g. user12
-proxyPass {optional}	Specify the user password for proxy authentication. e.g. password12
-proxyDigest {optional}	Specify this attribute if proxy digest authentication is required. e.g. -proxyDigest
-batchCount {optional}	Specify the number of batches to be executed for load testing. e.g. 3
-requestCount {optional}	Specify the number of concurrent requests to be executed within a batch for load testing. e.g. 100
-xmlObjectId {optional}	Specify a unique id to be associated with xml content Object Identifier and reference URI for the enveloping signatures. e.g. book-1
-verbose {optional}	Specify this attribute to display request/response processing details on console and also store them on disk. e.g. -verbose **Note:** Request and response files will be stored at the location: [ADSS Server Test Tool Home]/verbose/

4.1 PDF Signing Request with Server Hashing

For use when you wish to create PDF signatures by sending PDF document to ADSS Server for signing

Using Minimum Attributes

```
-service signing -server http://localhost:8777/adss/signing/hdsi -type pdf -
in data/signing/input/test_input_unsigned.pdf -out
data/signing/output/test_input_signed.pdf -client samples_test_client
```

In the above example, the ADSS Test Tool utility sends a PDF signing request to the ADSS Server over HTTP interface. The input PDF file is signed using the default signing profile defined within ADSS Client Manager for this application, identified by the client originator ID “samples_test_client”. The default signing certificate associated with the signing profile is used.

Using Optional Attributes

```
-service signing -server http://localhost:8777/adss/signing/dss -mode dss -
type pdf -in data/signing/input/test_input_unsigned.pdf -out
data/signing/output/test_input_signed.pdf -client samples_test_client -
profile adss:signing:profile:001 -sigArea top_left -sigPage 1 -reason "I am
author of this document" -location "Egham, England" -contactInfo
"info@ascertia.com" -companyLogo data/signing/input/Digital-stamp.jpg -
handSig data/signing/input/andy-signature.jpg -fontRep C:\WINDOWS\Fonts -
city Egham -countryName England -stateOrProvince Surrey -postalCode 1784 -
signerRole Admin -alias samples_user_certificate -password password -verbose
```

In the above example, the ADSS Test Tool utility sends a PDF signing request to the ADSS Server over DSS interface having optional attributes. The input PDF file is signed using the signing profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application. The signing certificate associated with the signing profile is overridden by the certificate alias (Issued from ADSS Certification Service) provided in the request.

*The selected signature type inside Signing Profile should be **PDF (and PAdES profiles)***

4.2 PAdES Part 2 LTV Signing Request with Client 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 Test Tool to generate PAdES Part 2 LTV signatures.

```
-service signing -server http://localhost:8777/adss/signing/hdsi -mode http
-type PDF -in data/signing/input/test_input_unsigned.pdf -out
data/signing/output/PAdES_Part2_LTV.pdf -localHash -localHashAlgo SHA256 -
hashMode DETACHED -sigSize 40960 -profile adss:signing:profile:005 -sigArea
top_left -sigPage 1 -reason I_am_author_of_this_document -location
Egham_England -contactInfo support@ascertia.com -sigBy Andy -sigApp
data/signing/input/appearance.xml -companyLogo data/signing/input/Digital-
stamp.jpg -handSig data/signing/input/andy-signature.jpg -client
samples_test_client -verbose
```

In the above example, the ADSS Test Tool utility sends a PDF signing request with Client side hashing to the ADSS Server over HTTP interface. The input file is signed using the signing profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application.

The selected signature type inside Signing Profile should be ***PDF/PAdES Hash*** *and in Signature settings tab* ***Standard PDF Signature (PDF Signatures based on ISO 32000-1)*** *should be selected.*

The iText files for both Java and .NET versions are already available in the bin folder, while the license file is placed in the root directory.

4.3 PAdES Part 4 LTV Signing Request with Client 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 Test Tool to generate PAdES Part 4 LTV signatures

```
-service signing -server http://localhost:8777/adss/signing/hdsi -mode http
-type PDF -in data/signing/input/test_input_unsigned.PDF -out
data/signing/output/PAdES_Part4_LTV.PDF -localHash -localHashAlgo SHA256 -
hashMode DETACHED -pdfSigType PAdES-LTV -sigSize 40960 -profile
adss:signing:profile:005 -sigArea top_left -sigPage 1 -reason
I_am_author_of_this_document -location Egham_England -contactInfo
support@ascertia.com -sigBy Andy -sigApp data/signing/input/appearance.xml -
companyLogo data/signing/input/Digital-stamp.jpg -handSig
data/signing/input/andy-signature.jpg -client samples_test_client -verbose -
verificationAddress http://localhost:8777/adss/verification/dss -
verificationProfile adss:verification:profile:001 -timeStampAddress
http://localhost:8777/adss/tsa -policyId 1.2.3.4.5
```

In the above example, the ADSS Test Tool utility sends a PDF signing request with Client side hashing to the ADSS Server over HTTP interface. The input file is signed using the signing profile defined within the request, identified by the client originator ID "samples_test_client" defined within ADSS Client Manager for this application.

For creating PAdES Part4 LTV signatures using client hashing, you need the license for Signing and verification Service (TSA Service optional in case of external TSA).

The selected signature type inside Signing Profile should be ***PDF/PAdES Hash*** *and under Signature settings section, select the signature type* ***PAdES-E-BES with embedded timestamp (PAdES Extended Signatures based on ETSI standards).***

The iText files for both Java and .NET versions are already available in the bin folder, while the license file is placed in the root directory.

4.4 PAdES-B-LTA Signing Request with Client 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 Test Tool to generate PAdES-B-LTA signatures

```
-service signing -server http://localhost:8777/adss/signing/hdsi -mode http
-type PDF -in data/signing/input/test_input_unsigned.pdf -out
data/signing/output/PAdES_B_LTA.pdf -localHash -localHashAlgo SHA256 -
hashMode DETACHED -pdfSigType PAdES-B-LTA -sigSize 40960 -profile
adss:signing:profile:005 -sigArea top_left -sigPage 1 -reason
I_am_author_of_this_document -location Egham_England -contactInfo
support@ascertia.com -sigBy Andy -sigApp data/signing/input/appearance.xml -
```

```
companyLogo data/signing/input/Digital-stamp.jpg -handSig
data/signing/input/andy-signature.jpg -client samples_test_client -verbose -
verificationAddress http://localhost:8777/adss/verification/dss -
verificationProfile adss:verification:profile:001 -timeStampAddress
http://localhost:8777/adss/tsa -policyId 1.2.3.4.5
```

In the above example, the ADSS Test Tool utility sends a PDF signing request with Client side hashing to the ADSS Server over HTTP interface. The input file is signed using the signing profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application.

For creating PAdES-B-LTA signatures using client hashing, you need the license for Signing and verification Service (TSA Service optional in case of external TSA).

*The selected signature type inside Signing Profile should be **PDF/PAdES Hash** and under Signature settings section, select the signature type **PAdES-B-T (PAdES Baseline Signatures based on ETSI standards).***

The iText files for both Java and .NET versions are already available in the bin folder, while the license file is placed in the root directory.

4.5 File signing Request with Server hashing

For use when you wish to create File signatures by sending document to ADSS Server for signing

Using Minimum Attributes

```
-service signing -server http://localhost:8777/adss/signing/hdsi -mode http
-type FILE -in data/signing/input/test_input_unsigned.txt -out
data/signing/output/test_input_signed.pkcs7 -client samples_test_client -
profile adss:signing:profile:004 -verbose
```

In the above example, the ADSS Test Tool utility sends a File signing request to the ADSS Server over HTTP interface. The input file is signed using the signing profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application.

Using Optional Attributes

```
-service signing -server http://localhost:8777/adss/signing/hdsi -mode http
-type FILE -in data/signing/input/test_input_unsigned.txt -out
data/signing/output/test_input_signed.pkcs7 -client samples_test_client -
profile adss:signing:profile:004 -signingTime -signerRole Admin -city Egham
-postalCode 1784 -stateOrProvince Surrey -countryName England -docFormat
1.3.7.1.2 -contentTimeStamp -commitmentTypeIdentifier ProofOfSender -verbose
```

In the above example, the ADSS Test Tool utility sends a File signing request to the ADSS Server over HTTP interface. The input file is signed using the signing profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application. Optional signed attributes also sent in request to be added in the signature.

*The selected signature type inside Signing Profile should be either **CMS (and CAdES profiles)** or **PKCS#7***

4.6 File Signing Request with Client Hashing

For use when you wish the Non-PDF document to be hashed locally and the hash alone sent to ADSS Server for signing.

```
-service signing -server http://localhost:8777/adss/signing/dss -mode dss -
type cades -in data/signing/input/test_input_unsigned.txt -out
data/signing/output/test_input_signed.pkcs7 -client samples_test_client -
profile adss:signing:profile:007 -signingTime -signerRole Admin -city Egham
-postalCode 1784 -stateOrProvince Surrey -countryName England -docFormat
1.3.7.1.2 -contentTimeStamp -commitmentTypeIdentifier ProofOfSender -
localHash -localHashAlgo SHA256 -hashMode Detached -sigCert
data/signing/input/samples_test_signing_certificate.cer -verbose
```

In the above example, the ADSS Test Tool utility sends a File hash signing request to the ADSS Server over DSS interface. The input file is signed using the signing profile defined within the request, identified by the client originator ID “samples_test_client” , identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application. Optional signed attributes also sent in request to be added in the signature.

*The selected signature type inside Signing Profile should be **PKCS#1**. Navigate to Advance Settings of the Signing Profile and if -signHash is present in the request then mark the checkbox unchecked and vice versa*

4.7 XML Signing Request with Server Hashing

For use when you wish to create XML signatures by sending XML document to ADSS Server for signing

Using Minimum Attributes

```
-service signing -server http://localhost:8777/adss/signing/hdsi -mode http
-type XML -in data/signing/input/test_input_unsigned.xml -out
data/signing/output/test_input_signed.xml -client samples_test_client -
profile adss:signing:profile:003 -verbose
```

In the above example, the ADSS Test Tool utility sends a XML signing request to the ADSS Server over HTTP interface. The input file is signed using the signing profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application.

Using Optional Attributes

```
-service signing -server http://localhost:8777/adss/signing/dss -mode dss -
type XML -in data/signing/input/test_input_unsigned.xml -out
data/signing/output/test_input_signed.xml -client samples_test_client -
profile adss:signing:profile:003 -signingTime -signerRole Admin -city Egham
-postalCode 1784 -stateOrProvince Surrey -countryName England -docFormat XML
-contentTimeStamp -commitmentTypeIdentifier ProofOfSender -verbose
```

In the above example, the ADSS Test Tool utility sends a XML signing request to the ADSS Server over HTTP interface. The input file is signed using the signing profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application. Optional signed attributes also sent in request to be added in the signature.

*The selected signature type inside Signing Profile should be **XML Dsig (and XAdES profiles**)*

4.8 XML Signing Request with Client Hashing

For use when you wish the XML document to be hashed locally and the hash alone sent to ADSS Server for signing.

```
-service signing -server http://localhost:8777/adss/signing/hdsi -mode http
-type xades -in data/signing/input/test_input_unsigned.xml -out
data/signing/output/test_input_signed.xml -client samples_test_client -
profile adss:signing:profile:007 -signingTime -signerRole Admin -city Egham
-postalCode 1784 -stateOrProvince Surrey -countryName England -docFormat XML
-contentTimeStamp -commitmentTypeIdentifier ProofOfSender -localHash -
localHashAlgo SHA256 -hashMode Enveloped -signHash -sigCert
data/signing/input/samples_test_signing_certificate.cer -verbose
```

In the above example, the ADSS Test Tool utility sends a XML hash signing request to the ADSS Server over DSS interface. The input file is signed using the signing profile defined within the request, identified by the client originator ID "samples_test_client" defined within ADSS Client Manager for this application. Optional signed attributes also sent in request to be added in the signature.

*The selected signature type inside Signing Profile should be **PKCS#1**. Navigate to Advance Settings in Signing Profile, if **-signHash** is present in the request, then, mark the **'Compute hash at signing time'** checkbox uncheck and vice versa.*

4.9 MS Office Signing Request with Server Hashing

For use when you wish to create MS Office native signatures by sending MS Office Word/Excel document to ADSS Server for signing

MS Word

```
-service signing -server http://localhost:8777/adss/signing/dss -mode dss -
type ms_word -in data/signing/input/test_input_unsigned.docx -out
data/signing/output/test_input_signed.docx -client samples_test_client -
profile "Sample Profile to Sign Office Document" -sigField info@ascertia.com
-handSig data/signing/input/andy-signature.jpg -verbose
```

MS Excel

```
-service signing -server http://localhost:8777/adss/signing/hdsi -mode http
-type ms_excel -in data/signing/input/test_input_unsigned.xlsx -out
data/signing/output/test_input_signed.xlsx -client samples_test_client -
profile "Sample Profile to Sign Office Document" -sigField info@ascertia.com
-handSig data/signing/input/andy-signature.jpg -verbose
```

In the above example, the ADSS Test Tool utility sends a MS Office native signing request to the ADSS Server over DSS interface. The input file is signed using the signing profile defined within the request, identified by the client originator ID "samples_test_client" defined within ADSS Client Manager for this application.

*The selected signature type inside Signing Profile should be **MS Office***

4.10 MS Office Signing Request with Client Hashing

For use when you wish the MS Office Word/Excel 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 MS Office word/Excel document by ADSS Test Tool.

MS Word

```
-service signing -server http://localhost:8777/adss/signing/hdsi -mode http
-type ms_word -in data/signing/input/test_input_unsigned.docx -out
data/signing/output/test_input_signed.docx -client samples_test_client -
profile adss:signing:profile:007 -sigField info@ascertia.com -handSig
data/signing/input/andy-signature.jpg -localHash -localHashAlgo sha256 -
sigCert data/signing/input/samples_test_signing_certificate.cer -verbose
```

MS Excel

```
-service signing -server http://localhost:8777/adss/signing/dss -mode dss -
type ms_excel -in data/signing/input/test_input_unsigned.xlsx -out
data/signing/output/test_input_signed.xlsx -client samples_test_client -
profile adss:signing:profile:007 -sigField info@ascertia.com -handSig
data/signing/input/andy-signature.jpg -localHash -localHashAlgo sha256 -
sigCert data/signing/input/samples_test_signing_certificate.cer -verbose
```

In the above example, the ADSS Test Tool utility sends a MS Office hash signing request to the ADSS Server over HTTP interface. The Signature field configured in ADSS signing profile overridden by the signature field info@ascertia.com defined in the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application.

*The selected signature type inside Signing Profile should be **PKCS#1***

4.11 Authorized Signing Request for e-Sealing

For use when you wish to create e-Seal by sending document to ADSS Server for signing

For ACF Signing

```
-service signing -server http://localhost:8777/adss/signing/dss -mode dss -
client "samples_test_client" -type xml -in data\signing\input\ACF.xml -out
data\signing\output\ACF.xml -profile "adss:signing:profile:003 -verbose
```

In the above example, the ADSS Test Tool utility sends an ACF File signing request to the ADSS Server over DSS interface. The input file is signed using the signing profile defined within the request, identified by the client originator ID “samples_test_client”.

For Document Signing

```
-service signing -server http://localhost:8777/adss/signing/dss -mode dss -
client "samples_test_client" -type pdf -in data\signing\input\1.pdf -out
data\signing\output\1-signed.pdf -profile "adss:signing:profile:006" -
signedAuth data\signing\output\ACF.xml -verbose
```

In the above example, the ADSS Test Tool utility sends a document signing request along with ACF File for signing to the ADSS Server over DSS interface. The input file is signed using the signing profile defined within the request, identified by the client originator ID “samples_test_client”.

4.12 LDS Signing Request for E-Passport Chip

For use when you wish to sign E-Passport LDS (Logical Data Structure) by sending it to ADSS Server.

Using Minimum Attributes

```
-service signing -server http://localhost:8777/adss/signing/hdsi -mode http -
type EPASS_LDS -in data/signing/input/test_input_unsigned_lds_v1.pkcs7 -out
data/signing/output/test_input_signed_lds_v1.pkcs7 -client
samples_test_client -profile adss:signing:profile:009 -verbose
```

Using Optional Attributes

```
-service signing -server http://localhost:8777/adss/signing/hdsi -mode http -
type EPASS_LDS -in data/signing/input/test_input_unsigned_lds_v0.pkcs7 -out
data/signing/output/test_input_signed_lds_v0.pkcs7 -client
samples_test_client -profile adss:signing:profile:009 -
addDocSignerCertToSignature true -signingTime -verbose
```

4.13 Signing Request for Load Testing

User can optionally add batch count and request count to load test the signing service, one example is shown here:

```
-service signing -server http://localhost:8777/adss/signing/hdsi -type pdf -
in data/signing/input/test_input_unsigned.pdf -out
data/signing/output/test_input_signed.pdf -client samples_test_client -
batchCount 5 -requestCount 20
```

While doing load testing for a service try to avoid the use of verbose attribute in request because it will add a lot of I/O operations for dumping the request/response

4.14 Download Certificates Created via Key Manager

User can download the certificates created via Key Manager, one example is shown here:

```
-service signing -server http://localhost:8777/adss/signing/hcert -client
samples_test_client  -out -profile adss:signing:profile:001 -alias
samples_test_signing_certificate -mode http -verbose
```

5 Testing ADSS Verification Service

The ADSS Verification Service is able to verify digitally signed documents with a wide variety of document and signature formats and also the Signer certificates. The ADSS Server Test Tool has an in-built Help feature, use the following command to get help on ADSS Test Tool usage for the Verification Service:

```
-service verification -help
```

The usage information will be shown on the console. The detail of each attribute is shown in the below table:

If a request parameter value contains space character(s) then, enclose such values within double quotation marks. To avoid this, ensure spaces are not used, e.g. in the file paths.

Action	Notes
-service	Specifies the ADSS Server service name. e.g. verification
-server	Specifies the ADSS Server URL. e.g. http://localhost:8777/adss/verification/hsvi
-client	Specifies the client originator ID to identify this client application to the ADSS Server. e.g. samples_test_client **Note:** See the ADSS Server Admin Guide for further details on managing client applications within ADSS Server
-tranId	Specifies an ID to be associated with the verification transaction. e.g. transaction-001
-in	Specifies the path of the input file for verification i.e. signature/certificate to be validated. e.g. F:/in/sample_signed.pdf
-sigId	Specifies the signature ID for business application reference. e.g. signature001
-action	Specifies the verification operation to be performed. The possible values are: • SV (for signature verification) • CV (for certificate validation)
-type {optional}	Specifies the type of signatures to be verified. Possible values are: • SV (Signature Validation) requests: ○ pdf ○ xades ○ cades ○ cades_b ○ pkcs7 ○ cms ○ xml ○ ms_office

	• CV (Certificate Validation) request is X509
-content {optional}	Specifies the path for the unsigned content in case of detached signature verification. e.g. C:/input_folder/sample.pdf **Note:** This attribute is mandatory in case of detached signature verification.
-sendContentHash {optional}	Use this flag to send only content hash to verification service instead of full document. If this flag is used the Test-Tool will computes the hash **-content** file with given algorithm in -contentHashAlgorithm. **Note:** -contentHashAlgorithm is mandatory if this flag is set. Currently used for CAdES Detached Signatures.
-contentHashAlgorithm	Set the required content hash algorithm which is used to compute hash of doc given in -content flag e.g. SHA1 / SHA224 / SHA256 / SHA384 / SHA512 / SHA3-224 / SHA3-256 / SHA3-384 / SHA3-512 **Note:** Currently used for CAdES Detached Signatures.
-mode {optional}	Specifies the mode to be used by the client to send and receive the request/response from the ADSS Server. Possible values are: • HTTP (default) • DSS If this parameter is not provided in the request, then default mode (HTTP) will be used. **Note:** HTTP refers to the use of plain HTTP requests using Ascertia defined HTTP protocol. DSS refers to the standard OASIS DSS protocol.
-profile {optional}	Specifies a verification profile ID or Name. e.g. adss:verification:profile:001 **Note:** If this is not specified then the default verification profile configured in Client Manager will be used. Of course the profile name used in the command must already exist within the ADSS Verification Service.
-validationTime {optional}	Specify the time in GMT at which signatures/certificate needed to be verified. The date/time format is (yyyy/MM/dd hh:mm:ss) e.g. 2009/08/20 09:30:04
-PeppolCompliance {optional}	Specify this attribute if user needs to send a DSS request which is PEPPOL compliant. Possible values are: • True • False
-sigQuality {optional}	Specify the value for required signature quality level as per PEPPOL Trust ratings. e.g. 5
-certQuality {optional}	Specify the value for required certificate quality level as per PEPPOL Trust ratings. e.g. 5
-indAssurance {optional}	Specify the value for required independent assurance level as per PEPPOL Trust ratings.

	e.g. 3
-hashAlgQuality {optional}	Specify the value for required hash algorithm quality level as per PEPPOL Trust ratings. e.g. 4
-update {optional}	Specify this attribute to enhance the basic signatures into more advanced ETSI AdES formats. The possible values are: **For PAdES:** pades-ltv/pades-b-t/pades-b-lt/pades-b-lta **For CAdES Extended:** cades-t/cades-c/cades-x/cades-x-l/cades-a **For CAdES Baseline:** cades-b-t/cades-b-lt/cades-b-lta **For XAdES (ETSI TS 101 903) Signatures:** xades-t/xades-c/xades-x/xades-x-l/xades-a **For XAdES Extended Signatures:** xades-e-t/xades-e-c/xades-e-x/xades-e-x-l/xades-e-a **For XAdES Baseline Signatures:** xades-b-t/xades-b-lt / xades-b-lta
-out {optional}	Specifies the path for storing the response files e.g. “c:/output_folder”
-repondWith {optional}	Specify this attribute to receive verification information back from the ADSS Server. Possible comma separated values are: • key_value • hash_algo • signature_algo • x509_crl • ski • ocsp • timestamp • sign_hash • content_hash • content • key_usage • eku • basic_constraints • valid_from • valid_to • cert_serial_no • issuer_name • subject • crl_url • crl_no • long_term • cert_quality_level • sig_quality_level

	• x509_chain • timestamp_time • tsa_hash_algo • signature_type • tsa_name • lei_info **Note:** The flags "timestamp_time, tsa_name, tsa_hash_algo, signature_type, lei_info" only supported when -mode value HTTP is used.
-report {optional}	Specify this attribute when it is required to return an OASIS DSS-X Verification Report for the verification operation. Possible values are: • noDetails • noPathDetails • allDetails bullet
-include {optional}	Specify this attribute when it is required to return the additional extra elements along with OASIS DSS-X Verification Report. Possible comma separated values are: • verifier • certValues • revocation • binary
-returnVerificationTime {optional}	Specify this attribute when it is required to return the verification time in response message. The possible values are: • True • False **Note:** This flag is only supported when -mode value HTTP is used.
-reqTimeOut {optional}	Specify the time period in seconds that test tool should wait for a response from the ADSS Verification Service before closing the connection with it e.g. 30
-soapVer {optional}	Specify the soap version to be used for composing the XML based request. Possible values are: • 1.1 • 1.2
-sigMode {optional}	Specify the signature type to be used for signing the XML based request. Possible values are: • Enveloped • Enveloping
-clientAuthPfx {optional}	Specify the path of the Client Authentication PFX. e.g. C:/Certs/ClientAuthentication.pfx **Note:** This client authentication certificate must also be configured in ADSS Server Client Manager. Issuer CA of this client Authentication certificate must be registered inside Trust Manager with purpose CA for Verifying TLS Client certificates (After registering the Issuer CA, ADSS Server Core, Console and

	Service instances must be restarted from Windows Service panel or Unix Daemon). For detailed instructions on setting up TLS Client Authentication with ADSS Services, refer to the technical note titled ***'How to set up TLS Client Authentication communication with ADSS Services'*** in the Ascertia Community Portal knowledge base.
-clientAuthPfxPass {optional}	Specify the password for the Client Authentication PFX. e.g. password12
-reqSignPFX {optional}	Specify the path of the Request Signing PFX. e.g. C:/Certs/RequestSigning.pfx **Note:** This request signing certificate must also be configured in ADSS Server Client Manager.
-reqSignPfxPass {optional}	Specify the password for the Request Signing PFX. e.g. password12
-proxyHost {optional}	Specify the IP address or machine name of the proxy host. e.g. ProxyHostName or 192.168.102.12
-proxyPort {optional}	Specify the port no. for communication with proxy host. e.g. 8080
-proxyUser {optional}	Specify the user name for proxy authentication. e.g. user12
-proxyPass {optional}	Specify the user password for proxy authentication. e.g. password12
-proxyDigest {optional}	Specify this attribute if proxy digest authentication is required. e.g. -proxyDigest
-batchCount {optional}	Specify the number of batches to be executed for load testing. e.g. 3
-requestCount {optional}	Specify the number of concurrent requests to be executed within a batch for load testing. e.g. 100
-signatureTagPosition {optional}	Specify the position of signature tags in XAdES signatures to be verified in case of multiple signatures. e.g. 1,2
-signatureXpathToVerify {optional}	Specify the xpath of signature tags in XAdES signatures to be verified and in case of multiple signatures use “|” in between the signature xpaths. e.g. //ds:Signature[@Id='Signature_11299']| //ds:Signature[@Id='Signature_11549'] **Note**: This parameter will not work for XML DSig Signatures
-verbose {optional}	Specify this attribute to display request/response processing details on console and also store them on disk. e.g. -verbose

	Note: Request and response files will be stored at the location: [ADSS Server Test Tool Home]/verbose/

5.1 Certificate Validation Request

For use when you wish to validate certificate by sending target certificate to ADSS Server

```
-service verification -server http://localhost:8777/adss/verification/hsvi -
mode http -type pdf -in data/verification/input/test_input_signed.cer -
action cv -tranId Trans-011 -client samples_test_client -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate validation request to the ADSS Server over HTTP interface. The input target certificate is validated using the default verification profile defined within ADSS Client Manager for this application, identified by the client originator ID “samples_test_client”.

The signer certificate chain should be registered inside the Trust Manager and should also be added inside the Verification profile Trust anchor settings

5.2 PDF Signature Verification Request

For use when you wish to verify PDF signatures by sending PDF document to ADSS Server

PDF signatures should be enabled inside the Verification profile under signature settings

Using Minimum Attributes

```
-service verification -server http://localhost:8777/adss/verification/hsvi -
mode http -type pdf -in data/verification/input/test_input_signed.pdf -
action sv -tranId Trans-001 -client samples_test_client -verbose
```

In the above example, the ADSS Test Tool utility sends a PDF signature verification request to the ADSS Server over HTTP interface. The input signed PDF file is verified using the default verification profile defined within ADSS Client Manager for this application, identified by the client originator ID “samples_test_client”.

Using Optional Attributes

```
-service verification -server http://localhost:8777/adss/verification/dss -
mode dss -type pdf -profile adss:verification:profile:001 -in
data/verification/input/test_input_signed.pdf -action sv -tranId Trans-001 -
sigId Sig-001 -out data/verification/ResponseItems/ -respondWith
key_value,hash_algo,signature_algo,x509_crl,ski,ocsp,timestamp,sign_hash,con
tent_hash,content,key_usage,eku,basic_constraints,valid_from,valid_to,cert_s
erial_no,issuer_name,subject,crl_url,crl_no,long_term,cert_quality_level,sig
_quality_level,x509_chain,timestamp_time,tsa_hash_algo,signature_type,tsa_na
me,lei_info -sigQuality 2 -certQuality 3 -indAssurance 3 -hashAlgQuality 2 -
peppolCompliance true -report allDetails -include
verifier,certValues,revocation,binary -returnVerificationTime -client
samples_test_client -verbose
```

In the above example, the ADSS Test Tool utility sends a PDF signature verification request to the ADSS Server over DSS interface having optional attributes. The input signed PDF file is verified using the verification profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application.

The signer certificate chain should be registered inside the Trust Manager and should also be added inside the Verification profile Trust anchor settings

5.3 CMS Enveloping Signature Verification Request

For use when you wish to verify CMS enveloping signatures by sending it to ADSS Server

CMS signatures should be enabled inside the Verification profile under signature settings

Using Minimum Attributes

```
-service verification -server http://localhost:8777/adss/verification/hsvi -
mode http -type cades -in data/verification/input/test_input_signed.pkcs7 -
action sv -tranId Trans-001 -client samples_test_client -verbose
```

In the above example, the ADSS Test Tool utility sends a CAdES enveloping signature verification request to the ADSS Server over HTTP interface. The input signature file is verified using the default verification profile defined within ADSS Client Manager for this application, identified by the client originator ID "samples_test_client".

Using Optional Attributes

```
-service verification -server http://localhost:8777/adss/verification/dss -
mode dss -type cades -profile adss:verification:profile:001 -in
data/verification/input/test_input_signed.pkcs7 -action sv -tranId Trans-001
-sigId Sig-001 -out data/verification/ResponseItems/ -respondWith
key_value,hash_algo,x509_crl,ski,ocsp,timestamp,sign_hash,content_hash,conte
nt,key_usage,eku,basic_constraints,valid_from,valid_to,cert_serial_no,issuer
_name,subject,crl_url,crl_no,long_term,cert_quality_level,sig_quality_level,
x509_chain,timestamp_time,tsa_hash_algo,signature_type,tsa_name,lei_info -
sigQuality 2 -certQuality 3 -indAssurance 3 -hashAlgQuality 2 -
peppolCompliance true -report allDetails -include
verifier,certValues,revocation,binary -returnVerificationTime -client
samples_test_client -verbose
```

In the above example, the ADSS Test Tool utility sends a CAdES signature verification request to the ADSS Server over DSS interface. The input signed signature file is verified using the verification profile defined within the request, identified by the client originator ID "samples_test_client" defined within ADSS Client Manager for this application.

*For PCS#7 signatures use the type attribute value: **-type pkcs7***

The signer certificate chain should be registered inside the Trust Manager and should also be added inside the Verification profile Trust anchor settings

5.4 CMS Detached Signature Verification Request

For use when you wish to verify CMS detached signatures by sending it to ADSS Server

CMS signatures should be enabled inside the Verification profile under signature settings

Using Minimum Attributes

```
-service verification -server http://localhost:8777/adss/verification/hsvi -
mode http -type cades -in data/verification/input/test_input_signed.pkcs7 -
content data/signing/input/test_input_unsigned.txt -action sv -tranId Trans-
001 -client samples_test_client -verbose
```

In the above example, the ADSS Test Tool utility sends a CAdES detached signature along with original content in a verification request to the ADSS Server over HTTP interface. The input signature file is verified using the default verification profile defined within ADSS Client Manager for this application, identified by the client originator ID “samples_test_client”.

Using Optional Attributes

```
-service verification -server http://localhost:8777/adss/verification/dss -
mode dss -type cades -profile adss:verification:profile:001 -in
data/verification/input/test_input_signed.pkcs7 -content
data/signing/input/test_input_unsigned.txt -action sv -tranId Trans-001 -
sigId Sig-001 -out data/verification/ResponseItems/ -respondWith
key_value,hash_algo,x509_crl,ski,ocsp,timestamp,sign_hash,content_hash,conte
nt,key_usage,eku,basic_constraints,valid_from,valid_to,cert_serial_no,issuer
_name,subject,crl_url,crl_no,long_term,cert_quality_level,sig_quality_level,
x509_chain,timestamp_time,tsa_hash_algo,signature_type,tsa_name,lei_info -
sigQuality 2 -certQuality 3 -indAssurance 3 -hashAlgQuality 2 -
peppolCompliance true -report allDetails -include
verifier,certValues,revocation,binary -returnVerificationTime -client
samples_test_client -verbose
```

In the above example, the ADSS Test Tool utility sends a CAdES detached signature along with original content in a verification request to the ADSS Server over DSS interface. The input signed signature file is verified using the verification profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application.

For PCS#7 signatures use the type attribute value: ***-type pkcs7***

The signer certificate chain should be registered inside the Trust Manager and should also be added inside the Verification profile Trust anchor settings

5.5 XML Signature Verification Request

For use when you wish to verify XML signatures by sending it to ADSS Server

XML signatures should be enabled inside the Verification profile under signature settings

Using Minimum Attributes

```
-service verification -server http://localhost:8777/adss/verification/hsvi -
mode http -type xades -in data/verification/input/test_input_signed.xml -
action sv -tranId Trans-001 -client samples_test_client -verbose
```

In the above example, the ADSS Test Tool utility sends a XAdES signature verification request to the ADSS Server over HTTP interface. The input signature file is verified using the default verification profile defined within ADSS Client Manager for this application, identified by the client originator ID “samples_test_client”.

Using Optional Attributes

```
-service verification -server http://localhost:8777/adss/verification/dss -
mode dss -type xades -profile adss:verification:profile:001 -in
data/verification/input/test_input_signed.xml -action sv -tranId Trans-001 -
sigId Sig-001 -out data/verification/ResponseItems/ -respondWith
key_value,hash_algo,x509_crl,ski,ocsp,timestamp,sign_hash,content_hash,conte
nt,key_usage,eku,basic_constraints,valid_from,valid_to,cert_serial_no,issuer
_name,subject,crl_url,crl_no,long_term,cert_quality_level,sig_quality_level,
x509_chain,timestamp_time,tsa_hash_algo,signature_type,tsa_name,lei_info -
sigQuality 2 -certQuality 3 -indAssurance 3 -hashAlgQuality 2 -
peppolCompliance true -report allDetails -include
verifier,certValues,revocation,binary -returnVerificationTime -client
samples_test_client -signatureTagPosition 1 -signatureXpathToVerify
//ds:Signature[@Id='Signature_11299']  -verbose
```

In the above example, the ADSS Test Tool utility sends a XAdES signature verification request to the ADSS Server over DSS interface. The input signed signature file is verified using the verification profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application.

*For XML Dsig signatures use the type attribute value: **-type xml***

The signer certificate chain should be registered inside the Trust Manager and should also be added inside the Verification profile Trust anchor settings

5.6 MS Office Signature Verification Request

For use when you wish to verify MS Office signatures by sending it to ADSS Server

MS Office signatures should be enabled inside the Verification profile under signature settings

Using Minimum Attributes

```
-service verification -server http://localhost:8777/adss/verification/hsvi -
mode http -type ms_office -in data/verification/input/test_input_signed.docx
-action sv -tranId Trans-001 -client samples_test_client -verbose
```

In the above example, the ADSS Test Tool utility sends a MS Office signature verification request to the ADSS Server over HTTP interface. The input signature file is verified using the default verification profile defined within ADSS Client Manager for this application, identified by the client originator ID “samples_test_client”.

Using Optional Attributes

```
-service verification -server http://localhost:8777/adss/verification/dss -
mode dss -type ms_office -profile adss:verification:profile:001 -in
data/verification/input/test_input_signed.xlsx -action sv -tranId Trans-001
-sigId Sig-001 -out data/verification/ResponseItems/ -respondWith
key_value,hash_algo,x509_crl,ski,ocsp,timestamp,sign_hash,content_hash,conte
nt,key_usage,eku,basic_constraints,valid_from,valid_to,cert_serial_no,issuer
```

```
_name,subject,crl_url,crl_no,long_term,cert_quality_level,sig_quality_level,
x509_chain,timestamp_time,tsa_hash_algo,signature_type,tsa_name,lei_info -
sigQuality 2 -certQuality 3 -indAssurance 3 -hashAlgQuality 2 -
peppolCompliance true -report allDetails -include
verifier,certValues,revocation,binary -returnVerificationTime -client
samples_test_client -verbose
```

In the above example, the ADSS Test Tool utility sends a MS Office signature verification request to the ADSS Server over DSS interface. The input signed Ms office document file is verified using the verification profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application.

Sending only Signature for verification

For use when you wish to verify MS Office signatures by sending only signatures to ADSS Server instead of complete document. Need to add this additional flag in request -sendSignaturesOnly

```
-service verification -server http://localhost:8777/adss/verification/hsvi -
mode http -type ms_office -in data/verification/input/test_input_signed.docx
-action sv -tranId Trans-001 -client samples_test_client -verbose -
sendSignaturesOnly
```

In the above example, the ADSS Test Tool utility sends a MS Office signature instead of complete document in a verification request to the ADSS Server over HTTP interface. The input signature file is verified using the default verification profile defined within ADSS Client Manager for this application, identified by the client originator ID “samples_test_client”.

The signer certificate chain should be registered inside the Trust Manager and should also be added inside the Verification profile Trust anchor settings

5.7 Verification Request Using Document Hash

For use when only document hash is required to send to verification service instead of complete document.

```
-service verification -server http://localhost:8777/adss/verification/hsvi -mode http
-type cades -in data/verification/input/test_input_signed_cades_blta_detached.pkcs7 -
content data/signing/input/test_input_unsigned.txt -sendContentHash -
contentHashAlgorithm SHA256 -action sv -tranId cades-blta-dochash -client
samples_test_client -verbose
```

5.8 Verification Request for Signature Enhancement

For use when you wish to enhance an existing basic AdES signatures i.e. i.e. PAdES/ CAdES/ XAdES to advanced AdES signatures.

```
-service verification -server http://localhost:8777/adss/verification/dss -
mode dss -type pdf -in data/verification/input/test_input_signed.pdf -action
sv -tranId Trans-001 -client samples_test_client -verbose -update pades-ltv
-out data/verification/output/enhanced_test_signed.pdf
```

Signature enhancement settings should be enabled inside the verification Profile under advanced settings

The signer certificate chain should be registered inside the Trust Manager and should also be added inside the Verification profile Trust anchor settings

5.9 Verification Request for Load Testing

User can optionally add batch count and request count to load test the Verification service, one example is shown here:

```
-service verification -server http://localhost:8777/adss/verification/hsvi -
mode http -type pdf -in data/verification/input/test_input_signed.pdf -
action sv -tranId Trans-001 -client samples_test_client -batchCount 5 -
requestCount 20
```

While doing load testing for a service try to avoid the use of verbose attribute in request because it will add a lot of I/O operations for dumping the request/response

The signer certificate chain should be registered inside the Trust Manager and should also be added inside the Verification profile Trust anchor settings

6 Testing ADSS Certification Service

The ADSS Certification Service provides an XML/SOAP web-services CA that enables business applications to request key generation and/or certification. The ADSS Server Test Tool can make certification requests to the ADSS Certification Service, requests can also be made using a CMC or EST interface over HTTP/S.

The ADSS Server Test Tool has an in-built Help feature, use the following command to get help on ADSS Test Tool usage for the Verification Service:

```
-service certification -help
```

The usage information will be shown on the console. The detail of each attribute is shown in the below table:

If a request parameter value contains space character(s) then, enclose such values within double quotation marks. To avoid this, ensure spaces are not used, e.g. in the file paths.

Action	Notes
-service	Specifies the ADSS Server service name. e.g. certification
-server	Specifies the ADSS Server URL. e.g. http://localhost:8777/adss/certification/csi **Note:** For more details see ADSS Certification Service Interface URLs
-action	Specifies the certification operation to be performed. The possible values are: • create • renew • rekey • delete • revoke • changePassword • recover • import • authorize • unauthorized • profileInfo • sendSCT • caCerts • certificateInfo
-client	Specifies the client originator ID to identify this client application to the ADSS Server. e.g. samples_test_client **Note:** See the ADSS Server Admin Guide for further details on managing client applications within ADSS Server

-clientSecret	Specifies the client secret to authenticate this client application to the ADSS Server. e.g. 762b9a0c421fe4f64f47a6135545d436979450d6 **Note:** This parameter is used for **EST** interface only
-userId	Specifies the user friendly id of the user/device administrator. e.g. “John” **Note:** It is mandatory in case of register user request
-profile {optional}	Specifies a certification profile ID or Name. e.g. adss:certification:profile:001 **Note:** If this is not specified then the default certification profile configured in Client Manager will be used. Of course the profile name used in the command must already exist within the ADSS Certification Service.
-mode {optional}	Specifies the mode to be used by the client to send and receive the request/response from the ADSS Server. Possible values are: • XML (default) • CMC • EST **Note:** • If this parameter is not provided in the request, then default mode (XML) will be used. • For CMC request the PKCS10 should be of PEM format and request should be send over TLS Mutual Authentication, For detailed instructions on setting up TLS Client Authentication with ADSS Services, refer to the technical note titled ***'How to set up TLS Client Authentication communication with ADSS Services'*** in the Ascertia Community Portal knowledge base • For EST request the PKCS10 should be of PEM/Base64 format and request should be send with basic authentication in the absence of TLS client certificate and in case request is send over Mutual TLS Authentication then client credential parameters are optional, For detailed instructions on setting up TLS Client Authentication with ADSS Services, refer to the technical note titled ***'How to set up TLS Client Authentication communication with ADSS Services'*** in the Ascertia Community Portal knowledge base.
-pkiRequestMode {optional}	Specifies type of enrollment request to be used when request mode is CMC. The possible values are: • Simple (Default) • Full If you want to send the PKCS10 then use the simple option and if you want to send the PKCS7 then use the full option. **Note:** If this parameter is not provided in the request, then default mode (Simple) will be used.
-requestId {optional}	Specifies an ID to be associated with the certification transaction. e.g. transaction-001

-alias {optional}	Specifies an alias (unique identifier) for the certificate to be created or renewed. e.g. test-certificate
-password {optional}	Specifies the password for the certificate to be created. e.g. password123 **Note:** This is mandatory field if certificates are required to be generated on hardware crypto source i.e. USB tokens or HSM, otherwise if this attribute is not provided in the request and crypto source is software then ADSS Certification Service create and manage the password automatically. This attribute can be used with create, renew, changePassword and delete operations.
-newPassword {optional}	Specifies a new password for the already existing pkcs12 in case of changePassword or renew request. e.g. password321
-subject {optional}	Specifies the Subject DN for the certificate to be created or renewed e.g. "CN=Jhon Doe~encoding=UTF8String,OU=Development~encoding=PrintableString,O=Ascertia~encoding=UTF8String,C=GB~encoding=UniversalString" **Note:** This attribute can be used in case of renew request only when the certificate profile have the settings Renew certificate using new key pair.
-san {optional}	Specifies the Subject Alternative Name for the certificate to be created. Possible values are: rfc822Name==value1~value2&dNSName==value1~value2& iPAddress==value1~value2&uniformResourceIdentifier==value1~value2& otherName==OID=value,encoding=UTF8String~OID=value, encoding=OctetString~OID=value,encoding=PrintableString &directoryName==value1,encoding=UTF8String~value2,encoding=UTF8String uniformResourceIdentifier ==value1~value2 registeredID==value1~value2 ediPartyName==some value for nameAssigner, encoding:UTF8String=Some Value for partyName,encoding:UTF8String~value 1=test value 2,encoding=PrintableString Custom RDN can also be passed in the directoryName in the following format OID=value e.g. 1.2.3.4.5=customRDN e.g. "rfc822Name==jhon2@ascertia.com&ediPartyName==nameAssigner,encoding:UTF8String=partyName,encoding:UTF8String~nameAssigner1,encoding:UTF8String=partyName,encoding:UTF8String~nameAssigner1,encoding:UTF8String=partyName,encoding:UniversalString&registeredID==1.2.3.4~0.1.2&dNSName==www.ascertia.com&iPAddress==192.168.1.1~192.168.10.10&directoryName==CN=Jhon Doe,encoding=UTF8String,OU=Development=UniversalString,O=Ascertia,C=GB~CN=Jhon,encoding=UTF8String,OU=HR,O=Ascertia,C=GB,1.2.3.4.5=customRDN &uniformResourceIdentifier==https://services.example.com/endpoint"

-subjectDirectoryAttributes {optional}	Specifies the Subject Directory Attributes for certificate to be created. Possible values are:
	dateOfBirth=value1, placeOfBirth=value1
	e.g.
	dateOfBirth=19590927120000Z, placeOfBirth=PK

-keySize {optional}	Specifies the size of the key to be generated. e.g. 2048
-keyType {optional}	Specifies the algorithm to be used for generating key. Possible values are: • RSA • ECDSA • ML-DSA • ML-KEM
-curveType {optional}	Specifies the curve type to be used for ECDSA key generation. Possible values are: • NIST_P • SEC2_K • BRAINPOOL_R • BRAINPOOL_T
-validityPeriod {optional}	Specify the validity period for the certificate to be created or renewed. e.g. 12
-custom Extension	Specifies the Custom extension for certificate to be created. Possible values are: "oid=.3.6.1.4.1.59382.3.2~value=value1&oid=1.3.6.1.4.1.59382.3.3-value=value2”
-validityUnit {optional}	Specify the validity period unit of the certificate in following possible values: • MINS • HOURS • DAYS • MONTHS • YEARS
-validTo {optional}	Specify the validity date for the certificate to be created or renewed. The date format should be yyyy-MM-dd'T'HH:mm:ss e.g. 2020-02-10T15:53:23 **Note:** It is alternate of -validityPeriod/validityUnit, one need to use -validTo or -validityPeriod/-validityUnit for certificate expiry date. If both are used then it will create problems with the time calculation
-issuerDN {optional}	Specifies the Subject DN of the issuer CA certificate e.g. -issuerDN "OU=Development,O=Ascertia,C=GB,CN=ADSS Samples Test CA" **Note:** This attribute is used in case of revoke request only.

-cert {optional}	Specifies the path of the certificate file e.g. c:/certs/JohnDoe.pfx **Note:** This attribute is used for operations other than create.
-pkcs7 {optional}	Specifies the PKCS7/p7b (certification chain), if it is required to store it with the certificate on the server. e.g. C:/certs/JohnDoe.pkcs7
-pkcs10 {optional}	Specifies the PKCS10 (Certificate Request) which will be used to create the certificate. e.g. C:/certs/JohnDoe.p10
-publicKey {optional}	Specifies the PublicKey which will be used to create the certificate. e.g. C:/certs/JohnDoe.pub
-revReason {optional}	Specifies a revocation reason while revoking a certificate. The possible values are: • unspecified (Default) • keyCompromise • cACompromise • affiliationChanged • superseded • cessationOfOperation • certificateHold • removeFromCRL • privilegeWithdrawn • aACompromise **Note:** If this attribute is not added in the revoke request then certificate will be revoked with reason code unspecified.
-invalidityDate {optional}	Specifies the invalidity date to be used for certificate revoke request. The date format should be YYYY-MM-DD HH:MM:SS e.g. "2010-02-27 23:08:55"
-holdInstCode {optional}	Specifies the hold instruction code if the revocation reason is certificateHold. Possible values are • id-holdinstruction-none (Default) • id-holdinstruction-callissuer • id-holdinstruction-rejec **Note:** If revocation reason is other than certificateHold then this attribute will be ignored. If revocation reason is certificateHold and this attribute is not provided in the request then default (id-holdinstruction-none) value is used.
-out {optional}	Specifies the path for storing the response files. e.g. “c:/output_folder”
-sctVersion {optional}	Specifies the SCT version. **Note:** This attribute is used in case of sendSCT request only
-sctLogID	Specifies the base64 encoded Log ID.

{optional}	**Note:** This attribute is used in case of sendSCT request only.
-sctTimestamp {optional}	Specifies the timestamp. **Note:** This attribute is used in case of sendSCT request only.
-sctSignature {optional}	Specifies the base64 encoded SCT signature. **Note:** This attribute is used in case of sendSCT request only.
-sctServerAddress {optional}	Specifies the Certificate Transparency Log Server Address. **Note:** This attribute is used in case of sendSCT request only.
-authCertAlias {optional}	Specifies an authorization certificate alias (unique identifier) for the certificate to be associated with Qualified certificate e.g. Auth1-Certificate
-repondWith {optional}	Specify this attribute to receive information back from the ADSS Server. Possible comma separated values are: • certificate • pkcs7 • pkcs10 • pkcs12 • expiry_date,password
-reqTimeOut {optional}	Specify the time period in seconds that test tool should wait for a response from the ADSS Certification Service before closing the connection with it. e.g. 30
-soapVer {optional}	Specify the soap version to be used for composing the XML based request. Possible values are: • 1.1 • 1.2
-sigMode {optional}	Specify the signature type to be used for signing the XML based request. Possible values are: • Enveloped • Enveloping
-clientAuthPfx {optional}	Specify the path of the Client Authentication PFX. e.g. C:/Certs/ClientAuthentication.pfx **Note:** This client authentication certificate must also be configured in ADSS Server Client Manager. Issuer CA of this client Authentication certificate must be registered inside Trust Manager with purpose CA for Verifying TLS Client certificates (After registering the Issuer CA, ADSS Server Core, Console and Service instances must be restarted from Windows Service panel or Unix Daemon). For detailed instructions on setting up TLS Client Authentication with ADSS Services, refer to the technical note titled ***'How to set up TLS Client Authentication communication with ADSS Services'*** in the Ascertia Community Portal knowledge base.
-clientAuthPfxPass {optional}	Specify the password for the Client Authentication PFX. e.g. password12

-reqSignPFX {optional}	Specify the path of the Request Signing PFX. e.g. C:/Certs/RequestSigning.pfx **Note:** This request signing certificate must also be configured in ADSS Server Client Manager.
-reqSignPfxPass {optional}	Specify the password for the Request Signing PFX. e.g. password12
-proxyHost {optional}	Specify the IP address or machine name of the proxy host. e.g. ProxyHostName or 192.168.102.12
-proxyPort {optional}	Specify the port no. for communication with proxy host. e.g. 8080
-proxyUser {optional}	Specify the user name for proxy authentication. e.g. user12
-proxyPass {optional}	Specify the user password for proxy authentication. e.g. password12
-proxyDigest {optional}	Specify this attribute if proxy digest authentication is required. e.g. -proxyDigest
-batchCount {optional}	Specify the number of batches to be executed for load testing. e.g. 3
-requestCount {optional}	Specify the number of concurrent requests to be executed within a batch for load testing. e.g. 100
-serialNumber {optional}	Specify this serial Number to revoke, reinstate or retrieve a certificate e.g. -34fe43242343dwd434
-status {optional}	Specify the status of a certificate to be retrieved in get Certificates request, if no status is given then by default certificates with all the statuses will be returned. e.g. ACTIVE,SUSPENDED,REVOKED
-caAlias {optional}	Specify the CA alias of a certificate to be retrieved in get Certificates request, e.g. ADSS Samples Test CA
-startIndex {optional}	In the get Certificates request, specify the index of the first record to include in the results that will be retrieved., e.g. 0
-maxRecords {optional}	In the get Certificates request, specify the max number of records to include in the results that will be retrieved., e.g. 20
-securityLevel {optional}	Specifies security Level to be used for generating key in case of ML-DSA and ML-KEM algorithms.
-protectPdfPass {optional}	Set Password to parse PDF which protected by password.
-seatId {optional}	Seat ID refers to seat identification of an authorized end user of the service. It is an optional field only in case of Digicert ONE MPKI CA.

	e.g some-seat-id@example.com
-deleteParam {optional}	When the -deleteParam parameter is sent to the ADSS Certification Service, it instructs the system to delete the private key associated with the specified certificate stored on the ADSS Server. If the server does not contain the private key referenced by the parameter, an error message will be returned. **Note:** For security purposes, it is recommended to include the **PKCS12 key password** when submitting a delete request using the **-deleteParam** option. This password should match the one you specified during the certificate generation or issuance process.
-verbose {optional}	Specify this attribute to display request/response processing details on console and also store them on disk. e.g. -verbose **Note:** Request and response files will be stored at the location: [ADSS Server Test Tool Home]/verbose/

6.1 Certificate Creation Request

For use when you wish to create a certificate by sending request to ADSS Server

<u>Using Minimum Attributes</u>

Sending SubjectDN

```
-service certification -server http://localhost:8777/adss/certification/csi
-mode xml -out data/certification/output -action create -client
samples_test_client -subject "CN=Jhon Doe,OU=Development,O=Ascertia,C=GB" -
password password -alias Jhon -verbose
```

Sending PublicKey

```
-service certification -server http://localhost:8777/adss/certification/csi
-mode xml -out data/certification/output -action create -client
samples_test_client -subject "CN=Jhon Doe,OU=Development,O=Ascertia,C=GB" -
password password -alias Jhon -publicKey data/certification/input/sample.pub
-verbose
```

Sending PKCS#10

```
-service certification -server http://localhost:8777/adss/certification/csi
-mode xml -out data/certification/output -action create -client
samples_test_client -pkcs10 data/certification/input/sample.p10 -password
password -alias JhonP10 -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate creation request to the ADSS Server over XML interface. The input subjectDN/pkcs10 is used by the default certification profile defined within ADSS Client Manager for this application to generate the certificate, identified by the client originator ID “samples_test_client”.

Using Optional Attributes

```
-service certification -server http://localhost:8777/adss/certification/csi
-mode xml -out data\certification\output -action create -client
samples_test_client -subject CN=Jhon2 -password password -alias Jhon2 -
profile adss:certification:profile:001 -respondWith
certificate,pkcs7,pkcs12,expiry_date,password -keySize 2048 -keyType RSA -
san "rfc822Name==jhon2@ascertia.com&dNSName==www.ascertia.com" -
subjectDirectoryAttributes dateOfBirth=19590927120000Z,placeOfBirth=PK -
validityPeriod 36 -issuerDN "CN=ADSS Samples Test CA,OU=Ascertia Software
Distribution,O=Ascertia Limited,C=GB" -reqTimeOut 300 -securityLevel 5 -
verbose
```

In the above example, the ADSS Test Tool utility sends a certificate creation request to the ADSS Server having optional attributes. The input subjectDN is used to generate the certificate by the certification profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application.

6.2 Certificate Rekey Request

For use when you wish to rekey an existing certificate by sending request to ADSS Server

```
-service certification -server http://localhost:8777/adss/certification/csi -
action rekey -client samples_test_client -mode xml -alias Jhon -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate rekey request to the ADSS Server over XML interface. The input certificate alias is used by the certification profile defined within ADSS Client Manager for this application to rekey the certificate using new key pair, identified by the client originator ID “samples_test_client”.

6.3 Certificate Renew Request

For use when you wish to renew an existing certificate by sending request to ADSS Server

```
-service certification -server http://localhost:8777/adss/certification/csi -
action renew -client samples_test_client -mode xml -alias Jhon -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate renew request to the ADSS Server over XML interface. The input certificate alias is used by the certification profile defined within ADSS Client Manager for this application to renew the certificate using existing key pair, identified by the client originator ID “samples_test_client”.

Certificate can be renewed using existing key pair or new key pair. For more details see the certification profile

6.4 Certificate Recover Key Request

For use when you wish to Recover an existing key pair by sending request to ADSS Server

```
-service certification -server http://localhost:8777/adss/certification/csi
-action recover -client samples_test_client -mode xml -out
data/certification/output -alias Jhon -respondWith
pkcs12,certificate,pkcs7,password,expiry_date,pkcs10 -verbose
```

In the above example, the ADSS Test Tool utility sends a recover key to the ADSS Server over XML interface. The input certificate alias is used by the certification profile defined within ADSS Client Manager for this application to recover the key pair, identified by the client originator ID “samples_test_client”.

6.5 Certificate Change Password Request

For use when you wish to change the password for an existing key pair by sending request to ADSS Server

```
-service certification -server http://localhost:8777/adss/certification/csi
-action changePassword -client samples_test_client -mode xml -password
password -newPassword password123 -alias Jhon -verbose
```

In the above example, the ADSS Test Tool utility sends a change password request to the ADSS Server over XML interface. The input certificate alias is used by the certification profile defined within ADSS Client Manager for this application to change the password of an existing key pair, identified by the client originator ID "samples_test_client".

6.6 Certificate Revoke Request

For use when you wish to revoke an existing certificate by sending request to ADSS Server

```
-service certification -server http://localhost:8777/adss/certification/csi
-action revoke -client samples_test_client -mode xml -alias Jhon -
holdInstCode id-holdinstruction-none -revReason certificateHold -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate revoke request to the ADSS Server over XML interface. The input certificate alias is used by the certification profile defined within ADSS Client Manager for this application to revoke it, identified by the client originator ID "samples_test_client".

```
-service certification -server http://localhost:8777/adss/certification/csi
-action revoke -client samples_test_client -mode xml -serialNumber
3f0bd1a3ff15c5c2ccc69d3fb6d0ccb3951d4d6d -issuerDN "CN=ADSS Samples Test
CA,OU=Ascertia Software Distribution,O=Ascertia Limited,C=GB" -holdInstCode
id-holdinstruction-none -revReason certificateHold -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate revoke request to the ADSS Server over XML interface. The input certificate serial number and IssuerDN is used by the certification profile defined within ADSS Client Manager for this application to revoke it, identified by the client originator ID "samples_test_client".

Certificate which is revoked with hold instruction code option can be reinstate otherwise the certificate got revoked permanently

6.7 Certificate Reinstate Request

For use when you wish to reinstate an existing certificate by sending request to ADSS Server

```
-service certification -server http://localhost:8777/adss/certification/csi
-action revoke -client samples_test_client -mode xml -alias Jhon -
holdInstCode id-holdinstruction-none -revReason removeFromCRL -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate reinstate request to the ADSS Server over XML interface. The input certificate alias which is already revoked with hold instruction code is used by the certification profile defined within ADSS Client Manager for this application to reinstate it, identified by the client originator ID "samples_test_client".

Certificate which is revoked with hold instruction code option can only be reinstate

6.8 Certificate Delete Request

For use when you wish to delete an existing certificate by sending request to ADSS Server

```
-service certification -server http://localhost:8777/adss/certification/csi
-action delete -client samples_test_client -mode xml -alias John -verbose -
deleteParam pkcs12
```

In the above example, the ADSS Test Tool utility sends a certificate delete request to the ADSS Server over XML interface. The input certificate alias is used by the certification profile defined within ADSS Client Manager for this application to delete it, identified by the client originator ID “samples_test_client”.

If Certificate was generated on hardware crypto source e.g. HSM then the certificate not got deleted until it is configured inside Global Settings

If the -`deleteParam` is included in the request, the ADSS Server will permanently remove the corresponding certificate’s private key from its database.

6.9 Certificate Authorize Request

For use when you wish to associate an authorization certificate with the Qualified certificate by sending request to ADSS Server

```
-service certification -server http://localhost:8777/adss/certification/csi
-action authorize -client samples_test_client -mode xml -profile
adss:certification:profile:001 -out data/ -alias Jhon -verbose -
authCertAlias Jhon-Doe_Auth
```

In the above example, the ADSS Test Tool utility sends a certificate authorize request to the ADSS Server over XML interface. The input certificate alias is used by the certification profile defined within ADSS Client Manager for this application to associate it with a qualified certificate, identified by the client originator ID “samples_test_client”.

6.10 Certificate Unauthorize Request

For use when you wish to disassociate an authorization certificate from the Qualified certificate by sending request to ADSS Server

```
-service certification -server http://localhost:8777/adss/certification/csi
-action unauthorize -client samples_test_client -mode xml -profile
adss:certification:profile:001 -out data/ -alias Jhon  -verbose -
authCertAlias Jhon-Doe1_Auth
```

In the above example, the ADSS Test Tool utility sends a certificate unauthorize request to the ADSS Server over XML interface. The input certificate alias is used by the certification profile defined within ADSS Client Manager for this application to disassociate it from a qualified certificate, identified by the client originator ID “samples_test_client”.

6.11 Certification Request for Load Testing

User can optionally add batch count and request count to load test the Certification service, one example is shown here:

```
-service certification -server http://localhost:8777/adss/certification/csi
-mode xml -out data/certification/output -action create -client
samples_test_client -subject "CN=Jhon Doe,OU=Development,O=Ascertia,C=GB" -
password password -alias Jhon -batchCount 5 -requestCount 20
```

While doing load testing for a service try to avoid the use of verbose attribute in request because it will add a lot of I/O operations for dumping the request/response

6.12 Certificate Profile Info Request

This request is used to get specified ADSS Certification profile.

```
-service certification -server http://localhost:8777/adss/certification/csi
-action profileInfo -client samples_test_client -mode xml -out
data/ra/output -profile adss:certification:profile:001 -reqTimeOut 300 -
requestId Req-001 -verbose
```

6.13 Send SCT Request

This request is used to send SCT to the pre issued certificate and get the final certificate.

```
-service certification -server http://localhost:8777/adss/certification/csi
-action sendSCT -mode xml -out data/certification/output -alias test9 -
client samples_test_client -profile adss:certification:profile:002 -
sctVersion 0 -sctLogID CiCwzIPlpfl9a698CcwoSQSHKsfoixMsY1C3xv0m4Wxsdw== -
sctTimestamp 1599121313182 -sctSignature
CAQQAxpIMEYCIQCSDUGjtc1dYJbG6Wn/YtgqsK3RuaFRixnc5/j0F0NjJgIhAO8jiFaUmXLsr+Y0
eueB77JpIvl2uHaXFJtRnfyjbJJx -sctSeverAddress ct.googleapis.com/testtube -
verbose
```

In the above example, the ADSS Test Tool utility sends SCT information of given certificate alias to the ADSS Server over XML interface

6.14 Get Certificates Request

User can get a certificate on the basis of Subject DN, Serial Number, User ID, Status or CA Alias, or by using any combination of these parameters. However, it should be noted that the user must provide at least one certificate attribute amongst Subject DN, User ID, Serial Number and CA Alias in order to get the certificate. The user can also use pagination when calling this API by specifying the Start index and Max record information.

```
-service certification -server https://localhost:8779/adss/certification/csi
-clientAuthPfx "data/certification/input/certification_tls_client.pfx" -
clientAuthPfxPass password -action certificateInfo -mode xml -client
samples_test_client -subject "CN=Jhon Doe,OU=Development,O=Ascertia,C=GB" -
userId test-user -serialNumber 3f0bd1a3ff15c5c2ccc69d3fb6d0ccb3951d4d6d -
status ACTIVE -respondWith pkcs7 -caAlias "ADSS Samples Test CA" -startIndex
0 -maxRecords 20 -verbose
```

In the above example, the ADSS Test Tool utility sends request to get certificates to the ADSS Server over XML interface.

6.15 Create Certificate with CMC Protocol

In order to create a certificate with CMC Protocol, there are some pre-requisites that must to be fulfilled:

- We need to register the `clientAuthPfx` by creating it's key in Key Manager and registering the key in Client Manager module of ADSS Server.
- The '.p10' file must be placed to be placed in data folder, hence, data input path must be provided in the request.

```
-service certification -server https://localhost:8779/adss/certification/cmc
-action create -clientAuthPfx
"data/certification/input/certification_tls_client.pfx" -clientAuthPfxPass
password -out data/certification/output -profile
adss:certification:profile:001 -pkcs10
data/certification/input/sample_cmc.p10 -mode cmc -verbose -pkiRequestMode
simple
```

6.16 Revoke Statement with certificate hold with CMC Protocol

This request will be used to revoke a certificate with CMC protocol.

```
-service certification -server https://localhost:8779/adss/certification/cmc
-action revoke -alias 175103990491229587877414066965871795078474957243 -
clientAuthPfx "data/certification/input/certification_tls_client.pfx" -
clientAuthPfxPass password -mode cmc -issuerDN "CN=ADSS Samples Test
CA,OU=Ascertia Software Distribution,O=Ascertia Limited,C=GB" -revReason
certificateHold -verbose -profile adss:certification:profile:001 -
pkiRequestMode simple
```

6.17 Create Certificate with EST Protocol

In order to create a certificate with EST Protocol, there are some pre-requisites that must to be fulfilled. All EST requests will first authenticate a client before actually processing the request and client authentication mechanism allowed in EST protocol as mentioned below:

1. TLS Client Authentication
2. HTTP Basic Authentication
3. TLS Client Authentication with HTTP Basic Authentication

TLS Client Authentication:

In this authentication mechanism, client will be authenticated using its TLS Client Certificate. This certificate must be registered against a client in ADSS Client Manager.

For this authentication method we need to set the Client PFX file path in **clientAuthPfx** Parameter in Test Tool EST command.

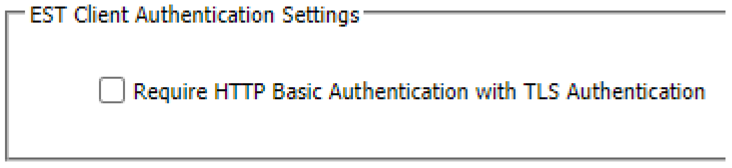
HTTP Basic Authentication:

In this authentication mechanism, the client will send its Client ID & Secret in request.

- We need to generate the **clientSecret** in Client Manager module of ADSS Server and use generated client secret in EST command to perform basic authentication for Mutual TLS communication with ADSS Server and we can use Non TLS communication for testing purpose with ADSS Server.

TLS Client Authentication with HTTP Basic Authentication:

If following checkbox is checked in Client Manager under Certification Service then we need to set client credentials (client id and client secret) and client PFX file in test tool command as well for TLS Communication.


EST Client Authentication Settings
Require HTTP Basic Authentication with TLS Authentication

6.18 Certificate Creation Request with EST Protocol (Simpleenroll API)

For use when you wish to create a certificate by sending request to ADSS Server

Non TLS Communication:

```
-service certification -server http://localhost:8777/.well-known/est/simpleenroll -action create -client samples_test_client -clientSecret 762b9a0c421fe4f64f47a6135545d436979450d6 -out data/certification/output -profile adss:certification:profile:001 -pkcs10 data/certification/input/sample.p10 -mode est -verbose
```

Mutual TLS Communication:

```
-service certification -server https://localhost:8779/.well-known/est/simpleenroll -clientAuthPfx "data/certification/input/tls_client.pfx" -clientAuthPfxPass password -action create  -out data/certification/output -profile adss:certification:profile:001 -pkcs10 data/certification/input/sample.p10 -mode est -verbose
```

```
-service certification -server https://localhost:8779/.well-known/est/simpleenroll -client samples_test_client -clientSecret 762b9a0c421fe4f64f47a6135545d436979450d6 -clientAuthPfx "data/certification/input/tls_client.pfx" -clientAuthPfxPass password -action create  -out data/certification/output -profile adss:certification:profile:001 -pkcs10 data/certification/input/sample.p10 -mode est -verbose
```

6.19 Certificate Rekey Request with EST Protocol (Simplereenroll API)

For use when you wish to rekeyed an existing certificate by sending request to ADSS Server

```
-service certification -server https://localhost:8779/.well-known/est/simplereenroll -action rekey -clientAuthPfx "data/certification/input/tls_client.pfx" -clientAuthPfxPass password -out data/certification/output -profile adss:certification:profile:001 -pkcs10 data/certification/input/sample.p10 -mode est -verbose
```

In the above example, the ADSS Test Tool utility sends a PKCS#10 rekey request to the ADSS Server over EST interface. In response, you receive the rekeyed certificate which is placed in output folder in .p7b format.

6.20 Certificate Renew Request with EST Protocol (Simplereenroll API)

For use when you wish to renew an existing certificate by sending request to ADSS Server

```
-service certification -server https://localhost:8779/.well-known/est/simplereenroll -action renew -clientAuthPfx
```

```
"data/certification/input/tls_client.pfx" -clientAuthPfxPass password -out
data/certification/output -profile adss:certification:profile:001 -pkcs10
data/certification/input/sample.p10 -mode est -verbose
```

In the above example, the ADSS Test Tool utility sends a PKCS#10 renew request to the ADSS Server over EST interface. In response, you receive the renewed certificate, which is placed in output folder in .p7b format.

6.21 Certificate Creation Request with EST Protocol (Fullcmc API)

For use when you wish to create a certificate by sending request to ADSS Server

```
-service certification -server https://localhost:8779/.well-
known/est/fullcmc -action create -clientAuthPfx
"data/certification/input/tls_client.pfx" -clientAuthPfxPass password -out
data/certification/output -profile adss:certification:profile:001 -pkcs10
data/certification/input/sample_cmc.p10 -mode est -verbose
```

6.22 Certificate Revoke Request with EST Protocol (Fullcmc API)

For use when you wish to revoke a certificate by sending request to ADSS Server

```
-service certification -server https://localhost:8779/.well-
known/est/fullcmc -action revoke -clientAuthPfx
"data/certification/input/tls_client.pfx" -clientAuthPfxPass password -
issuerDN "CN=ADSS Samples Test CA,OU=Ascertia Software
Distribution,O=Ascertia Limited,C=GB" -revReason certificateHold -alias
0c6266d6e296e1ff442b26f2c509f6eacc857f4b -mode est -verbose
```

6.23 Get CA Certificate with EST Protocol (cacerts API)

For use when you wish to get CA certificates by sending request to ADSS Server

```
-service certification -server https://localhost:8778/.well-
known/est/cacerts -action caCerts -client samples_test_client -profile
adss:certification:profile:001 -out data/certification/output -mode est -
verbose
```

6.24 Certificate and Key Pair Creation Request with EST Protocol (Serverkeygen API)

For use when you wish to create a certificate with newly generated key pair at server end by sending request to ADSS Server

```
-service certification -server https://localhost:8779/.well-
known/est/serverkeygen -action create -clientAuthPfx
"data/certification/input/tls_client.pfx" -clientAuthPfxPass password -out
data/certification/output -profile adss:certification:profile:001 -pkcs10
data/certification/input/sample.p10 -mode est -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate create request to the ADSS Server over EST interface. In response, you receive the certificate and private key as per PKCS8 standard. In output folder there are two files one is *.p7b certificate and other is *.pem file which contains newly generated private key.

7 Testing ADSS XKMS Service

The ADSS XKMS Service module provides advanced certificate validation services compliant with the W3C XKMS and PEPPOL validation protocol. The ADSS Server Test Tool can make requests to the ADSS XKMS Service asking for a certificate's status to be checked, i.e. it is issued by a trusted CA, expiry, revocation, and existence of particular key usages & extended key usages etc.

The ADSS Server Test Tool has an in-built Help feature, use the following command to get help on ADSS Test Tool usage for the XKMS Service:

```
-service xkms -help
```

The usage information will be shown on the console. The detail of each attribute is shown in the below table:

If a request parameter value contains space character(s) then, enclose such values within double quotation marks. To avoid this, ensure spaces are not used, e.g. in the file paths.

Action	Notes
-service	Specifies the ADSS Server service name. e.g. xkms
-server	Specifies the ADSS Server URL. e.g. http://localhost:8777/adss/xkms
-action	Specifies the XKMS operation to be performed. e.g. validate **Note:** Currently only validate operation is supported.
-requestId	Specifies an ID to be associated with the XKMS transaction. e.g. Request-001
-in	Specifies the path of the certificate file to be validated. e.g. “c:/in/Andy.cer” **Note:** Both DER and PEM encoded certificates are supported.
-out	Specifies the path for storing the response files. e.g. “c:/output_folder”
-respondWith	Specify this attribute to receive validation information back from the ADSS Server. Possible comma separated values are: • x509_cert • key_name • key_value • spki • x509_chain • x509_crl • ocsp • eid_quality • ocsp_no_cache • validation_details bullet

-client {optional}	Specifies the client originator ID to identify this client application to the ADSS Server. e.g. samples_test_client **Note:** See the ADSS Server Admin Guide for further details on managing client applications within ADSS Server.
-profile {optional}	Specifies a XKMS profile ID or Name e.g. adss:xkms:profile:001 **Note:** If this is not specified then the default XKMS profile configured in Client Manager will be used. Of course the profile name used in the command must already exist within the ADSS Verification Service.
-opaqueClientData {optional}	Specify a value that is opaque to the XKMS service. The XKMS service will return the value of the OpaqueClientData element unmodified in the response with status code Success. e.g. 25489651 **Note:** The length of the -opaqueClientData should not be greater than 256 bytes.
-keyUsage {optional}	Specify this attribute to get response on whether certificate is allowed for the following key usage signature, encryption, and exchange. Possible comma separated values are: • digitalSignature • nonrepudiation • keyAgreement
-useKeyWith {optional}	Specify a value for the smime, smtp and/or tls_https. XKMS service will verify that if the value for smime matches the email address in “Subject DN” or “Subject Alternative Name” of the target certificate. The values for smtp and tls_https are matched with the “Common Name” of the target certificate. e.g. smime=(email_address),tls_smtp=(smtp_server_address),tls_https=(web_server_address)
-validationTime {optional}	Specify the time in GMT at which certificate needed to be validated. The date/time format is (yyyy/MM/dd hh:mm:ss) e.g. 2016/08/22 23:08:55
-reqTimeOut {optional}	Specify the time period in seconds that test tool should wait for a response from the ADSS XKMS Service before closing the connection with it. e.g. 30
-soapVer {optional}	Specify the soap version to be used for composing the XML based request. Possible values are: • 1.1 • 1.2
-sigMode {optional}	Specify the signature type to be used for signing the XML based request. Possible values are: • Enveloped • Enveloping
-clientAuthPfx {optional}	Specify the path of the Client Authentication PFX e.g. C:/Certs/ClientAuthentication.pfx

	Note: This client authentication certificate must also be configured in ADSS Server Client Manager. Issuer CA of this client Authentication certificate must be registered inside Trust Manager with purpose CA for Verifying TLS Client certificates (After registering the Issuer CA, ADSS Server Core, Console and Service instances must be restarted from Windows Service panel or Unix Daemon). For detailed instructions on setting up TLS Client Authentication with ADSS Services, refer to the technical note titled ***'How to set up TLS Client Authentication communication with ADSS Services'*** in the Ascertia Community Portal knowledge base.
-clientAuthPfxPass {optional}	Specify the password for the Client Authentication PFX. e.g. password12
-reqSignPFX {optional}	Specify the path of the Request Signing PFX. e.g. C:/Certs/RequestSigning.pfx **Note:** This request signing certificate must also be configured in ADSS Server Client Manager.
-reqSignPfxPass {optional}	Specify the password for the Request Signing PFX. e.g. password12
-proxyHost {optional}	Specify the IP address or machine name of the proxy host. e.g. ProxyHostName or 192.168.102.12
-proxyPort {optional}	Specify the port no. for communication with proxy host. e.g. 8080
-proxyUser {optional}	Specify the user name for proxy authentication. e.g. user12
-proxyPass {optional}	Specify the user password for proxy authentication. e.g. password12
-proxyDigest {optional}	Specify this attribute if proxy digest authentication is required. e.g. -proxyDigest
-batchCount {optional}	Specify the number of batches to be executed for load testing. e.g. 3
-requestCount {optional}	Specify the number of concurrent requests to be executed within a batch for load testing. e.g. 100
-verbose {optional}	Specify this attribute to display request/response processing details on console and also store them on disk. e.g. -verbose **Note:** Request and response files will be stored at the location: [ADSS Server Test Tool Home]/verbose/

7.1 Certificate Validation Request

For use when you wish to validate Certificate by sending it to ADSS Server

Using Minimum Attributes

```
-service xkms -server http://localhost:8777/adss/xkms -action validate -in
data/xkms/input/test_input_signed.cer -out data/xkms/output -requestId
Request-001 -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate validation request to the ADSS Server over XKMS interface. The input certificate file is validated using the default XKMS Profile for Unauthenticated Clients defined within ADSS XKMS Service under Service Manager.

Using Optional Attributes

```
-service xkms -server http://localhost:8777/adss/xkms -action validate -in
data/xkms/input/test_input_signed.cer -out data/xkms/output -requestId
Request-001 -profile adss:xkms:profile:001 -respondWith
x509_cert,key_name,key_value,spki,x509_chain,x509_crl,ocsp,eid_quality,ocsp_
no_cache,validation_details -opaqueClientData Ascertia_1569 -keyUsage
digitalSignature,nonRepudiation -validationTime "2016/08/22 09:08:55" -
client samples_test_client -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate validation request to the ADSS Server having optional attributes. The input certificate file is validated using the XKMS Profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application.

*Historical validation must be enabled inside the XKMS profile under Advanced Settings if **-validationTime** attribute is provided in the request*

The signer certificate chain should be registered inside the Trust Manager and should also be added inside the XKMS Profile Trust Anchor settings

7.2 XKMS Request for Load Testing

User can optionally add batch count and request count to load test the XKMS Service, one example is shown here:

```
-service xkms -server http://localhost:8777/adss/xkms -action validate -in
data/xkms/input/test_input_signed.cer -out data/xkms/output -requestId
Request-001 -batchCount 5 -requestCount 20
```

While doing load testing for a service try to avoid the use of verbose attribute in request because it will add a lot of I/O operations for dumping the request/response

The signer certificate chain should be registered inside the Trust Manager and should also be added inside the XKMS Profile Trust Anchor settings

8 Testing ADSS SCVP Service

The ADSS SCVP Service provides a FIPS 201 and RFC 5055 compliant validation authority. The ADSS Server Test Tool can make requests to the ADSS SCVP Service asking for a certificate chain validation as well as complex delegated path discovery (DPD) also known as path building and delegated path validation (DPV).

The ADSS Server Test Tool has an in-built Help feature, use the following command to get help on ADSS Test Tool usage for the XKMS Service:

```
-service scvp -help
```

The usage information will be shown on the console. The detail of each attribute is shown in the below table:

If a request parameter value contains space character(s) then, enclose such values within double quotation marks. To avoid this, ensure spaces are not used, e.g. in the file paths.

Action	Notes
-service	Specifies the ADSS Server service name. e.g. scvp
-server	Specifies the ADSS Server URL. e.g. http://localhost:8777/adss/scvp
-requestType	Specify the request type. The possible values are CV/VP/policyInfo. If not provided CV request will be processed by default.
-in	Specifies the path of the certificate file to be validated. e.g. "c:/in/Andy.cer" **Note:** Both DER and PEM encoded certificates are supported.
-policy	Specifies a SCVP validation policy to be used e.g. 1.3.6.1.5.5.7.19.1 **Note:** Of course the validation policy OID used in the command must already exist within the ADSS SCVP Service.
-certChecks	Specifies whether the request type is DPD (Delegated Path Discovery) or DPV (Delegated Path Validation). Possible OID values are: • 1.3.6.1.5.5.7.17.1 (For DPD) • 1.3.6.1.5.5.7.17.3 (For DPV)
-userPolicySet {optional}	Specifies single/multiple comma separated certificate policy identifiers that the SCVP server must use when validating a certification path. e.g. 1.2.3.45.58, 1.2.4.58.78
-inhibitPolicyMapping {optional}	Specifies either policy mapping is allowed or not during the certification path validation. The possible values are: • True • False
-requireExplicitPolicy {optional}	Specifies an input to the certification path validation algorithm, and it controls whether there must be at least one valid policy in the certificate policies extension. When an explicit policy is required, it is necessary for all certificates in

	the path to contain an acceptable policy identifier in the certificate policies extension. The possible values are: • True • False
-inhibitAnyPolicy {optional}	Specifies an input to the certification path validation algorithm and it controls whether the anyPolicy OID is processed or ignored when evaluating certificate policy. The possible values are: • True • False **Note:** If the client wants the server to ignore the anyPolicy OID, inhibitAnyPolicy must be set to true in the request.
-out {optional}	Specifies the path for storing the response files. e.g. “c:/output_folder”
-wantBacks {optional}	Specify this attribute to receive validation information back from the SCVP Server. Possible comma separated values are: • best_cert_path • revocation_info • public_key_info • cert • all_cert_paths • ee_revocation_info • ca_revocation_info
-respFlags {optional}	Specifies this attribute to indicate SCVP server to include optional features in the certificate validation response. Possible comma separated values are: • fullReqInRes • resValidPolByRef • protectRes • cachedRes
-keyUsages {optional}	Specifies a single/multiple key usages to get response on whether certificate is allowed for the specified key usage or not during the certificate validation For AND operator use comma separated KU For OR operator use space separated KU bullet e.g. "digitalSignature,nonRepudiation keyAgreement"
-extendedKeyUsages {optional}	Specifies a single/multiple comma separated extended key usages to get response on whether certificate is allowed for the specified extended key usage or not during the certificate validation e.g. “timeStamping,emailSigning” **Note:** If emailSigning bit is set in extendedKeyUsages in SCVP request then SCVP service must check emailSigning bit in extended key usage in target certificate and also check it’s pre-requisite in key usages like digitalSignature and nonRepudiation.

-specifiedKeyUsages {optional}	Specifies a single/multiple comma separated specified values for extended key usages to get response on whether certificate contains the specified values for extended key usage or not during the certificate validation. e.g. "id-kp-serverAuth" **Note:** If id-kp-serverAuth bit is set in specifiedKeyUsages in SCVP request then SCVP service must check id-kp-serverAuth bit in extended key usage in target certificate and also check it's pre-requisite in key usages.
-hashAlgo {optional}	Specify this attribute if fullRequestInResponse attribute is set to true. This object identifier indicating which hash algorithm the server should use to compute the hash value for the requestHash item in the response. The possible values are: • sha1 • sha224 • sha256 • sha384 • sha512 • RipeMD128 • RipeMD160
-sigAlgo {optional}	Specify this attribute to indicate signature algorithm to be used by the SCVP server to sign the response message e.g. sha256withRSAEncryption
-validationAlgo {optional}	Specifies the validation algorithm to be used by the SCVP server during certificate validation. The value of this item can be determined by agreement between the client and the server. e.g. basicValidationAlgo
-validationNames {optional}	Specifies the subject name (comma separated RDNs) that must appear in the end entity certificate. e.g. cn=andy,c=gb,o=ascertia
-interCerts {optional}	Specifies the path of the intermediate certificate file for target certificate validation e.g. "c:/in/intermediate.cer" **Note:** Both DER and PEM encoded certificates are supported.
-trustAnchors {optional}	Specifies the single/multiple comma separated paths for the issuer certificate chain up to Root CA for target certificate validation. e.g. "c:/in/intermediate.cer,c:/in/root.cer" **Note:** Both DER and PEM encoded certificates are supported
-reqRefs {optional}	Specifies single/multiple comma separated key/value pairs for SCVP Server identification in case of SCVP relay. e.g. "dns_name==cn=abc,c=au,st=xyz&directory_name==c=au,st=xyz"
-reqName {optional}	Specifies the requester name (key/value pairs) that need to be included in response by SCVP Server in case of DPV. e.g. "dns_name==cn=abc,c=au,st=xyz"
-requestorText {optional}	Specifies a free text (e.g. reason for request) for inclusion in the SCVP response. e.g. "This validation call is for test purpose only"

-responderName {optional}	Specifies the SCVP responder name (key/value pairs) expected in SCVP response. e.g. dns_name==cn=abc,c=au,st=xyz
-validationTime {optional}	Specify the time in GMT at which certificate needed to be validated. The date/time format is (yyyy/MM/dd kk:mm:ss). e.g. 2009/08/20 09:30:04
-nonce {optional}	Specifies a nonce value (a unique number used once) to ensure that the response message is fresh and is not a replay. e.g. 56324
-reqTimeOut {optional}	Specify the time period in seconds that test tool should wait for a response from the ADSS SCVP Service before closing the connection with it. e.g. 30
-clientAuthPfx {optional}	Specify the path of the Client Authentication PFX e.g. C:/Certs/ClientAuthentication.pfx **Note:** This client authentication certificate must also be configured in ADSS Server Client Manager. Issuer CA of this client Authentication certificate must be registered inside Trust Manager with purpose CA for Verifying TLS Client certificates (After registering the Issuer CA, ADSS Server Core, Console and Service instances must be restarted from Windows Service panel or Unix Daemon). For detailed instructions on setting up TLS Client Authentication with ADSS Services, refer to the technical note titled ***'How to set up TLS Client Authentication communication with ADSS Services'*** in the Ascertia Community Portal knowledge base.
-clientAuthPfxPass {optional}	Specify the password for the Client Authentication PFX. e.g. password12
-reqSignPFX {optional}	Specify the path of the Request Signing PFX. e.g. C:/Certs/RequestSigning.pfx **Note:** This request signing certificate must also be configured in ADSS Server Client Manager.
-reqSignPfxPass {optional}	Specify the password for the Request Signing PFX. e.g. password12
-proxyHost {optional}	Specify the IP address or machine name of the proxy host. e.g. ProxyHostName or 192.168.102.12
-proxyPort {optional}	Specify the port no. for communication with proxy host. e.g. 8080
-proxyUser {optional}	Specify the user name for proxy authentication. e.g. user12
-proxyPass {optional}	Specify the user password for proxy authentication. e.g. password12
-proxyDigest {optional}	Specify this attribute if proxy digest authentication is required. e.g. -proxyDigest

-batchCount {optional}	Specify the number of batches to be executed for load testing. e.g. 3
-requestCount {optional}	Specify the number of concurrent requests to be executed within a batch for load testing. e.g. 100
-verbose {optional}	Specify this attribute to display request/response processing details on console and also store them on disk. e.g. -verbose **Note:** Request and response files will be stored at the location: [ADSS Server Test Tool Home]/verbose/

8.1 Certificate Validation Request

For use when you wish to validate Certificate by sending it to ADSS Server

Using Minimum Attributes

```
-service scvp -server http://localhost:8777/adss/scvp -in
data/scvp/input/sample.cer -policy 1.3.6.1.5.5.7.19.1 -certChecks
1.3.6.1.5.5.7.17.3 -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate validation (DPD) request to the ADSS Server over SCVP interface. The input certificate file is validated using the SCVP Validation Policy defined within ADSS SCVP Service.

Using Optional Attributes

```
-service scvp -server http://localhost:8777/adss/scvp -in
data/scvp/input/sample.cer -policy 1.3.6.1.5.5.7.19.1 -out data/scvp/output
-certChecks 1.3.6.1.5.5.7.17.1 -nonce -wantback
best_cert_path,revocation_info,public_key_info,cert,all_cert_paths,ee_revoca
tion_info,ca_revocation_info, -respFlags
fullReqInRes,resValidPolByRef,protectRes,cachedRes, -hashAlgo sha512 -
sigAlgo sha256withRSAEncryption -requestorText "Test Request" -userPolicySet
"1.2.3.45,2.3.56.78" -inhibitPolicyMapping true -requireExplicitPolicy true
-inhibitAnyPolicy false -trustAnchors
"data\scvp\input\IssuerCA.cer,data\scvp\input\RootCA.cer" -keyUsages
"digitalSignature,nonRepudiation keyAgreement" -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate validation (DPV) request to the ADSS Server having optional attributes. The input certificate file is validated using the SCVP Validation Policy defined within ADSS SCVP Service.

Either Root certificate or signer certificate chain should be registered inside the Trust Manager and should also be added inside the SCVP Profile Trust Anchor settings

8.2 Validation Policy Request

For use when you wish to see ADSS Server SCVP validation policy.

Using Minimum Attributes

```
-service scvp -server http://localhost:8777/adss/scvp -requestType VP -
verbose
```

In the above example, the ADSS Test Tool utility sends a request to get the validation policy configuration supported by ADSS SCVP Service. Cached validation policy response is returned.

Using Optional Attributes

```
-service scvp -server http://localhost:8777/adss/scvp -requestType VP -nonce
12345 -verbose
```

In the above example, the ADSS Test Tool utility sends a nonce parameter in addition to other basic parameters. This indicates the ADSS SCVP server to compute fresh response and do not send the cached response.

Either Root certificate or signer certificate chain should be registered inside the Trust Manager and should also be added inside the SCVP Profile Trust Anchor settings

8.3 Validation Policy Request Rest API

For use when you wish to see ADSS Server SCVP validation policies.

Using Minimum Attributes

```
-service scvp -server http://localhost:8777/adss/service/scvp -requestType
policyInfo -verbose
```

In the above example, the ADSS Test Tool utility sends a request to get the configurations of all the validation policies configured in ADSS SCVP Service.

Using Optional Attributes

```
-service scvp -server http://localhost:8777/adss/service/scvp -requestType
policyInfo -policy 1.3.6.1.5.5.7.19.1 -verbose
```

In the above example, the ADSS Test Tool utility sends a request to get the configurations of specific validation policy configured in ADSS SCVP Service.

Note: This command is only supported in Test Tool Java.

Either Root certificate or signer certificate chain should be registered inside the Trust Manager and should also be added inside the SCVP Profile Trust Anchor settings

8.4 SCVP Request for Load Testing

User can optionally add batch count and request count to load test the SCVP Service, one example is shown here:

```
-service scvp -server http://localhost:8777/adss/scvp -in
data/scvp/input/sample.cer -policy 1.3.6.1.5.5.7.19.1 -certChecks
1.3.6.1.5.5.7.17.3 -batchCount 5 -requestCount 20
```

While doing load testing for a service try to avoid the use of verbose attribute in request because it will add a lot of I/O operations for dumping the request/response

Either Root certificate or signer certificate chain should be registered inside the Trust Manager and should also be added inside the SCVP Profile Trust Anchor settings

9 Testing ADSS OCSP Service

The OCSP Service is an RFC6960-compliant certificate status validation service. It can respond for multiple CAs, using unique OCSP response signing keys and validation policies. It works in conjunction with the CRL Manager Service.

The ADSS Server Test Tool has an in-built Help feature, use the following command to get help on ADSS Test Tool usage for the XKMS Service:

```
-service ocsp -help
```

The usage information will be shown on the console. The detail of each attribute is shown in the below table:

If a request parameter value contains space character(s) then, enclose such values within double quotation marks. To avoid this, ensure spaces are not used, e.g. in the file paths.

Action	Notes
-service	Specifies the ADSS Server service name. e.g. ocsp
-server	Specifies the ADSS Server URL. e.g. http://localhost:8777/adss/ocsp
-userCert	Specifies the path of the target certificate file to be validated. e.g. “c:/in/Andy.cer” Note: Both DER and PEM encoded certificates are supported.
-issuerCert	Specifies the path of the Issuer certificate file for the validation of target certificate. e.g. “c:/in/issuer.cer” **Note:** Both DER and PEM encoded certificates are supported.
-serviceLocator {optional}	Specifies this attribute to include the service locator extension with in the OCSP request. e.g. -serviceLocator
-verifyResponse {optional}	Specify this attribute to verify the OCSP response. e.g. -verifyResponse
-prefSigAlgo {optional}	Specifies the single/multiple comma separated signature algorithm to be used for OCSP response signing (for OCSP responder) and response verification. Possible values are: • sha1withRSAEncryption • sha224withRSAEncryption • sha256withRSAEncryption • sha384withRSAEncryption • sha512withRSAEncryption • sha3-224withRSAEncryption • sha3-256withRSAEncryption • sha3-384withRSAEncryption • sha3-512withRSAEncryption

	• RipeMD128withRSA • RipeMD160withRSA • ecdsaWithSha1 • ecdsaWithSha224 • ecdsaWithSha256 • ecdsaWithSha384 • ecdsaWithSha512 **Note:** If provided signature algorithm is not supported by the ADSS OCSP service then unauthorized error will be returned.
-nonce {optional}	Specifies a nonce value (a unique number used once) to ensure that the response message is fresh and is not a replay. e.g. 28475
-hashAlgo {optional}	Specifies hashing algorithm to be used in composing CertID for OCSP request. Possible values are: • SHA1 • SHA224 • SHA256 • SHA384 • SHA512 • SHA3-224 • SHA3-256 • SHA3-384 • SHA3-512
-reqTimeOut {optional}	Specify the time period in seconds that test tool should wait for a response from the ADSS OCSP Service before closing the connection with it. e.g. 30
-clientAuthPfx {optional}	Specify the path of the Client Authentication PFX e.g. C:/Certs/ClientAuthentication.pfx **Note:** This client authentication certificate must also be configured in ADSS Server Client Manager. Issuer CA of this client Authentication certificate must be registered inside Trust Manager with purpose CA for Verifying TLS Client certificates (After registering the Issuer CA, ADSS Server Core, Console and Service instances must be restarted from Windows Service panel or Unix Daemon). For detailed instructions on setting up TLS Client Authentication with ADSS Services, refer to the technical note titled ***'How to set up TLS Client Authentication communication with ADSS Services'*** in the Ascertia Community Portal knowledge base.
-clientAuthPfxPass {optional}	Specify the password for the Client Authentication PFX. e.g. password12
-reqSignPFX {optional}	Specify the path of the Request Signing PFX. e.g. C:/Certs/RequestSigning.pfx **Note:** This request signing certificate must also be configured in ADSS Server Client Manager.

-reqSignPfxPass {optional}	Specify the password for the Request Signing PFX. e.g. password12
-proxyHost {optional}	Specify the IP address or machine name of the proxy host. e.g. ProxyHostName or 192.168.102.12
-proxyPort {optional}	Specify the port no. for communication with proxy host. e.g. 8080
-proxyUser {optional}	Specify the user name for proxy authentication. e.g. user12
-proxyPass {optional}	Specify the user password for proxy authentication. e.g. password12
-proxyDigest {optional}	Specify this attribute if proxy digest authentication is required. e.g. -proxyDigest
-batchCount {optional}	Specify the number of batches to be executed for load testing. e.g. 3
-requestCount {optional}	Specify the number of concurrent requests to be executed within a batch for load testing. e.g. 100
-verbose {optional}	Specify this attribute to display request/response processing details on console. e.g. -verbose **Note:** Support to store OCSP request and response files currently not available.

9.1 Certificate Status Validation Request

For use when you wish to validate Certificate status by sending it to ADSS Server

Using Minimum Attributes

```
-service ocsp -server http://localhost:8777/adss/ocsp -userCert
data/ocsp/input/test_input_target.cer -issuerCert
data/ocsp/input/test_input_issuer.cer -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate status validation request to the ADSS Server over OCSP interface. The input certificate file is validated using the validation settings defined within ADSS OCSP Service Registered CA settings.

Using Optional Attributes

```
-service ocsp -server http://localhost:8777/adss/ocsp -userCert
data/ocsp/input/test_input_target.cer -issuerCert
data/ocsp/input/test_input_issuer.cer -serviceLocator -verifyResponse -
prefSigAlgo SHA1withRSA,SHA256withRSA,SHA384withRSA,SHA512withRSA -nonce
25489651 -hashAlgo SHA256 -reqTimeOut 10 -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate status validation request to the ADSS Server having optional attributes. The input certificate file is validated using the validation settings defined within ADSS OCSP Service Registered CA settings.

The Issuer certificate should be registered inside the Trust Manager and should also be configured inside the OCSP Service

The Issuer certificate should be registered inside the Trust Manager and should also be configured inside the OCSP Service

9.2 OCSP Request for Load Testing

User can optionally add batch count and request count to load test the OCSP Service, one example is shown here:

```
-service ocsp -server http://localhost:8777/adss/ocsp -userCert
data/ocsp/input/test_input_target.cer -issuerCert
data/ocsp/input/test_input_issuer.cer -batchCount 5 -requestCount 20
```

While doing load testing for a service try to avoid the use of verbose attribute in request because it will add a lot of I/O operations for dumping the request/response

The Issuer certificate should be registered inside the Trust Manager and should also be configured inside the OCSP Service

10 Testing ADSS TSA Service

The TSA Service is an RFC3161-compliant service that provides secure timestamping for any type of data. It is possible to set-up multiple TSA instances with unique keys and optional proxy the TSA requests to other back-end time stamp authorities.

The ADSS Server Test Tool has an in-built Help feature, use the following command to get help on ADSS Test Tool usage for the XKMS Service:

```
-service tsa -help
```

The usage information will be shown on the console. The detail of each attribute is shown in the below table:

If a request parameter value contains space character(s) then, enclose such values within double quotation marks. To avoid this, ensure spaces are not used, e.g. in the file paths.

Action	Notes
-service	Specifies the ADSS Server service name e.g. tsa
-server	Specifies the ADSS Server URL e.g. http://localhost:8777/adss/tsa
-in	Specifies the path of the input file for timestamping e.g. “c:/input.pdf” For timestamp verification request, we will use this parameter to send the original data to set into the ExtData extension.
-policy {optional}	Specifies a TSA policy to be used e.g. 1.3.6.1.5 **Note:** Of course the TSA policy OID used in the command must already exist within the ADSS TSA Service.
-digestAlgo {optional}	Specifies the digest algorithm to be used for computing Message Imprint. Possible values are: • SHA1 • SHA224 • SHA256 • SHA384 • SHA512 • RipeMd160 • RipeMd128 • SHA3-224 • SHA3-256 • SHA3-384 • SHA3-512 **Note:** RipeMd160 is not supported in .NET Core 8.

-nonce {optional}	Specifies a nonce value (a unique number used once) to ensure that the response message is fresh and is not a replay. e.g. 28475
-reqCert {optional}	Specify this flag if it is required to include the TSA certificate in response. e.g. -reqCert
-verifyResponse {optional}	Specify this attribute to verify the TSA response. e.g. -verifyResponse
-reqTimeOut {optional}	Specify the time period in seconds that test tool should wait for a response from the ADSS TSA Service before closing the connection with it. e.g. 30
-action {optional}	This specifies the type of request, with possible values including: • create • renew • verify This parameter is mandatory when requesting a renew or verify timestamp. If it is not provided, the default type will be set to create.
-timeStampToken {optional}	This parameter is mandatory when the `–action` is set to renew or verify. It is used to send the timestamp token for either verification or renewal.
-extHashAlgo {optional}	This parameter specifies the digest algorithm for computing the message imprint for the ExtHash extension. It can accept multiple digest algorithms as a comma-separated list (e.g., SHA1, SHA224, SHA256, etc.). Possible values are: • SHA1 • SHA224 • SHA256 • SHA384 • SHA512 • RipeMd160 • RipeMd128 • SHA3-224 • SHA3-256 • SHA3-384 • SHA3-512 If multiple hash algorithms are provided, they will be used to compute a sequence of Message Imprints for the ExtHash extension. **Note:** RipeMd160 is not supported in .NET Core 8.
-extMethod {optional}	It specifies a comma-separated list of protection method OIDs for the 'ExtMethod' extension, for example: 1.0.18014.2.1
-requestId {optional}	Specify the request ID for the timestamp verification request. e.g. Request-01

-reqExtensions {optional}	It specifies the custom extensions (OID and value) to be added to the TSA request, For example: oid=2.5.4.5,value=123456789!!oid=2.16.484.101.10.316.20.37.1117,value=2023-08-10 23:00:00.
-clientAuthPfx {optional}	Specify the path of the Client Authentication PFX e.g. C:/Certs/ClientAuthentication.pfx **Note:** This client authentication certificate must also be configured in ADSS Server Client Manager. Issuer CA of this client Authentication certificate must be registered inside Trust Manager with purpose CA for Verifying TLS Client certificates (After registering the Issuer CA, ADSS Server Core, Console and Service instances must be restarted from Windows Service panel or Unix Daemon). For detailed instructions on setting up TLS Client Authentication with ADSS Services, refer to the technical note titled ***'How to set up TLS Client Authentication communication with ADSS Services'*** in the Ascertia Community Portal knowledge base.
-clientAuthPfxPass {optional}	Specify the password for the Client Authentication PFX. e.g. password12
-proxyHost {optional}	Specify the IP address or machine name of the proxy host. e.g. ProxyHostName or 192.168.102.12
-proxyPort {optional}	Specify the port no. for communication with proxy host. e.g. 8080
-proxyUser {optional}	Specify the user name for proxy authentication. e.g. user12
-proxyPass {optional}	Specify the user password for proxy authentication. e.g. password12
-proxyDigest {optional}	Specify this attribute if proxy digest authentication is required. e.g. -proxyDigest
-batchCount {optional}	Specify the number of batches to be executed for load testing. e.g. 3
-requestCount {optional}	Specify the number of concurrent requests to be executed within a batch for load testing. e.g. 100
-verbose {optional}	Specify this attribute to display request/response processing details on console and also store them on disk e.g. -verbose **Note:** Request and response files will be stored at the location: [ADSS Server Test Tool Home]/verbose/

10.1 Timestamping Request

For use when you wish to generate a secure timestamp token by sending it to ADSS Server

Using Minimum Attributes

```
-service tsa -server http://localhost:8777/adss/tsa -in
data/tsa/input/test_input_unsigned.txt -verbose
```

In the above example, the ADSS Test Tool utility sends a secure timestamp generation request to the ADSS Server over TSA interface. The secure timestamp token is generated against the input file using the settings defined within ADSS TSA Service Default TSA Policy.

Using Optional Attributes

```
-service tsa -server http://localhost:8777/adss/tsa -in
data/tsa/input/test_input_unsigned.txt -policy 1.2.3.4.5 -digestAlgo SHA256
-nonce 25489651 -reqCert -verifyResponse -reqTimeOut 10 -verbose
```

In the above example, the ADSS Test Tool utility sends a secure timestamp generation request to the ADSS Server having optional attributes. The secure timestamp token is generated against the input file using the settings defined within ADSS TSA Service Default TSA Policy.

The timestamp response signing certificate issuer chain up to Root CA should be registered inside the Trust Manager.

10.2 Timestamping Request Compliance with ISO/IEC

For use when you want to generate a secure timestamp token in accordance with ISO/IEC by sending it to the ADSS Server.

Create:

```
-service tsa -server http://localhost:8777/adss/tsa -in
data/tsa/input/test_input_unsigned.txt -policy 1.2.3.4.5 -digestAlgo SHA512
-nonce 25489651 -reqCert -verifyResponse -reqTimeOut 10 -verbose -
extHashAlgo SHA256,SHA384,SHA512,SHA3-512 -extMethod 1.0.18014.2.1 -action
create
```

In the above example, the ADSS Test Tool utility sends a secure timestamp generation request to the ADSS Server over TSA interface. The secure timestamp token is generated against the input file using the settings defined within ADSS TSA Service Default TSA Policy.

Renew:

```
-service tsa -server http://localhost:8777/adss/tsa -in
data/tsa/input/test_input_unsigned.txt -policy 1.2.3.4.5 -digestAlgo SHA256
-nonce 25489651 -reqCert -verifyResponse -reqTimeOut 10 -verbose -
extHashAlgo SHA256,SHA3-384,SHA3-512 -extMethod 1.0.18014.2.1 -
timeStampToken data/tsa/input/test_input_timestamp_token.pkcs7 -action renew
```

In the example above, the ADSS Test Tool utility sends a secure timestamp renewal request to the ADSS Server via the TSA interface. A secure, new timestamp token is generated based on the input file, utilizing the settings specified in the ADSS TSA Service Default TSA Policy.

10.3 Timestamp Verification Request Compliance with ISO/IEC

For use when you want to verify a timestamp token in accordance with ISO/IEC by submitting it to the ADSS Server.

```
-service tsa -server http://localhost:8777/adss/tsa -action verify -in
data/tsa/input/test_input_unsigned.txt -timeStampToken
data/tsa/input/test_input_timestamp_token.pkcs7 -reqTimeOut 100 -verbose -
requestId req-01
```

In the example above, the ADSS Test Tool utility submits a timestamp verification request to the ADSS Server through the TSA interface. The service will validate the timestamp token in accordance with ISO/IEC guidelines, responding with either a granted or rejected.

10.4 Timestamping Request for Load Testing

User can optionally add batch count and request count to load test the TSA Service, one example is shown here:

```
-service tsa -server http://localhost:8777/adss/tsa -in
data/tsa/input/test_input_unsigned.txt -batchCount 5 -requestCount 20
```

While doing load testing for a service try to avoid the use of verbose attribute in request because it will add a lot of I/O operations for dumping the request/response

The timestamp response signing certificate issuer chain up to Root CA should be registered inside the Trust Manager.

11 Testing ADSS LTANS Service

The ADSS LTANS Service provides long-term archiving and notary services that follow the IETF LTANS draft and RFC 4998 ERS standard. The ADSS Server Test Tool can make requests to the ADSS LTANS Service for generating evidence records (ERS) data.

The ADSS Server Test Tool has an in-built Help feature, use the following command to get help on ADSS Test Tool usage for the XKMS Service:

```
-service ltans -help
```

The usage information will be shown on the console. The detail of each attribute is shown in the below table:

If a request parameter value contains space character(s) then, enclose such values within double quotation marks. To avoid this, ensure spaces are not used, e.g. in the file paths.

Action	Notes
-service	Specifies the ADSS Server service name. e.g. ltans
-server	Specifies the ADSS Server URL. e.g. http://localhost:8777/adss/hltans **Note:** For more details see ADSS LTANS Service Interface URLs
-action	Specifies the LTANS operation to be performed. The possible values are: • archive • export • delete • verify • status • listids • renew **Note:** The required LTANS operation must be assigned to the client under Client Manager
-client	Specifies the client originator ID to identify this client application to the ADSS Server. e.g. samples_test_client **Note:** See the ADSS Server Admin Guide for further details on managing client applications within ADSS Server.
-in	Specifies the path of the input file for timestamping. e.g. “c:/input.pdf”
-type	Specifies the extension of the document to be archived. e.g. PDF **Note:** This can be any standard document extension or a custom extension.
-mode {optional}	Specifies the mode to be used by the client to send and receive the request/response from the ADSS Server. Possible values are: • XML (default)

	• HTTP If this parameter is not provided in the request, then default mode (XML) will be used. **Note:** HTTP refers to the use of plain HTTP requests using Ascertia defined HTTP protocol. DSS refers to the standard OASIS DSS protocol.
-tranId {optional}	Specifies an ID to be associated with the LTANS transaction. e.g. transaction-001
-profile {optional}	Specifies a LTANS profile ID or Name. e.g. adss:ltan:profile:001 **Note:** If this is not specified then the default LTANS profile configured in Client Manager will be used. Of course the profile name used in the command must already exist within the ADSS LTANS Service.
-metaItems {optional}	Specifies single/multiple comma separated key/value pairs, additional information need to be stored with the archive. e.g. Author=JohnDoe,Organization=Ascertia,City=Egham,Country=England **Note:** It is mandatory in case of export or listIds operation.
-reference {optional}	Specifies the reference ID of the archived data for export or delete operation only. e.g. 2408123338615240415 **Note:** It is mandatory in case of export or delete operation.
-out {optional}	Specifies the path for storing the original file which is archived earlier in case of export operation. e.g. “c:/output_folder/OriginalDocument.PDF” **Note:** It is also optional in case of export operation.
-remoteFilePath {optional}	Specifies the remote file path for the data to be picked/published for archive/export operation. e.g. “//AscertiaServer2/Documents/sample.doc”
-nonce {optional}	Specifies a nonce value (a unique number used once) to ensure that the response message is fresh and is not a replay. e.g. 28475
-serial {optional}	Specifies a serial number (a unique number used once) to be associated with LTANS transaction.
-reqTimeOut {optional}	Specify the time period in seconds that test tool should wait for a response from the ADSS LTANS Service before closing the connection with it. e.g. 30
-soapVer {optional}	Specify the soap version to be used for composing the XML based request. Possible values are: • 1.1 • 1.2
-sigMode {optional}	Specify the signature type to be used for signing the XML based request. Possible values are: • Enveloped

	• Enveloping
-clientAuthPfx {optional}	Specify the path of the Client Authentication PFX. e.g. C:/Certs/ClientAuthentication.pfx **Note:** This client authentication certificate must also be configured in ADSS Server Client Manager. Issuer CA of this client Authentication certificate must be registered inside Trust Manager with purpose CA for Verifying TLS Client certificates (After registering the Issuer CA, ADSS Server Core, Console and Service instances must be restarted from Windows Service panel or Unix Daemon). For detailed instructions on setting up TLS Client Authentication with ADSS Services, refer to the technical note titled ***'How to set up TLS Client Authentication communication with ADSS Services'*** in the Ascertia Community Portal knowledge base.
-clientAuthPfxPass {optional}	Specify the password for the Client Authentication PFX e.g. password12
-reqSignPFX {optional}	Specify the path of the Request Signing PFX e.g. C:/Certs/RequestSigning.pfx **Note:** This request signing certificate must also be configured in ADSS Server Client Manager.
-reqSignPfxPass {optional}	Specify the password for the Request Signing PFX. e.g. password12
-proxyHost {optional}	Specify the IP address or machine name of the proxy host. e.g. ProxyHostName or 192.168.102.12
-proxyPort {optional}	Specify the port no. for communication with proxy host. e.g. 8080
-proxyUser {optional}	Specify the user name for proxy authentication. e.g. user12
-proxyPass {optional}	Specify the user password for proxy authentication. e.g. password12
-proxyDigest {optional}	Specify this attribute if proxy digest authentication is required. e.g. -proxyDigest
-batchCount {optional}	Specify the number of batches to be executed for load testing. e.g. 3
-requestCount {optional}	Specify the number of concurrent requests to be executed within a batch for load testing. e.g. 100
-verbose {optional}	Specify this attribute to display request/response processing details on console and also store them on disk. e.g. -verbose **Note:** Request and response files will be stored at the location: [ADSS Server Test Tool Home]/verbose/

11.1 Archive Request

For use when you wish to archive a document by sending it to ADSS Server

Using Minimum Attributes

```
-service ltans -server http://localhost:8777/adss/ltap -action archive -
client samples_test_client -in data\ltans\input\test_input_signed.pdf -type
PDF -verbose
```

In the above example, the ADSS Test Tool utility sends an archiving request to the ADSS Server over Default XML interface. The input PDF file is archived using the default LTANS Profile defined within ADSS Client Manager for this application, identified by the client originator ID “samples_test_client”.

Using Optional Attributes

```
-service ltans -server http://localhost:8777/adss/hltans -action archive -
client samples_test_client -in data\ltans\input\test_input_signed.pdf -type
PDF -mode http -tranId Transaction_101 -profile adss:ltan:profile:001 -
metaItems Author=JohnDoe,Organization=Ascertia,City=Egham,Country=England -
nonce 25489651 -serial 0015544 -verbose
```

In the above example, the ADSS Test Tool utility sends an archiving request to the ADSS Server over HTTP interface having optional attributes. The input PDF file is archived using the LTANS Profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application.

11.2 Verify Request

For use when you wish to verify an already generated archive by sending a request to ADSS Server

Using Minimum Attributes

```
-service ltans -server http://localhost:8777/adss/ltap -action verify -
client samples_test_client -reference 338713421561493403 -verbose
```

In the above example, the ADSS Test Tool utility sends an archiving verify request to the ADSS Server over Default XML interface. The input archive reference number is verified using the default LTANS Profile defined within ADSS Client Manager for this application, identified by the client originator ID “samples_test_client”.

Using Optional Attributes

```
-service ltans -server http://localhost:8777/adss/hltans -action verify -
client samples_test_client -reference 338713421561493403 -mode http -profile
adss:ltan:profile:001 -metaItems
Author=JohnDoe,Organization=Ascertia,City=Egham,Country=England -nonce
25489651 -verbose
```

In the above example, the ADSS Test Tool utility sends an archive verify request to the ADSS Server over HTTP interface having optional attributes. The input archive reference number is verified using the LTANS Profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application.

11.3 Export Request

For use when you wish to export the original document using the archive reference number by sending a request to ADSS Server

```
-service ltans -server http://localhost:8777/adss/hltans -action export -
client samples_test_client -reference 338713421561493403 -mode http -profile
adss:ltan:profile:001 -out data\ltans\output\OriginalDocument.pdf -verbose
```

In the above example, the ADSS Test Tool utility sends an archive export request to the ADSS Server over HTTP interface having optional attributes. The input archive reference number is used to export the original document which is archived earlier using the LTANS Profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application.

11.4 Delete Request

For use when you wish to delete an existing archive data using the archive reference number by sending a request to ADSS Server

```
-service ltans -server http://localhost:8777/adss/hltans -action delete -
client samples_test_client -reference 338713421561493403 -mode http -verbose
```

In the above example, the ADSS Test Tool utility sends an archive deletion request to the ADSS Server over HTTP interface. The input archive reference number is used to delete the existing archive.

11.5 Status Request

For use when you wish to find the status of an existing archive data using the archive reference number by sending a request to ADSS Server

```
-service ltans -server http://localhost:8777/adss/hltans -action status -
client samples_test_client -reference 338713421561493403 -mode http -verbose
```

In the above example, the ADSS Test Tool utility sends an archive status request to the ADSS Server over HTTP interface. The input archive reference number is used to find the status of an existing archive.

11.6 ListID Request

For use when you wish to find the list of all the archives generated by a client so far using meta items by sending a request to ADSS Server

```
-service ltans -server http://localhost:8777/adss/hltans -action listids -
client samples_test_client -mode http -metaItems
Author=JohnDoe,Organization=Ascertia,City=Egham,Country=England -verbose
```

In the above example, the ADSS Test Tool utility sends an archive listsids request to the ADSS Server over HTTP interface. The input Meta Item is used as search criteria to find all the archive having the same Meta Items (generated by this client only).

11.7 Renew Request

For use when you wish to renew an existing archive by sending a request to ADSS Server

```
-service ltans -server http://localhost:8777/adss/hltans -action renew -
client samples_test_client -reference 338713421561493403 -mode http -verbose
```

In the above example, the ADSS Test Tool utility sends an archive renew request to the ADSS Server over HTTP interface. The input archive reference number is used to find and renew the existing archive.

11.8 Archive Request for Load Testing

User can optionally add batch count and request count to load test the LTANS Service, one example is shown here:

```
-service ltans -server http://localhost:8777/adss/ltap -action archive -
client samples_test_client -in data\ltans\input\test_input_signed.pdf -type
PDF -batchCount 5 -requestCount 20
```

While doing load testing for a service try to avoid the use of verbose attribute in request because it will add a lot of I/O operations for dumping the request/response

12 Testing ADSS RA Service

The ADSS RA Service receives and manages certificate signing requests (CSRs, also known as PKCS#10 / CSRs) from end-entities that can include human users, web servers, network devices and other entities. ADSS RA Service also manages user registration, certificate enrolment etc for ADSS server Authorise Remote Signing solution.

The ADSS Server Test Tool has an in-built Help feature, use the following command to get help on ADSS Test Tool usage for the Verification Service:

```
-service registration -help
```

The usage information will be shown on the console. The detail of each attribute is shown in the below table:

If a request parameter value contains space character(s) then, enclose such values within double quotation marks. To avoid this, ensure spaces are not used, e.g. in the file paths.

Action	Notes
-service	Specifies the ADSS Server service name. e.g. registration
-server	Specifies the ADSS Server URL. e.g. http://localhost:8777/adss/ra/cri Note: For more details see ADSS RA Service Interface URLs
-action	Specifies the registration operation to be performed. The possible values are: • create • revoke • status • delete • renew • rekey • registerUser • updateUser • deleteUser • getUser • getUsers • getUserDevices • getUserCertificates • deleteUserDevice • changePassword • recoverPassword • confirmRecoverPassword • changeEmail • confirmChangeEmail • changeMobile

	• confirmchangeMobile • profileInfo **Note:** Following parameters must be used with registerUser option: • -userName • -userId • -mobileNumber • -emailAddress
-client	Specifies the client originator ID to identify this client application to the ADSS Server. e.g. samples_test_client **Note:** See the ADSS Server Admin Guide for further details on managing client applications within ADSS Server.
-mode {optional}	Specifies the mode to be used by the client to send and receive the request/response from the ADSS Server. Possible values are: XML (default) SCEP If this parameter is not provided in the request, then default mode (XML) will be used. “XML” option uses the original Ascertia Proprietary Protocol. Note: Following parameters must be used with SCEP option: • - requestMethod • -operation • -keySize • -encCert
-requestMethod	Specifies the request method to be used when mode SCEP is used. Possible values are: • get • post **Note:** This parameter is mandatory only when SCEP mode is used.
-operation	Specifies a SCEP operation to be performed. Possible values are: • pkcsreq • getcert • getcacert • getcacaps **Note:** This parameter is mandatory only when SCEP mode is used.
-keySize	Specifies the size of the SCEP Key to be generated. Possible values are: • 1024 • 2048 • 3072 • 4096 **Note:** This parameter is mandatory only when SCEP mode is used.
-encCert	Specifies the path of the encryption certificate to encrypt the SCEP request

	e.g. C:\sampleCertificate.cer **Note:** This parameter is mandatory only when SCEP mode is used.
-senderNonce	Specifies a nonce value (a unique number used once) to ensure that the response message is fresh and is not a replay. e.g. 28475 **Note:** This parameter is mandatory only when SCEP mode is used.
-userName	Specifies the name of the user/device administrator. e.g. "John Doe"
-userId	Specifies the user friendly id of the user/device administrator. e.g. "John" **Note:** It is mandatory in case of register user request
-mobileNumber	Specifies the mobile number of the user/device administrator. e.g. "00445214410225" **Note:** It is mandatory in case of register user request
-emailAddress	Specifies the email address of the user/device administrator. e.g. "john@ascertia.com **Note:** It is mandatory in case of register user request
-userStatus	Specifies the status of registered user. Possible values are: • ACTIVE • INACTIVE • BLOCKED
-requestId {optional}	Specifies an ID to be associated with the registration transaction. e.g. REQ-001
-transactionId {optional}	Specifies an ID to be used for asynchronous request. e.g. transaction-001
-profile {optional}	Specifies a registration profile ID or Name. e.g. adss:registration:profile:001 **Note:** If this is not specified then the default registration profile configured in Client Manager will be used. Of course the profile name used in the command must already exist within the ADSS RA Service.
-alias {optional}	Specifies a certificate alias for the certificate to be generated. e.g. test-certificate
-subject {optional}	Specifies the Subject DN for the certificate to be created. e.g. "CN=John Doe,OU=Ascertia Software Distribution,O=Ascertia Limited,C=GB" **Note:** This parameter is mandatory if the parameter -pkcs10 is not used.
-password {optional}	Specifies the password for the certificate (PKCS#12) to be created. e.g. password123

-challengePassword {optional}	Specifies a random value generated in the Device sub-module of ADSS Server and provided to the device to communicate securely with ADSS Server. e.g. password321 **Note:** This parameter is used only when SCEP mode is used.
-pkcs10 {optional}	Specifies the PKCS10 which will be used to create the certificate. e.g. C:/certs/JohnDoe.p10 **Note:** This parameter is mandatory if the parameter -subject is not used.
-san {optional}	Specifies the Subject Alternative Name for certificate to be created. Possible values are: rfc822Name==value1~value2&dNSName==value1~value2& iPAddress==value1~value2&uniformResourceIdentifier==value1~value2& otherName==OID=value,encoding=UTF8String~OID=value, encoding=OctetString~OID=value,encoding=PrintableString &directoryName==value1~value2 e.g. "rfc822Name==jhon1@ascertia.com~jhon2@ascertia.com&dNSName==www.ascertia.com,www.ascertia2.com&iPAddress==192.168.1.1~192.168.10.10&uniformResourceIdentifier==value1~value2&otherName==1.2.3.4.5.6=Test Value,encoding=UTF8String~1.2.3.4.5.7=test value 2,encoding=OctetString&directoryName==CN=Jhon Doe1,OU=Development,O=Ascertia,C=GB~CN=Jhon2,OU=HR,O=Ascertia,C=GB"
-custom Extension	Specifies the Custom extension for certificate to be created. Possible values are: "oid=.3.6.1.4.1.59382.3.2~value=value1&oid=1.3.6.1.4.1.59382.3.3-value=value2”
-validityPeriod {optional}	Specify the validity period for the certificate to be created or renewed. e.g. 12
-validityUnit {optional}	Specify the validity period unit of the certificate in following possible values: • MINS • HOURS • DAYS • MONTHS • YEARS
-serialNumber {optional}	Specifies the serial number of the target certificate. e.g. 254GT52 **Note:** This parameter is used only when SCEP mode is used.
-signatureAlgo {optional}	Specifies the signature algorithm used to sign the SCEP request. Possible values are: • SHA1WithRSAEncryption • SHA256WithRSAEncryption • SHA384WithRSAEncryption • SHA512WithRSAEncryption **Note:** This parameter is used only when SCEP mode is used.

-revReason {optional}	Specifies a revocation reason while revoking a certificate. The possible values are: • unspecified (Default) • keyCompromise • cACompromise • affiliationChanged • superseded • cessationOfOperation • certificateHold • removeFromCRL • privilegeWithdrawn • aACompromise **Note:** If this attribute is not added in the revoke request then certificate will be revoked with reason code unspecified.
-holdInstCode {optional}	Specifies the hold instruction code if the revocation reason is certificateHold. Possible values are • id-holdinstruction-none (Default) • id-holdinstruction-callissuer • id-holdinstruction-rejec **Note:** If revocation reason is other than certificateHold then this attribute will be ignored. If revocation reason is certificateHold and this attribute is not provided in the request then default (id-holdinstruction-none) value is used.
-out {optional}	Specifies the path for storing the response files. e.g. “c:/output_folder”
-reqTimeOut {optional}	Specify the time period in seconds that test tool should wait for a response from the ADSS RA Service before closing the connection with it. e.g. 30
-soapVer {optional}	Specify the soap version to be used for composing the XML based request. Possible values are: • 1.1 • 1.2
-sigMode {optional}	Specify the signature type to be used for signing the XML based request. Possible values are: • Enveloped • Enveloping
-clientAuthPfx {optional}	Specify the path of the Client Authentication PFX. e.g. C:/Certs/ClientAuthentication.pfx **Note:** This client authentication certificate must also be configured in ADSS Server Client Manager. Issuer CA of this client Authentication certificate must be registered inside Trust Manager with purpose CA for Verifying TLS Client certificates (After registering the Issuer CA, ADSS Server Core, Console and Service instances must be restarted from Windows Service panel or Unix Daemon). For detailed instructions on setting up TLS Client Authentication with ADSS Services, refer to the technical

	note titled ***'How to set up TLS Client Authentication communication with ADSS Services'*** in the Ascertia Community Portal knowledge base.
-clientAuthPfxPass {optional}	Specify the password for the Client Authentication PFX. e.g. password12
-reqSignPFX {optional}	Specify the path of the Request Signing PFX. e.g. C:/Certs/RequestSigning.pfx **Note:** This request signing certificate must also be configured in ADSS Server Client Manager.
-reqSignPfxPass {optional}	Specify the password for the Request Signing PFX. e.g. password12
-proxyHost {optional}	Specify the IP address or machine name of the proxy host. e.g. ProxyHostName or 192.168.102.12
-proxyPort {optional}	Specify the port no. for communication with proxy host. e.g. 8080
-proxyUser {optional}	Specify the user name for proxy authentication. e.g. user12
-proxyPass {optional}	Specify the user password for proxy authentication. e.g. password12
-proxyDigest {optional}	Specify this attribute if proxy digest authentication is required. e.g. -proxyDigest
-batchCount {optional}	Specify the number of batches to be executed for load testing. e.g. 3
-requestCount {optional}	Specify the number of concurrent requests to be executed within a batch for load testing. e.g. 100
-verbose {optional}	Specify this attribute to display request/response processing details on console and also store them on disk. e.g. -verbose **Note:** Request and response files will be stored at the location: [ADSS Server Test Tool Home]/verbose/

12.1 Certificate Creation Request

For use when you wish to create a certificate by sending request to ADSS Server

By default RA profile is set to Asynchronous mode and after sending the create request operator need to approve the pending request from ADSS Server console. For more details see the End-user Certificates

Using Minimum Attributes

Sending SubjectDN

```
-service registration -server http://localhost:8777/adss/ra/cri -mode xml -
out data/ra/output -action create -client samples_test_client -subject
"CN=John Doe,OU=Development,O=Ascertia,C=GB" -password password -alias John
-userName Jhon-001 -emailAddress johndoe@ascertia.com -verbose
```

Sending PKCS#10

```
-service registration -server http://localhost:8777/adss/ra/cri -mode xml -
out data/ra/output -action create -client samples_test_client -pkcs10
data/ra/input/sample.p10 -password password -alias JohnPKCS10 -userName
Jhon-002 -emailAddress johndoe2@ascertia.com -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate creation request to the ADSS Server over XML interface. The input subjectDN/PKCS10 is used by the default RA profile defined within ADSS Client Manager for this application to generate the certificate, identified by the client originator ID “samples_test_client”.

Using Optional Attributes

```
-service registration -server http://localhost:8777/adss/ra/cri -mode xml -
out data/ra/output -action create -client samples_test_client -subject
"CN=John Doe,OU=Development,O=Ascertia,C=GB" -password password -alias
John.doe -userName John-001 -emailAddress johndoe@ascertia.com -profile
adss:ra:profile:001 -requestId Create-001 -keySize 2048 -keyType RSA -san
"rfc822Name==johndoe@ascertia.com&dNSName==www.ascertia.com" -validityPeriod
36 -issuerDN "CN=ADSS Samples Test CA,OU=Ascertia Software
Distribution,O=Ascertia Limited,C=GB" -reqTimeOut 30 -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate creation request to the ADSS Server having optional attributes. The input subjectDN is used to generate the certificate by the RA profile defined within the request, identified by the client originator ID “samples_test_client” defined within ADSS Client Manager for this application.

Operator need to enable the certificate extensions inside the relevant certificate template e.g. ***Default Document Signing Template*** *under key manager otherwise the SAN will not become the part of the certificate. For more details see Certificate Templates.*

12.2 Certificate Revoke Request

For use when you wish to revoke an existing certificate by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
revoke -client samples_test_client -mode xml -alias John -holdInstCode id-
holdinstruction-none -revReason certificateHold -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate revoke request to the ADSS Server over XML interface. The input certificate alias is used by the RA profile defined within ADSS Client Manager for this application to revoke it, identified by the client originator ID "samples_test_client".

Certificate which is revoked with hold instruction code option can be reinstate otherwise the certificate got revoked permanently

12.3 Certificate Reinstate Request

For use when you wish to revoke an existing certificate by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
revoke -client samples_test_client -mode xml -alias John -holdInstCode id-
holdinstruction-none -revReason removeFromCRL -verbose
```

In the above example, the ADSS Test Tool utility sends a certificate reinstate request to the ADSS Server over XML interface. The input certificate alias which is already revoked with hold instruction code is used by the RA profile defined within ADSS Client Manager for this application to reinstate it, identified by the client originator ID "samples_test_client".

Certificate which is revoked with hold instruction code option can only be reinstate

12.4 Certificate Renew Request

For use when you wish to renew an existing certificate by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
renew -client samples_test_client -mode xml -alias John -verbose -subject
CN=John
```

12.5 Certificate Rekey Request

For use when you wish to rekey an existing certificate by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
rekey -client samples_test_client -mode xml -alias John -verbose -subject
CN=John
```

12.6 Certificate Delete Request

For use when you wish to delete an existing certificate by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
delete -client samples_test_client -mode xml -alias John -verbose
```

12.7 Register User Request

For use when you wish to register user by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
registerUser -client samples_test_client -mode xml -profile
adss:ra:profile:001 -userId Alice -userName Alice -password password -
mobileNumber +44789456123 -emailAddress info@ascertia.com -verbose
```

In the above example, the ADSS Test Tool utility sends a user registration request to the ADSS Server over XML interface. The input user information (userId, userName, mobileNumber, emailAddress) is used by the RA profile defined within ADSS Client Manager for this application to update user, identified by the client originator ID "samples_test_client".

For use when you wish to generate User key enrolment certificate for the above mentioned registered user

```
-service registration -server http://localhost:8777/adss/ra/cri -mode xml -
out data/ -action create -client samples_test_client -profile
adss:ra:profile:001 -subject CN=Alice,OU=Development,O=Ascertia,C=GB -
password password -userId Alice -alias Alice -verbose
```

This commands will only be successful if SAM configurations are done in ADSS Server. For more details please see the Quick-Guide-for-ADSS-SAM-Deployment available under doc directory of ADSS Server package.

12.8 Update User Request

For use when you wish to update registered user by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
updateUser -client samples_test_client -mode xml -userId Alice -userName
"Alice John" -userStatus INACTIVE -mobileNumber +44789456321 -emailAddress
support@ascertia.com -verbose
```

This commands will only be successful if SAM configurations are done in ADSS Server. For more details please see the Quick-Guide-for-ADSS-SAM-Deployment available under doc directory of ADSS Server package.

12.9 Delete User Request

For use when you wish to delete registered user by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
deleteUser -client samples_test_client -mode xml -profile
adss:ra:profile:001 -userId Alice -verbose
```

In the above example, the ADSS Test Tool utility sends delete user request to the ADSS Server over XML interface. The input userId is used by the RA profile defined within ADSS Client Manager for this application to delete user, identified by the client originator ID "samples_test_client".

This commands will only be successful if SAM configurations are done in ADSS Server. For more details please see the Quick-Guide-for-ADSS-SAM-Deployment available under doc directory of ADSS Server package.

12.10 Get User Request

For use when you wish to get registered user information (user name, email address, mobile number, user status) by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
getUser -client samples_test_client -mode xml -userId Alice -verbose
```

In the above example, the ADSS Test Tool utility sends get user request to the ADSS Server over XML interface. The input userId is used by the RA profile defined within ADSS Client Manager for this application to get user information, identified by the client originator ID “samples_test_client”.

This commands will only be successful if SAM configurations are done in ADSS Server. For more details please see the Quick-Guide-for-ADSS-SAM-Deployment available under doc directory of ADSS Server package.

12.11 Get Users Request

For use when you wish to get list of users register to a specific RA client by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
getUsers -client samples_test_client -mode xml -verbose
```

In the above example, the ADSS Test Tool utility sends get users request to the ADSS Server over XML interface. The input client is used by the RA profile defined within ADSS Client Manager for this application to get register users list, identified by the client originator ID “samples_test_client”.

This commands will only be successful if SAM configurations are done in ADSS Server. For more details please see the Quick-Guide-for-ADSS-SAM-Deployment available under doc directory of ADSS Server package.

12.12 Get User Devices Request

For use when you wish to get list of devices register to a specific user by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
getUserDevices -client samples_test_client -mode xml -userId Alice -verbose
```

In the above example, the ADSS Test Tool utility sends get user devices request to the ADSS Server over XML interface. The input userId is used by the RA profile defined within ADSS Client Manager for this application to get register devices list, identified by the client originator ID “samples_test_client”.

This commands will only be successful if SAM configurations are done in ADSS Server. For more details please see the Quick-Guide-for-ADSS-SAM-Deployment available under doc directory of ADSS Server package.

12.13 Delete Device Request

For use when you wish to registered device of a user by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
deleteUserDevice -client samples_test_client -mode xml -userId Alice -
deviceId 28fd4763-15b0-4a05-837e-475b3dec6a91 -verbose
```

In the above example, the ADSS Test Tool utility sends delete user request to the ADSS Server over XML interface. The input userId and deviceId is used by the RA profile defined within ADSS Client Manager for this application to delete user device, identified by the client originator ID "samples_test_client".

This commands will only be successful if SAM configurations are done in ADSS Server. For more details please see the Quick-Guide-for-ADSS-SAM-Deployment available under doc directory of ADSS Server package.

12.14 Get User Certificates Request

For use when you wish to get list of certificates register to a specific user by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
getUserCertificates -client samples_test_client -mode xml -profile
adss:ra:profile:001 -userId Alice -verbose
```

In the above example, the ADSS Test Tool utility sends get user certificates request to the ADSS Server over XML interface. The input userId is used by the RA profile defined within ADSS Client Manager for this application to get user certificates list, identified by the client originator ID "samples_test_client".

This commands will only be successful if SAM configurations are done in ADSS Server. For more details please see the Quick-Guide-for-ADSS-SAM-Deployment available under doc directory of ADSS Server package.

12.15 Change Password Request

For use when you wish to change password of a registered user by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
changePassword -client samples_test_client -mode xml -userId Alice -
userOldPassword password -userNewPassword p@ssword12 -verbose
```

In the above example, the ADSS Test Tool utility sends change user password request to the ADSS Server over XML interface. The input userId is used by the RA profile defined within ADSS Client Manager for this application to change user password, identified by the client originator ID "samples_test_client".

This commands will only be successful if SAM configurations are done in ADSS Server. For more details please see the Quick-Guide-for-ADSS-SAM-Deployment available under doc directory of ADSS Server package.

12.16 Recover Password Request

For use when you wish to reset password of a registered user by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
recoverPassword -client samples_test_client -mode xml -userId Alice -verbose
```

In the above example, the ADSS Test Tool utility sends recover user password request to the ADSS Server over XML interface. The input userId is used by the RA profile defined within ADSS Client Manager for this application to reset user password, identified by the client originator ID "samples_test_client".

*This call just reset the registered user password and return OTP via Email and SMS, the two OTP's are used in the "**Confirm Recover Password**" Request to permanently change password.*

This commands will only be successful if SAM configurations are done in ADSS Server. For more details please see the Quick-Guide-for-ADSS-SAM-Deployment available under doc directory of ADSS Server package.

12.17 Confirm Recover Password Request

For use when you wish to permanently reset password of a registered user by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
confirmRecoverPassword -client samples_test_client -mode xml -userId Alice -
emailOtp 123456789 -mobileOtp 123456789 -userNewPassword p@ssword12 -verbose
```

In the above example, the ADSS Test Tool utility sends confirm recover user password request to the ADSS Server over XML interface. The input userId, emailOtp, mobileOtp and user new password is used by the RA profile defined within ADSS Client Manager for this application to reset user password, identified by the client originator ID "samples_test_client".

This commands will only be successful if SAM configurations are done in ADSS Server. For more details please see the Quick-Guide-for-ADSS-SAM-Deployment available under doc directory of ADSS Server package.

12.18 Change Email Request

For use when you wish to change email of a registered user by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
changeEmail -client samples_test_client -mode xml -userId Alice -
userNewEmail info@ ascertia.com -verbose
```

In the above example, the ADSS Test Tool utility sends change user email request to the ADSS Server over XML interface. The input userId and userNewEmail is used by the RA profile defined within ADSS Client Manager for this application to change user email, identified by the client originator ID "samples_test_client".

*This call just returns OTP via Email and SMS, the two OTP's are used in the "**Confirm change email**" request to permanently change user email.*

This commands will only be successful if SAM configurations are done in ADSS Server. For more details please see the Quick-Guide-for-ADSS-SAM-Deployment available under doc directory of ADSS Server package.

12.19 Confirm Change Email Request

For use when you wish to confirm change email of a registered user by sending request to ADSS Server

```
-service   registration   -server   http://localhost:8777/adss/ra/cri   -action
confirmChangeEmail  -client  samples_test_client  -mode  xml  -userId  Alice  -
emailOtp        123456789        -mobileOtp        123456789         -verbose
```

In the above example, the ADSS Test Tool utility sends confirm change user email request to the ADSS Server over XML interface. The input userId, emailOtp and mobileOtp is used by the RA profile defined within ADSS Client Manager for this application to confirm change user email, identified by the client originator ID “samples_test_client”.

This commands will only be successful if SAM configurations are done in ADSS Server. For more details please see the Quick-Guide-for-ADSS-SAM-Deployment available under doc directory of ADSS Server package.

12.20 Change Mobile Number Request

For use when you wish to change mobile number of a registered user by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
changeMobile -client samples_test_client -mode xml -userId Alice -
userNewMobile +44789456321 -verbose
```

In the above example, the ADSS Test Tool utility sends change user mobile request to the ADSS Server over XML interface. The input userId and userNewMobile is used by the RA profile defined within ADSS Client Manager for this application to change user mobile number, identified by the client originator ID “samples_test_client”.

*This call just returns OTP via Email and SMS, the two OTP’s are used in the “**Confirm change mobile**” request to permanently change user mobile.*

This commands will only be successful if SAM configurations are done in ADSS Server. For more details please see the Quick-Guide-for-ADSS-SAM-Deployment available under doc directory of ADSS Server package.

12.21 Confirm Change Mobile Request

For use when you wish to confirm change mobile number of a registered user by sending request to ADSS Server

```
-service registration -server http://localhost:8777/adss/ra/cri -action
confirmchangeMobile -client samples_test_client -mode xml -userId Alice -
emailOtp 123456789 -mobileOtp 123456789 -verbose
```

In the above example, the ADSS Test Tool utility sends confirm change user mobile request to the ADSS Server over XML interface. The input userId, emailOtp and mobileOtp is used by the RA profile defined within ADSS Client Manager for this application to confirm change user mobile number, identified by the client originator ID “samples_test_client”.

This commands will only be successful if SAM configurations are done in ADSS Server. For more details please see the Quick-Guide-for-ADSS-SAM-Deployment available under doc directory of ADSS Server package.

12.22 RA Profile Info Request

This request is used to get specified ADSS RA profile.

```
-service registration -server http://localhost:8777/adss/ra/cri -action
profileInfo -client samples_test_client -mode xml -requestId Req-001 -out
data/ra/output  -profile adss:ra:profile:001 -reqTimeOut 300 -verbose
```

13 Support for special Characters

ADSS Server Test tool by default cannot handle the special character support via console. An alternative approach can be used achieve this. Please follow these instructions:

- Open a text editor which supports special character encoding
- Write your command in it having (special characters)
- Save this text file on disk e.g. command.txt
- Launch ADSS Server Test Tool, run this query to execute your command saved in text file e.g.

  `-commandFilePath “d:\command.txt”`

13.1 Displaying Arabic Characters in Command Prompt

If Arabic characters do not appear correctly when using the ADSS Server Test Tool in the Command Prompt, follow these steps to ensure proper character encoding:

1. Go to the **Test Tool installation directory**.
2. Open the **`setclasspath.bat`** file in a text editor.
3. Locate the line that contains **`chcp`** (which sets the code page for the Command Prompt).
4. Change its value to **`chcp 65001`**, which sets the code page to UTF-8 to support Arabic characters.
5. Save and close the file.

This configuration ensures that the Command Prompt uses UTF-8 encoding, allowing Arabic text to be displayed correctly.

14 Saving ADSS Server Test Tool Requests

ADSS Test Tool provides the ability to save commands for later use

14.1 Save Command

Execute any request e.g. a signing request:

```
-service signing -server http://localhost:8777/adss/signing/hdsi -type pdf -
in data/signing/input/test_input_unsigned.pdf -out
data/signing/output/test_input_signed.pdf -client samples_test_client
```

After successful execution write the command below to save the command just used under any alias as defined by the user e.g.:

```
-saveCommand SigningRequest
```

This will save the last request under the alias **SigningRequest** at the following location:

<ADSS Server Test Tool Home directory>/conf/SigningRequest.ser

14.2 Show command

Users can list all the saved commands by executing the following:

```
-showCommands
```

The commands will be listed as:

```
alias :: SigningRequest

value :: -service signing -server http://localhost:8777/adss/signing/hdsi -
type pdf -in data/signing/input/test_input_unsigned.pdf -out
data/signing/output/test_input_signed.pdf -client samples_test_client
```

14.3 Load command

Users can load a specific saved command by using its alias as shown below:

```
-loadcommand SigningRequest
```

Pressing the enter key will load the saved command on client tool console as follows:

```
-service signing -server http://localhost:8777/adss/signing/hdsi -type pdf -
in data/signing/input/test_input_unsigned.pdf -out
data/signing/output/test_input_signed.pdf -client samples_test_client
```

Pressing enter key will execute this command.

*** End of document ***