PAdES baseline and PAdES additional profiles signatures Conformance Testing Tools Documentation

Version 1.0 June 2016

Abstract

This document provides a high level overview of the PAdES baseline signatures and PAdES additional signatures profiles conformance-testing tool. This document also highlights its most relevant functions. This tool has been deployed at the ETSI Portal on Electronic Signatures, and used by participants in the PAdES remote Plugtests™ event started on 4th May 2015, organized and supported by ETSI CTI (Centre for Testing and Interoperability).

ETSI (European Telecommunications Standards Institute)
06921 Sophia Antipolis CEDEX, France
Tel +33 4 92 94 42 00
info@etsi.org
www.etsi.org

Contents

1 Introduction

In answer to the European Commission Mandate 460 on Electronic Signatures Standardization, ETSI designed and developed a set of tools for automatically testing conformance of digital signatures and associated packages against the following technical specifications:

- EN 319 122 parts 1 and 2: CAdES core specification and baseline profile respectively
- EN 319 132 parts 1 and 2: XAdES core specification and baseline profile respectively
- EN 319 142 parts 2, 3, 4, and 7: PAdES basic, BES and EPES, LTV, and baseline profiles respectively
- EN 319 162 parts 1 and 2: ASiC core specification and baseline profile respectively

This document provides a high level overview of the PAdES baseline signatures and extended PAdES signatures conformance-testing tool in its final status, and also highlights its most relevant functions. This tool has been developed in Java language.

The tool is available at http://signatures-conformance-checker.etsi.org/pub/index.shtml.

2 References

[1] ETSI EN 319 142 Part 1: "Electronic Signatures and Infrastructures (ESI); PAdES digital signatures; Part 1: building blocks and PAdES baseline signatures"

[2] ETSI EN 319 142 Part 2: "Electronic Signatures and Infrastructures (ESI); PAdES digital signatures; Part 2: Additional PAdES signatures profiles"

[3] “CAdES baseline and extended CAdES signatures Conformance Testing Tools documentation”

[4] Bruce Eckel: “Thinking in Patterns”

3 PAdES conformance testing tool user guide

3.1 Running the tool

This clause shows how to use the PAdES conformance testing tool (PAdES conformance checker or PAdESCC will also be used hereinafter).

For running the tool the following command has to be called:

java –jar PAdESConformanceChecker –in <inputFile> -outFolder <outputFolder> -testSpec <referencePAdESSpecification>

Where

<inputFile> contains the pathname of the file containing the PAdES structure.

<outputFolder> is the pathname of the folder where the results will be generated. If the folder does not exist, the tool will try to generate it.

<referencePAdESSpecification> identifies the ETSI specification against which the PAdES structure has to be tested. At present the tool admits the following values:

"EN31914201v010100" for testing PAdES signatures against PAdES baseline signatures as specified within ETSI EN 319 142 Part 1 [1] v1.1.1.

"EN31914202v010100" for testing PAdES signatures against PAdES additional signatures profiles as specified within ETSI EN 319 142 Part 2 [2] v1.1.1.

3.2 Results generated

If the tool successfully finalizes its work it generates a framework of folders and html files within the folder whose pathname has been passed in <outputFolder> argument (output folder hereinafter in the present clause).

Figure 1 shows the contents of the output folder generated by the PAdES conformance-testing tool when a PDF file contains one PAdES signature.

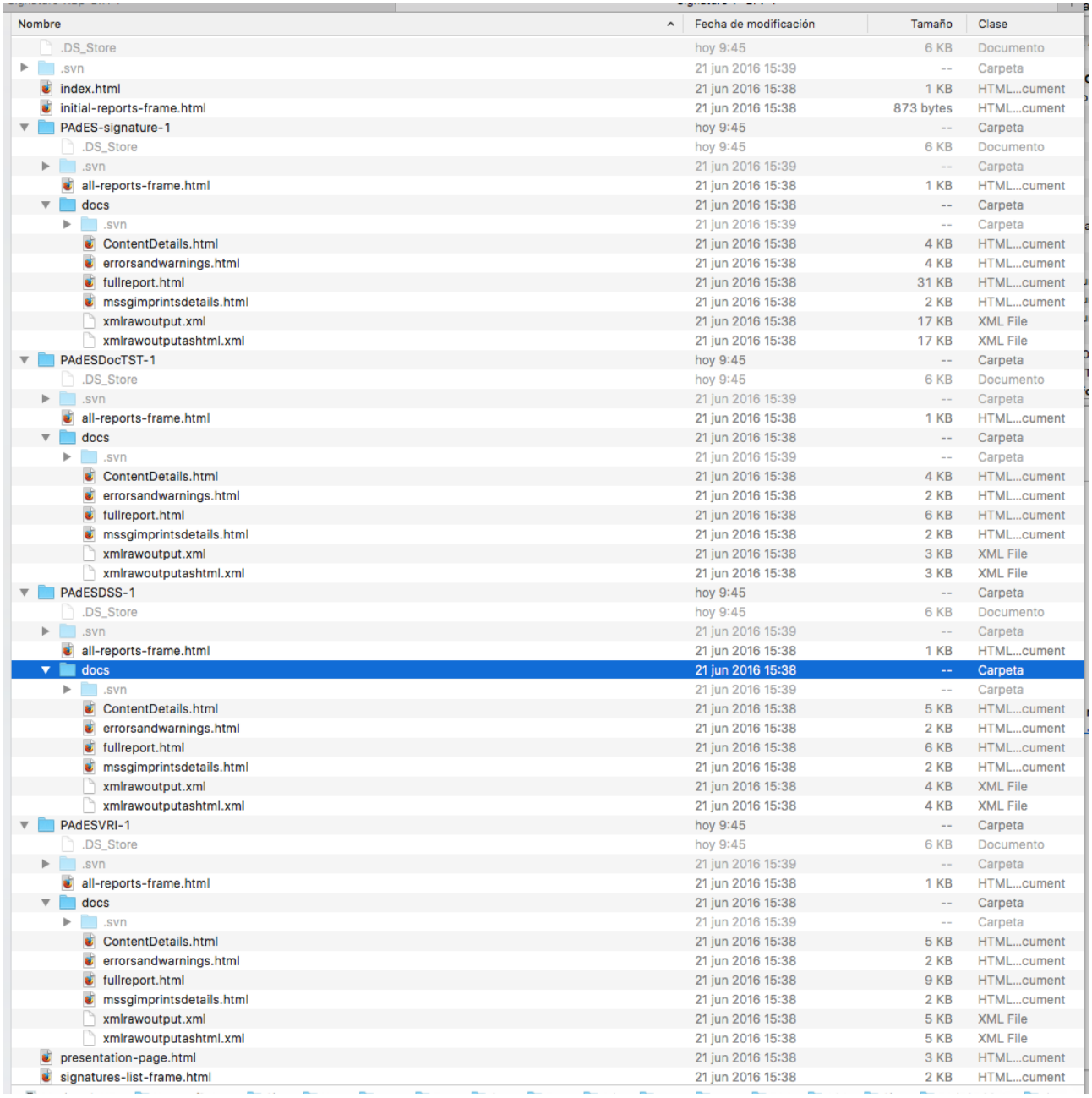


Figure 1: An example of contents of output folder generated by PAdES conformance testing tool

Within the output folder the following elements will appear:

1) **File "index.html"**: This is the page that the user of the tool needs to open in the web browser for inspecting the results. Figure 2 shows an example of the document after being opened with a web browser. It shows an html page with three frames.

   - The upper left frame (named "XML Input File Overview") contains a list of links to different groups of reports. Each report group corresponds to one of the main components incorporated into the PDF file as a result of the signing and/or signature's augmentation process.
   - The central frame, when the index.html is opened contains a presentation of the tool.
   - The lower left frame (named "Signature Reports view"), initially contains the text "When a signature is selected this frame will show links to its reports).

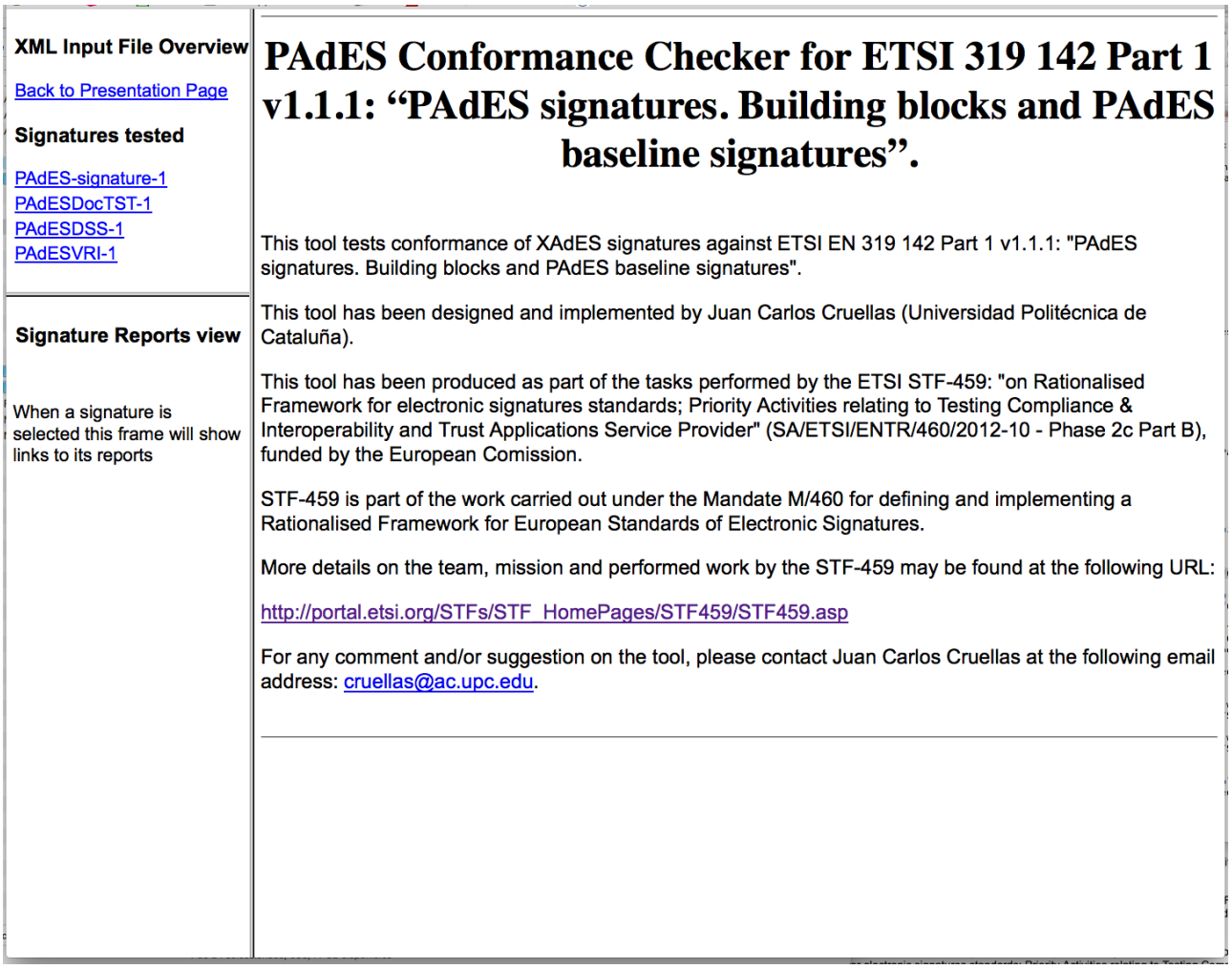


Figure 2: File "index.html" viewed in a web browser.

2) **File "signatures-list-frame.html"**: This is the file whose contents appear at the "XML Input File Overview" frame mentioned above. This frame includes a list of links to the results corresponding to each component incorporated into the PDF file as a result of the signing and/or signature's augmentation process. Figure 2 shows: one link to the set of reports corresponding to the signature itself (PAdES-Signature-1), one link to the set of reports corresponding to the DocumentTimeStamp already present within the PDF file (PAdESDocTST-1), one link to the set of reports corresponding to the DSS dictionary (PAdESDSS-1), and one link to the set of reports corresponding to the VRI dictionary (PAdESVRI-1).

3) **File "presentation-page.html"**: This is the file whose contents appear at central frame mentioned above.

4) **File "initial-reports-frame.html"**: This is the file whose contents appear at the "Signature Reports view" frame mentioned above. When one of the links found in the "XML Input File Overview" frame, pointing to the different PAdES components is selected, then this frame shows a list of links each one pointing to a specific

report. Each report shows different aspects of the results obtained by the PAdES conformance-testing tools as explained below. Figure 3 Shows an example of what is shown by the web browser after the link “PAdES-signature-1” is selected in the page shown in Figure 2.

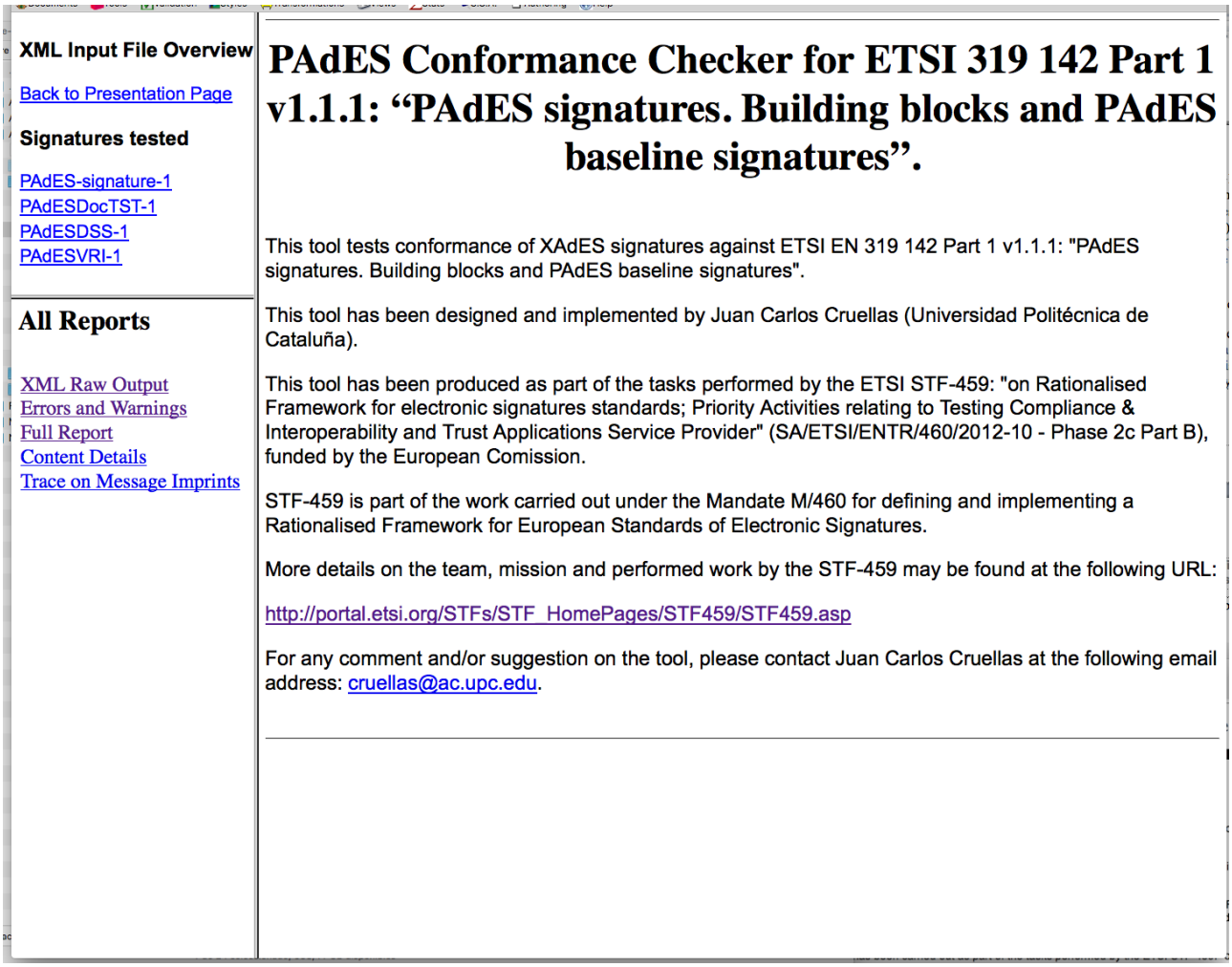


Figure 3: View of file "index.html" after selecting the likn “PAdES-Signature-1” in the "XML Input File Overview" frame.

5) **One or more folders with name "PAdES-signature-"<i>** (i has integer values starting in 1): One folder per each per `Signature` dictionary containing a CMS/CAdES signature. Each folder will contain the following elements:

- **File "all-reports-frame.html"**: This page contains links to the different reports generated by the tool. Its contents appear within the "Signature Reports view" frame once the user selects on of the links in the "XML Input File Overview" frame. More specifically it contains the following links: "XML Raw Output", "Full Report", "Errors and Warnings", "Content Details", and "Trace on Message Imprints". Each one is pointing to one of the documents that report on different aspects of the checks run. Contents of folder "docs " below provide detailed information on these reports.

- **Folder "docs"**: This folder contains the reports generated by the tool for the CAdES signature:

- File "xmlrawoutput.xml": This is the XML file that the default reporter generates.

- **File "xmlrawoutputashtml.xml":** This is the former file suitably modified for being correctly presented by most of the web browsers. This is the document presented by the tool when the user selects the "XML Raw Output" link in the "Signature Reports view" frame. Figure 4 shows an example of part of its contents for one CAdES signature. In this figure and in all the subsequent ones some of the data

identifying the generator of the signature or the issuer of a certificate have been occulted or changed for preserving their anonymity whenever it has been required.

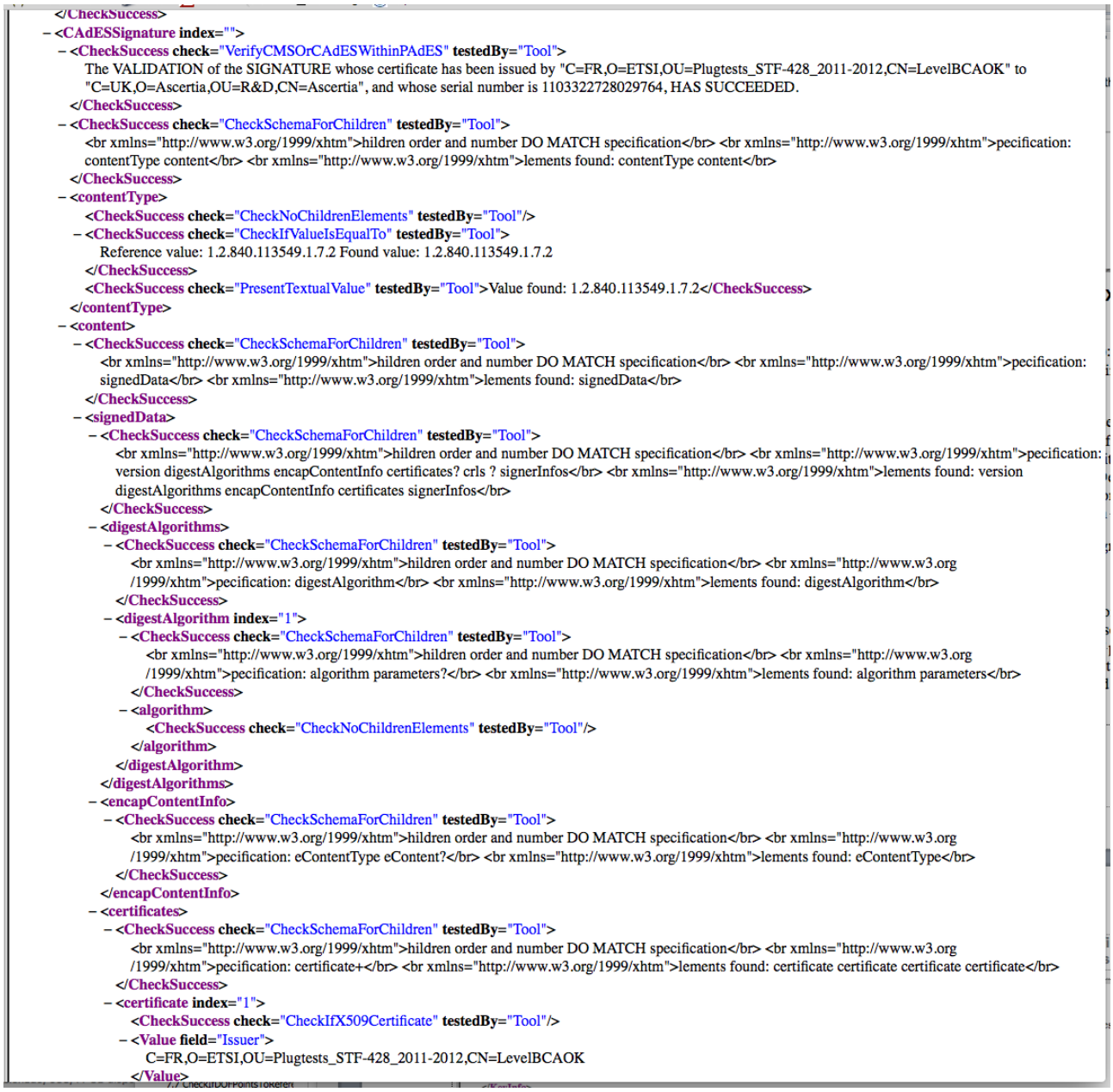
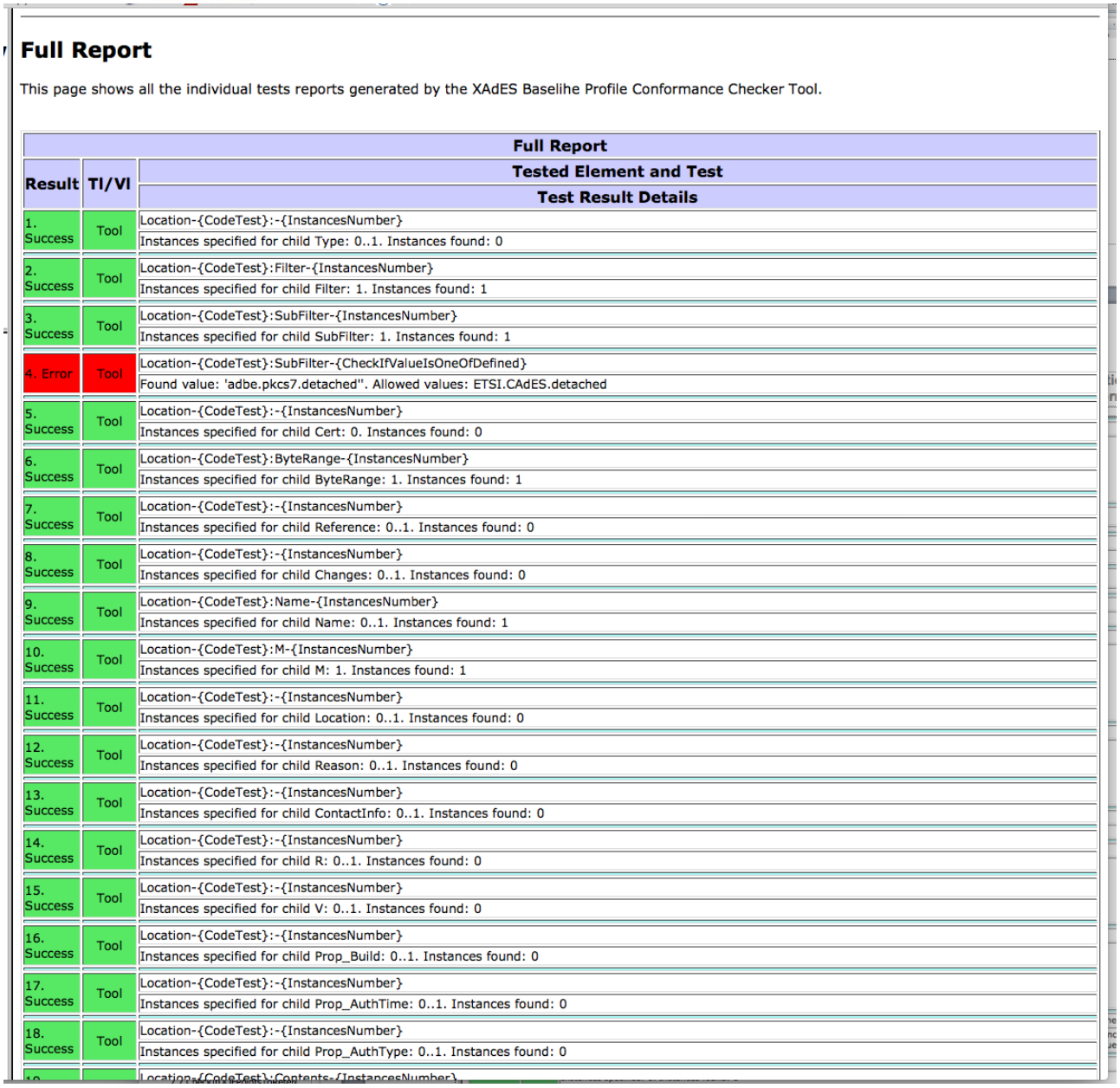


Figure 4: View of file "xmlrawoutputashtml.xml" corresponding to one CAdES signature.

- **File "fullreport.html"**: This is a document that reports on all the checks performed by the tool, indicating for each one if it has succeeded, failed or raised a warning; the component of the signature on which the check has been performed; the actual check performed and additional information whenever necessary. This is the document presented by the tool when the user selects the "Full Report" link in the "Signature Reports view" frame. Figure 5 shows an example of part of the contents of this file. Success reports have their first column cells coloured in green, failure reports have their first column cells coloured in red, warning reports have their first column cells coloured in yellow, and run exception reports have their first column cells coloured in blue. Each report includes an indication of the precise component on which the check has been performed and the code of the check performed. The reports also provide, whenever is necessary, additional details on the check performed.



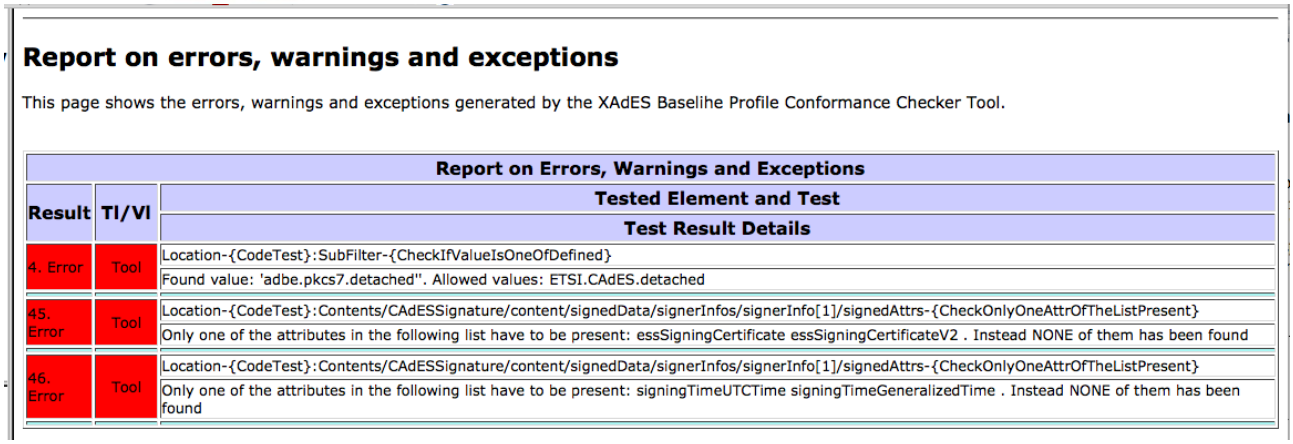
Full Report

This page shows all the individual tests reports generated by the XAdES Baselihe Profile Conformance Checker Tool.

Full Report		
Result	TI/VI	Tested Element and Test Test Result Details
1. Success	Tool	Location-{CodeTest}:-{InstancesNumber} Instances specified for child Type: 0..1. Instances found: 0
2. Success	Tool	Location-{CodeTest}:Filter-{InstancesNumber} Instances specified for child Filter: 1. Instances found: 1
3. Success	Tool	Location-{CodeTest}:SubFilter-{InstancesNumber} Instances specified for child SubFilter: 1. Instances found: 1
4. Error	Tool	Location-{CodeTest}:SubFilter-{CheckIfValueIsOneOfDefined} Found value: 'adbe.pkcs7.detached''. Allowed values: ETSI.CAdES.detached
5. Success	Tool	Location-{CodeTest}:-{InstancesNumber} Instances specified for child Cert: 0. Instances found: 0
6. Success	Tool	Location-{CodeTest}:ByteRange-{InstancesNumber} Instances specified for child ByteRange: 1. Instances found: 1
7. Success	Tool	Location-{CodeTest}:-{InstancesNumber} Instances specified for child Reference: 0..1. Instances found: 0
8. Success	Tool	Location-{CodeTest}:-{InstancesNumber} Instances specified for child Changes: 0..1. Instances found: 0
9. Success	Tool	Location-{CodeTest}:Name-{InstancesNumber} Instances specified for child Name: 0..1. Instances found: 1
10. Success	Tool	Location-{CodeTest}:M-{InstancesNumber} Instances specified for child M: 1. Instances found: 1
11. Success	Tool	Location-{CodeTest}:-{InstancesNumber} Instances specified for child Location: 0..1. Instances found: 0
12. Success	Tool	Location-{CodeTest}:-{InstancesNumber} Instances specified for child Reason: 0..1. Instances found: 0
13. Success	Tool	Location-{CodeTest}:-{InstancesNumber} Instances specified for child ContactInfo: 0..1. Instances found: 0
14. Success	Tool	Location-{CodeTest}:-{InstancesNumber} Instances specified for child R: 0..1. Instances found: 0
15. Success	Tool	Location-{CodeTest}:-{InstancesNumber} Instances specified for child V: 0..1. Instances found: 0
16. Success	Tool	Location-{CodeTest}:-{InstancesNumber} Instances specified for child Prop_Build: 0..1. Instances found: 0
17. Success	Tool	Location-{CodeTest}:-{InstancesNumber} Instances specified for child Prop_AuthTime: 0..1. Instances found: 0
18. Success	Tool	Location-{CodeTest}:-{InstancesNumber} Instances specified for child Prop_AuthType: 0..1. Instances found: 0
19.		Location-{CodeTest}:Contents-{InstancesNumber}.

Figure 5: View of file "fullreport.html" corresponding to one CAdES signature.

- **File "errorsandwarnings.html"**: This is a document similar to the former one except for the fact that it only reports the checks that have failed or raised a warning. This is the document presented by the tool when the user selects the "Errors and Warnings" link in the "Signature Reports view" frame. Figure 6 shows an example of the contents of this document.



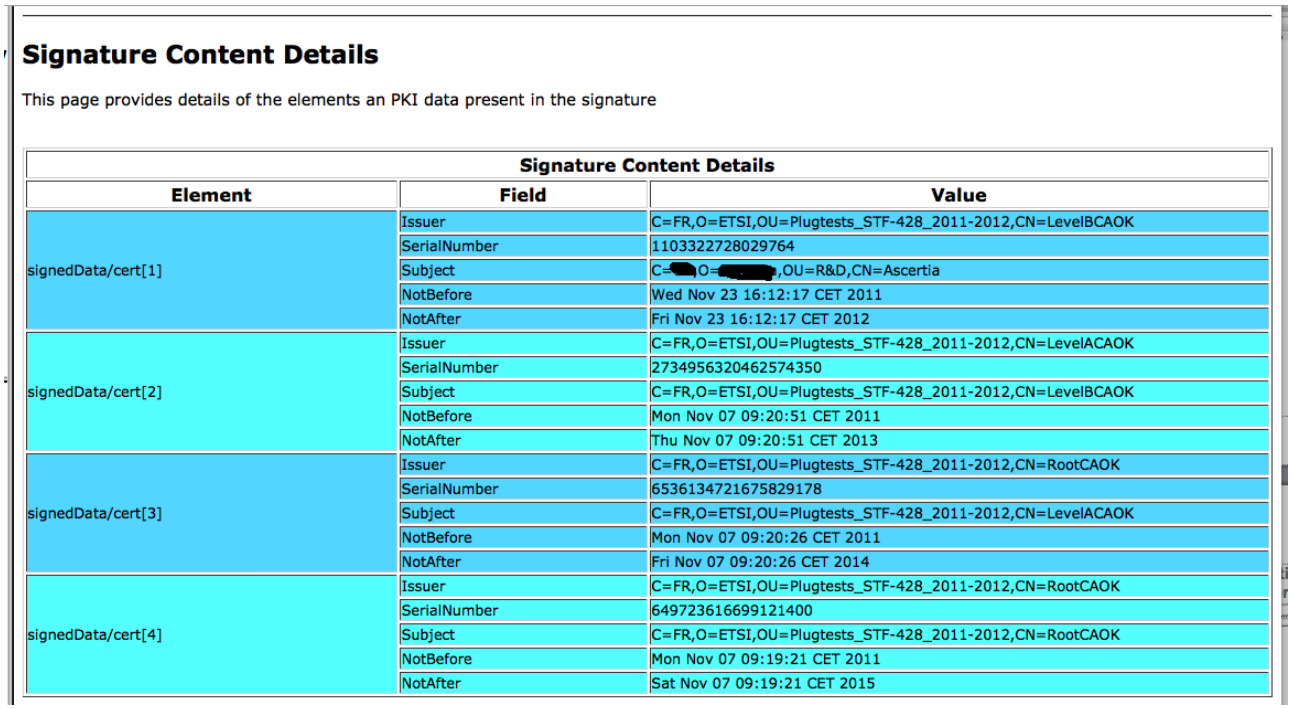
Report on errors, warnings and exceptions

This page shows the errors, warnings and exceptions generated by the XAdES Baselihe Profile Conformance Checker Tool.

Report on Errors, Warnings and Exceptions		
Result	TI/VI	Tested Element and Test Test Result Details
4. Error	Tool	Location-{CodeTest}:SubFilter-{CheckIfValueIsOneOfDefined} Found value: 'adbe.pkcs7.detached''. Allowed values: ETSI.CAdES.detached
45. Error	Tool	Location-{CodeTest}:Contents/CAdESSignature/content/signedData/signerInfos/signerInfo[1]/signedAttrs-{CheckOnlyOneAttrOfTheListPresent} Only one of the attributes in the following list have to be present: essSigningCertificate essSigningCertificateV2 . Instead NONE of them has been found
46. Error	Tool	Location-{CodeTest}:Contents/CAdESSignature/content/signedData/signerInfos/signerInfo[1]/signedAttrs-{CheckOnlyOneAttrOfTheListPresent} Only one of the attributes in the following list have to be present: signingTimeUTCTime signingTimeGeneralizedTime . Instead NONE of them has been found

Figure 6: View of file "errorsandwarnings.html" after selecting one of the CAdES signatures in the "XML Input File Overview" frame.

- **File "ContentDetails.html"**: This is a document that provides details of the PKI data found within the XAdES signatures, namely: X509 certificates, X509 attribute certificates, CRLs, OCSP responses, and time-stamp tokens. This is the document presented by the tool when the user selects the "Content Details" link in the "Signature Reports view" frame. Figure 7 shows an example of the appearance of this file in a web browser. Each type of PKI data have a different colour: details of X509 certificates and attribute certificates are shown in blue rows, time-stamp tokens are shown in pink rows, and details of revocation data (CRLs and OCSP responses are shown in green rows).



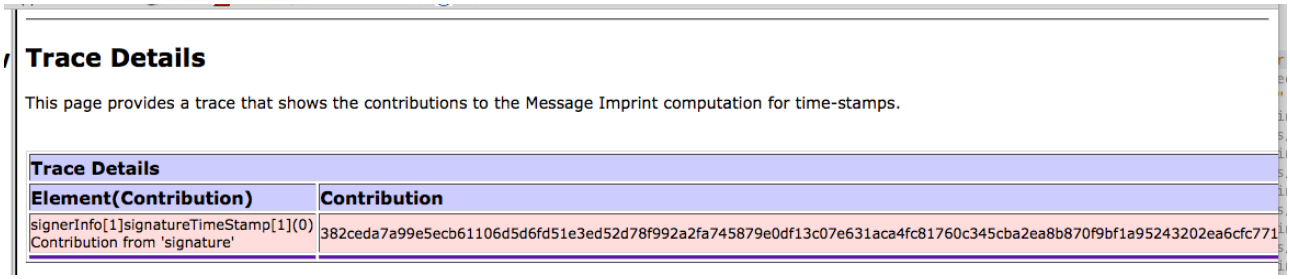
Signature Content Details

This page provides details of the elements an PKI data present in the signature

Signature Content Details		
Element	Field	Value
signedData/cert[1]	Issuer	C=FR,O=ETSI,OU=Plugtests_STF-428_2011-2012,CN=LevelBCAOK
	SerialNumber	1103322728029764
	Subject	C=[redacted],OU=R&D,CN=Ascertia
	NotBefore	Wed Nov 23 16:12:17 CET 2011
	NotAfter	Fri Nov 23 16:12:17 CET 2012
signedData/cert[2]	Issuer	C=FR,O=ETSI,OU=Plugtests_STF-428_2011-2012,CN=LevelACAOK
	SerialNumber	2734956320462574350
	Subject	C=FR,O=ETSI,OU=Plugtests_STF-428_2011-2012,CN=LevelBCAOK
	NotBefore	Mon Nov 07 09:20:51 CET 2011
	NotAfter	Thu Nov 07 09:20:51 CET 2013
signedData/cert[3]	Issuer	C=FR,O=ETSI,OU=Plugtests_STF-428_2011-2012,CN=RootCAOK
	SerialNumber	6536134721675829178
	Subject	C=FR,O=ETSI,OU=Plugtests_STF-428_2011-2012,CN=LevelACAOK
	NotBefore	Mon Nov 07 09:20:26 CET 2011
	NotAfter	Fri Nov 07 09:20:26 CET 2014
signedData/cert[4]	Issuer	C=FR,O=ETSI,OU=Plugtests_STF-428_2011-2012,CN=RootCAOK
	SerialNumber	649723616699121400
	Subject	C=FR,O=ETSI,OU=Plugtests_STF-428_2011-2012,CN=RootCAOK
	NotBefore	Mon Nov 07 09:19:21 CET 2011
	NotAfter	Sat Nov 07 09:19:21 CET 2015

Figure 7: View of file "ContentDetails.html" corresponding to one CAdES signature.

- **File "mssgimprintsdetails.html"**: This is a document that provides, for each time-stamp token found in the PAdES signature, the trace of every component of the signature that is concatenated for obtaining the input to the computation of the message imprint that should have been sent to the Time Stamp Authority. This page is normally used by implementers for debugging their own tools if the tool reports any problem with the message imprint verification of any time-stamp token, which has proved to be a relevant interoperability issue. In the case of the CMS/CAdES signature that is encapsulated within the Signature PDF dictionary, this page presents this trace only for signature-time-stamp attribute because no other attributes encapsulating time-stamp tokens are allowed.



Trace Details

This page provides a trace that shows the contributions to the Message Imprint computation for time-stamps.

Trace Details	
Element(Contribution)	**Contribution**
signerInfo[1]signatureTimeStamp[1](0) Contribution from 'signature'	382ceda7a99e5ecb61106d5d6fd51e3ed52d78f992a2fa745879e0df13c07e631aca4fc81760c345cba2ea8b870f9bf1a95243202ea6cfc771

Figure 8: View of file "mssgimprintsdetails.html" corresponding to the time-stamp token encapsulated within signature-time-stamp unsigned attribute found within the CAdES signature in the PAdES signature dictionary.

6) **One or more folders with name "PAdESDocTST-"<i>** (i has integer values starting in 1): One folder per each per `DocumentTimeStamp` dictionary containing a time-stamp token. Each folder will contain the same elements as in 5). The main difference is that the "mssgimprintsdetails.html" document is empty: the time-stamp token within this dictionary time-stamps the contents present in the PDF at the moment the dictionary is created, plus the contents of the dictionary itself except the bytes that include the time-stamp token itself.

7) **One folder PAdESDSS-1**: This folder contains reports corresponding to the DSS dictionary if present. Its contents are the same as in 6). Again the "mssgimprintsdetails.html" document is empty.

8) **One folder PAdESVRIS-1**: This folder contains reports corresponding to the VRI dictionary if present. Its contents are the same as in 6) and 7). Again the "mssgimprintsdetails.html" document is empty.

4 Contents of the reports generated by PAdESCC.

4.1 Introduction

As mentioned in clause 3.2 the content of folders **PAdES-Signature-<i>** will be a set of reports generated after checking conformance of a CMS/CAdES signature. The details of the contents of these reports can be found in the document "CAdES baseline and extended CAdES signatures Conformance Testing Tools documentation" [3], and will not be repeated here.

In consequence the present document contains details on the contents of the reports generated by the PAdES conformance-testing tool after checking DocumentTimeStamp, DSS and VRI dictionaries.

4.2 Full report contents

4.2.1 Introduction

This clause provides a detailed explanation of the different components of the full report that may be generated by any of the conformance-testing tools.

4.2.2 Structure of the table

The full report provides information of the tests performed by the conformance-testing tool in a table. The headers of the table are as indicated below.

<table>
<tr><td colspan="3">Full Report</td></tr>
<tr><td rowspan="2">Result</td><td rowspan="2">TI/VI</td><td>Tested Element and Test</td></tr>
<tr><td>Test Result details</td></tr>
</table>

The cell in the first column indicates the result of the specific test. Values allowed are:

- "Success". Indicates that the check has succeeded. In this case the background colour of this cell and the cell in the second column is green.
- "Failure". This value appears when the check does not succeed and in consequence the signature/container is not conformant against the reference specification. In this case the background colour of this cell and the cell in the second column is red.
- "Warning". In this case the background colour of this cell and the cell in the second column is yellow.
- "RunException". It appears for indicating that some exceptional situation occurs while the tool is being executed. In this case the background colour of this cell and the cell in the second column is dark blue.

The third column is divided in two rows. The content of the first row follows the pattern

Location-{CodeTest}: [LOCATION]-{[TEST CODE]}.

It provides information on two aspects of the tests:

- The LOCATION of the test is the component of the signature/container on which the test has been performed. The location is indicated by a path name resulting of concatenating the names of the different components that must be visited in the signature tree structure for going from the root element of the signature to the checked component. Components of the path name are separated by "/". When there are several sibling components with the same name, then integer indexes are used.

Example:
Location{CodeTest}:Contents/CAdESSignature/content/signedData/signerInfos/signerInfo[1]/signedAttrs/attribute[3]/attrValues/essSigningCertificateV2[1], indicates that the tested component is the first essSigningCertificateV2 attribute found by the tool (as the value of an attribute is a SET OF "first" means here the one that has been decoded first, not the first of a sequence as there is not such a sequence).

- The TEST CODE of the test is the unique identifier of the test performed on the component. The identifier is a name that tries to capture the essence of the semantics of the test so that users can easily understand what has been tested.

The content of the second row of the cell (additional information row hereinafter) in the third column is optional. It provides complementary information to the test performed, whenever is necessary.

The rest of the clause provides detailed explanations of the tests performed. This is done by providing details of the contents of the two rows of the cells in third column.

4.2.3 Contents common to all the conformance-testing tools

See "CAdES baseline and extended CAdES signatures Conformance Testing Tools documentation" [3] clause 4.2.3 for a detailed explanation of these contents.

4.2.4 Contents specific to PAdES conformance testing

4.2.4.1 Introduction

The contents that are specific to the reports generated by PAdES conformance-testing tool will be the ones detailed in "CAdES baseline and extended CAdES signatures Conformance Testing Tools documentation" [3] clause 4.2.4 and the contents detailed in clause 4.2.4.2 of the present document.

4.2.4.2 MatchReferencedSignature test

This component indicates the result of a test that tries to match the key of one of the VRI entries within the DSS dictionary. This is a component of the report corresponding to the conformance checking of VRI dictionaries.

EXAMPLES:

Location-{CodeTest}:VRI[4]-{MatchReferencedSignature}
This VRI dictionary corresponds to the signature that is within the signature whose signing certificate has the details that follow: Issuer:"C=FR,O=ETSI,OU=Plugtests_STF-428_2011-2012,CN=LevelBCAOK"; SerialNumber: 209924606203005; SubjectName: "C=CC,O=XXXX,CN=YYYY"

Location-{CodeTest}:VRI[1]-{MatchReferencedSignature}
This VRI dictionary corresponds to the signature that is within the X509 CRL whose details are as follows: CRL; version: 2, issuer: CN=LevelACAOK, OU=Plugtests_STF-428_2011-2012, O=ETSI, C=FR, thisUpdate: Thu Nov 24 11:54:53 CET 2011, nextUpdate: Sat Dec 24 11:54:53 CET 2011

5 High level overview of the conformance-testing tools software components

This clause provides a high-level overview of the main software components that are common to the conformance-testing tools.

5.1 Basic conformance-testing tools' conceptual model

One of the core concepts in the conceptual model of the conformance-testing tools is the **Component**. Each instance of this concept represents an individual component of the AdES signature under test. Depending on the AdES signature's format, there may be one or more types of components: for instance, XAdES signatures have only fields, whereas XAdES signatures, being XML based signatures, have elements and attributes. In PAdES signatures, the components may be fields of XAdES or CMS signatures present within the "Signature" dictionaries, or fields of PDF dictionaries. The Component class implements the **Composite** software design pattern.

Another core concept is the **Command**. Each instance of this concept represents most of the times one action to be performed by tool on a certain component of the AdES signature. There are commands for performing a certain check on one component of the signature, commands that in addition to perform this check perform an additional action (like storing a certain result in a variable that may be accessed by other command), commands invoked for navigating throughout the AdES signature's tree structure, commands for selecting a set of components of the signature, or for traverse a list of children components of a certain component, etc. The present document provides details of all the commands used by the PAdES conformance-testing tool. These Command objects are built by parsing a XML file, called **XML driving instructions file** hereinafter. Each conformance-testing tool uses at least one. Each XML element in these files (except a few of them, which identify different areas within the files) represent one command to be executed by the conformance-testing tools.

The third core concept is the **Interpreter**. The interpreter creates a sequence of command objects, as a result of parsing the XML driving instructions file whose components represent the commands to be executed by the tool for conducting the signature conformance testing process. Commands may be grouped in named **Commands Group**, a sequence of commands that has a name. This name allows to calling its execution from any point of the XML driving instructions file using the **ExecuteCommandsGroup** command. The named Commands Group are in fact a very basic implementation of functions. The Interpreter object uses **Factory Method design pattern** as implemented in "Thinking in Patterns" by Bruce Eckel [4], for creating the different Command objects.

The AdES conformance-testing tools incorporate also proxies of the AdES signature under test, **AdESSignatureProxy**, so that it provides a unique interface that may be used within the source code of the commands regardless of the specific AdES signature format.

While performing the tasks indicated by the commands in the sequence generated by the interpreter, the conformance-testing tool maintains information of the **Context**, including, among other things, the proxy of the signature under test, the reporter, and one cursor to the signature's component under test. There are three types of Context objects:

1) **AdESCCFileInputContext**. The object instance of this type is created as soon as the input file is parsed, and it is kept until the end of the conformance-testing tool. It is the context object that contains information related to the status in the processing of the whole input file. An input file can be a file containing one or more XAdES signatures, a file containing one or more parallel CAdES signatures within a single CAdES structure, a PDF file containing one or more PAdES signatures and/or DocumentTimeStamps, and an ASiC container containing one or more signature files, one or more detached signed files, manifest files, etc. The object instance of AdESCCFileInputContext includes a map of global variables whose keys are the names of the variables. These variables are fully accessible at any stage of the conformance-checking process.

2) **AdESCCFileWithSignaturesContext**. The object instance of this type is created as soon as one file with one or more signatures is parsed. When testing (C/P/X)AdES signatures, the input file and the file with signatures are the same file. However, when testing an ASiC container, the input file is the ASiC container itself, while the file with signatures is one of the files encapsulated within the container itself. This context object contains information related to the status in the processing of the file that encloses one or more signatures. It is kept until the processing of all the signatures present in the file is finalized.

3) **AdESCCSignatureContext**. The object instance of this type is created as soon as the processing of one signature is started, and it is kept until this processing is finalized. It contains status information related to the processing of one signature. It contains a map of "local" variables whose keys are the names of these variables. The conformance-testing tools incorporate commands for moving global variables to the map of "local" variables and vice versa. This allows to keep memory of relevant facts throughout the processing of an input file.

The conformance-testing tools also incorporate a **CheckReporter**, which reports the results of the checks performed, as a XML document: the **XMLRawReport**. The check reporter object is an object that is kept in the instance of AdESCCSignatureContext. Each signature tested requires a new check reporter object because each report generated by the conformance-testing tools correspond to one signature.

The conformance-testing tool also includes the **XSLTReporter**, in charge of applying different XSLT transformations to the XML raw report and generating HTML pages. Each HTML presents, in tabular form, different aspects of the results generated by the tool while performing the conformance-testing process. Clause 2.3 provides more details of the final output generated by the implemented XSLTReporter.

Each conformance-testing tool will require its own specific type of context objects, signatures proxies, signature components, file parsers, etc. The conformance-testing tools use the Abstract Factory design method for creating their own set of objects.

Figure 9 below shows the basic conceptual model that groups the aforementioned concepts.

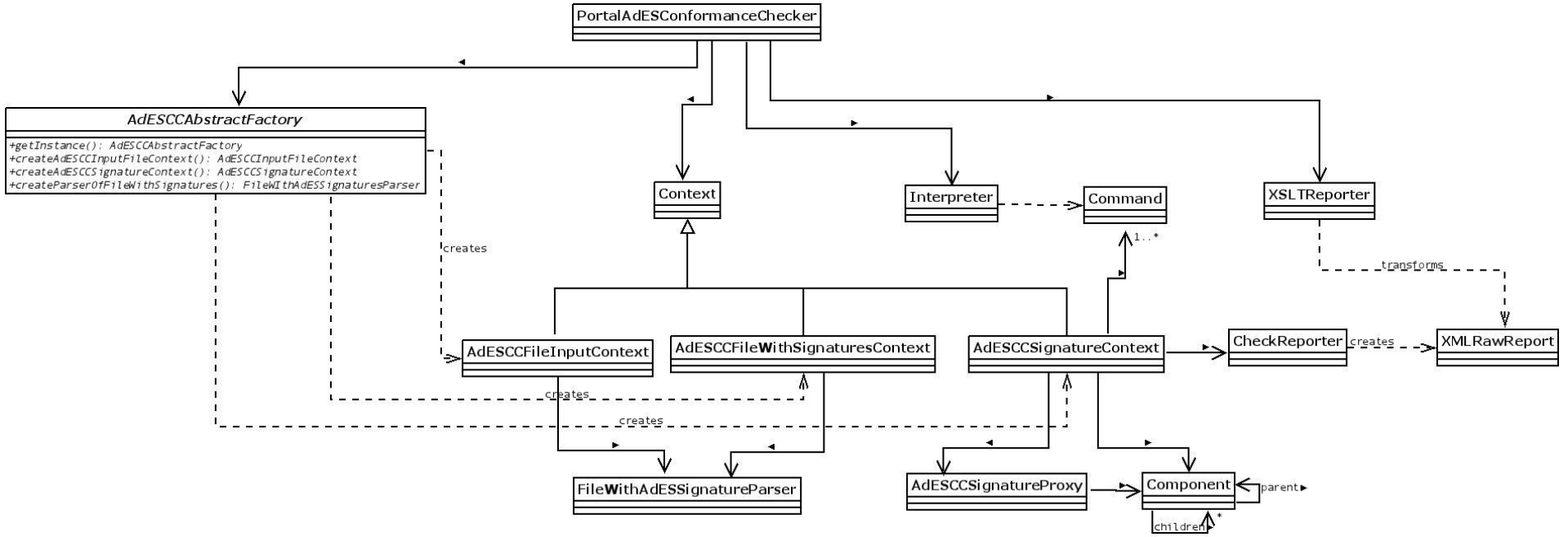


Figure 9: Basic conceptual model of the conformance-testing tools

The instance of PortalAdESConformanceChecker is responsible of coordinating the creation and work of the rest of the objects so that each one properly cooperate with others and jointly performs the suitable set of conformance checks on each component of the AdES signature under test.

In term of design, the conformance-testing tools design include design patterns wherever adequate: factory method, abstract factory, proxy, observer, strategy, command, decorator, are some of them.

5.2 Commands and XML driving instructions file

The AdES conformance-testing tools operation is driven by an input XML document within a file whose elements list the set of commands that the tools have to execute: the XML driving instructions file.

The suite of commands defined include two basic types: those commands that are generic, i.e. apply to more than one AdES signature format, and those ones that are specific, i.e. apply only to a certain AdES format.

Among the generic commands, there are also different types. Below follow mentions to some of them:

- Commands for performing structural checks: these are commands that order the tool to check that the children of the component under test are conformant with the syntactical specification of this component within the corresponding AdES signature format specification, i.e., that each child appears in the right position within the list of children and that the number of instances of that children is the expected number.

- Commands for controlling the navigation of the tool throughout the AdES signature. Examples of this type are, the command that orders to move the cursor from the current component to a certain named child, or the command that orders to iterate on each of the children of the AdES signature component pointed by the cursor, or the command that orders to iterate only on children that have a certain name.

- Commands for doing checks on specific components. Examples of this type includes commands for checking that the textual value of the component pointed by the cursor is equal to an expected value or another for checking that the textual value of the component pointed by the cursor is one among a set of textual values. Other example is the command for checking that the value of a certain component is a URI reference.

- Commands for doing checks on specific components and additional operations. Examples of this type include commands that check if the content of a certain component is a certain PKI Data (a X509 certificate, for instance), and store that data in a certain container variable (map or list) for being used afterwards by another command.

- The semantics of the commands define a kind of high-level programming language that allows to define the whole set of checks to be performed on a certain AdES signature in the XML driving instructions file.

As for the specific commands, i.e. those ones that only apply to one type of AdES signature format, they are directly related with particular requirements defined by the technical specification corresponding to that format.

It is worth to mention that the usage of the Command pattern and the XML driving instructions file, facilitates quicker adaptation of the conformance-testing tool to any change performed in the AdES signature format (or ASiC container) technical specifications. If a new requirement is included in one already existing component of a certain AdES signature format, like adding a new value to the potential repertoire of values that that component may take, no change should be implemented in the code source: instead the XML driving instructions file should be changed by adding this new value in the corresponding command. If, instead a new component is defined (like for instance seems likely to happen in the new ENs standardizing XAdES and CAdES), a part of checks that must be done on them very likely are actually performed by already existing concrete commands, so that no source code has to be added to the tool, but instead these commands have to be included in the set of commands within the XML driving instructions file. New concrete command classes will be required only when none of commands already developed, are able to perform a certain check required for these new components. This means that the conformance-testing tools are easily extensible by combining the development of new concrete command classes and the modification of the XML driving instructions file. The usage of the factory method in the Command hierarchy, as designed by Bruce Eckel in its “thinking in patterns” draft book, minimizes the amount of code to be added to the tool, as this design implies that the code for creating an instance of the new concrete command appears within the concrete command class itself.

It is also worth to mention that changes in the reference specifications can be quickly incorporated by changing the contents of this file and incorporating new Command classes if required. This also allows that the same tool can be used for checking signatures against different versions of the same specification, as each conformance testing process can be driven by the specific XML driving instructions file corresponding to one specific version.

5.3 Overview of the operation of the testing tools

All the conformance-testing tools have the XML driving instructions file, and the AdES signature to be tested as part of the inputs.

The Interpreter object parses the XML driving instructions file, creating the corresponding concrete commands and building the sequence of commands to be executed by the conformance-testing tool. In addition to that, the tool creates the AdES signature proxy after parsing the AdES signature to be tested, and creates the SignatureContext object ready for starting the execution of the sequence of commands.

After that, the tool iterates on the sequence of concrete command objects and executes them by invoking their execute() method. While the execute() method within a certain concrete command is executed, its code generate reports as XML elements, which are incorporated within the XMLRawReport. These XML elements may:

- Report that a certain check has been successfully performed on the AdES signature component pointed by the cursor, and provide additional information for the reader of the report (for instance, for checks on the message imprint within a certain time-stamp token, this XML element includes the value of the message imprint computed by the tool, which is equal to the message imprint value found within the time-stamp token).

- Report that a certain check has been performed on the AdES signature component pointed by the cursor, but the check has failed (i.e. the signature is not conformant against the corresponding technical specification). In such circumstances, the generated XML element includes detailed explanation of the reasons of the failure (for instance, for checks on the message imprint within a certain time-stamp token, this XML element includes both the value of the message imprint computed by the tool, and the message imprint value found within the time-stamp token).

- Report that a certain unexpected exception has occurred during the execution of a certain command. These exceptions may be generated by a number of different reasons: badly formed XML element within the XML driving instructions file, a bug within the conformance-testing tool, etc. Whenever one of these exceptions is reported, the developer of the conformance-testing tools should be notified and provided with the corresponding details.

- Include specific values of the AdES signature components for presenting them to the users in a separate report so that users may obtain readable information of certain core components of the tested AdES signatures. Examples of core components for which this kind of elements are generated are the PKI tokens present within the signatures, namely: X509 certificates, X509 attribute certificates, CRLs, OCSP responses, and time-stamp tokens.

When all the sequence of commands has been iterated, the XMLRawReport is completed. This report includes one XML element for each component of the AdES signature tested. Each element of this type contains all the reports corresponding to the checks performed on that specific component, and optionally a set of values of that component to be presented to the user.

The last task performed by conformance-testing tool is the generation of the final output for the user. See clause 4.2 for an explanation of these reports, and the examples attached to this report in the delivered package. This final output includes a framework of the 5 html files mentioned in clause 3.2 of the present document: the html file **representing the contents of the XMLRawReport report itself, the full report, the report on errors and warnings, the report on signature content details, and the report presenting a trace of the message imprint** computation for time-stamp tokens incorporated into the signature.

The conformance-testing tool generates each HTML file by applying a specific XSLT transformation designed for such purpose, to the XMLRawReport. The XSLTReporter object is in charge of applying these XSLT transformations to the XMLRawReport. Attached to this document, a couple of examples of the output HTML documents framework are delivered. These reports corresponds to XAdES signatures generated by participants in the XAdES interoperability

remote plugtest™ started on 1st October 2015. The details table is manipulated in order to preserve the anonymity of the generators of the signatures, according to the NDA signed by the participants in the interoperability event.

6 Specification of commands that may also be used by other conformance-testing tools

6.1 Introduction

The commands that are processed by the PAdES conformance-testing tool are the commands detailed in the other sub-clauses of clause 6 of the present document, and the commands used by the CAdES conformance-testing tool whose details may be found in the document "CAdES baseline and extended CAdES signatures Conformance Testing Tools documentation" [3], clauses 7 and 8.

6.2 CopyGlobalVarToContextVar

This command copies a global variable into a local variable so that it may be used by commands in the signature (or dictionary in the case of PAdES signatures) context.

Command arguments:

'varName' contains the name of the global variable. The local variable will have the same name.

6.3 CopyContextVarToGlobalVar

This command copies a local variable into a global variable so that it may be passed from one signature (or dictionary in the case of PAdES signatures) context to the other. The usage of this command and the former one allows passing variables from one context to the other as the conformance-testing tool switches from context to context.

Command arguments:

'varName' contains the name of the local variable. The global variable will have the same name.

7 Specification of commands that are applicable only for PAdES conformance testing

7.1 Introduction

This clause provides details on commands that are used only by PAdES conformance-testing tool.

7.2 HexUpCaseSHA1ToVariable

This command is executed when checking one component of a PDF dictionary whose value is hex-encoded. This command first gets the bytes of the dictionary component, then computes its SHA1 digest value and finally it encodes this digest value in hexadecimal in upper cases. This command is used for computing the upper cases hexadecimal encoding of the SHA1 digest value of the different signatures present within the PAdES signature under test in case the DSS dictionary has VRI entries, as their keys are precisely the upper cases hexadecimal encoding of the SHA1 digest value of the signature each VRI entry corresponds to.

Command arguments:

'resultTo' output argument that contains the name of a String local where the upper case hexadecimal encoding of the SHA1 digest value will be stored.

7.3 SigFieldHexUpCaseSHA1ToVariable

This command gets the upper case hexadecimal encoded SHA1 digest of the signature field of X509 CRLs and OCSP responses computed as specified in ETSI EN 319 142 Part 1 [1] clause 5.4.2.3. This command is invoked only if the PDF signed document contains VRI dictionary entries.

Command arguments:

'resultTo' output argument that contains the name of a String local that the command will create and where it will store the upper case hexadecimal encoding of the SHA1 digest value.

7.4 PKIMapSigSHA1ToDetails

This command takes the upper case hexadecimal encoded SHA1 digest of a signature, generates a String containing details that allow to identify the signature and adds a new entry in a map, whose keys are the upper case hexadecimal encoded SHA1 digest of signatures, and whose values are Strings containing details that allow to identify the aforementioned signatures.

Command arguments:

'file' identifies the name the XML driving instructions file that contains the commands required for checking conformance of the CAdES signature present within the PDF signature dictionary.

7.5 ExecuteCommandsForCAdESSignature

This command triggers the conformance checking of the CAdES signature present within a PDF signature dictionary. This command is invoked as part of the conformance checking of the PDF signature dictionary. This command retrieves the XML driving instructions file containing the commands required for checking conformance of the CAdES signature, change the signature context accordingly, and executes all the commands required by the retrieved XML driving instrucations file.

Command arguments:

'hexSHA1In' contains the name of the variable that contains the upper case hexadecimal encoding of SHA1 digest value of the signature.

'resultTo' contains the name of the variable that holds the Map where the command adds the new entry.

7.6 IterateOnAllChildrenAndChangeAnotationsTo

This command is executed when checking conformance of VRI entries. This command iterates on each VRI entry present within the PDF document, and changes its key.

VRI entries have as key the hexadecimal encoding of SHA1 digest value of signatures present elsewhere. PAdES conformance-testing tool considers the key of a PDF dictionary its name. In order to iterate on all the VRI entries within a PAdES signature, it is required to iterate on every child regardless its name (key). As the final reports indicate the component on which a certain check has been performed using its name, it is also worth to change this indication by a suitable name different than an hexadecimal encoding of a SHA1 digest value.

Command arguments:

'xmlAnotationName' identifies the indication that has to appear in the PAdES conformance-testing tool. In the current implementation the value of this argument is “VRI”.

7.7 MatchReferencedSignature

This command is executed when checking one VRI entry. This command takes the key of the VRI entry and checks if this value is present in a Map whose keys are upper case hexadecimal encoding of SHA1 digest of signatures, and whose values are strings that provide details of the signatures whose encoded SHA1 digest are the aforementioned keys.

The entries have been previously added to the map (see clause 7.4) whenever the PAdES conformance-testing tool has visited and checked either a CAdES signature within a PDF signature dictionary, a X509 CRL, or an OCSP response.

Command arguments:

*' **input'*** identifies a local variable that is a map. The keys of this map are upper case hexadecimal encoding of SHA1 digest of signatures, and the values are strings that provide details of the signatures whose encoded SHA1 digest are the aforementioned keys.