CAdES baseline and extended CAdES signatures Conformance Testing Tools Documentation

Version 1.0 June 2016

Abstract

This document provides a high level overview of the CAdES baseline signatures and extended CAdES signatures 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 CAdES remote Plugtests™ event started on June 11th 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 CAdES baseline signatures and extended CAdES signatures conformance-testing, 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 319 122 Part 1: "Electronic Signatures and Infrastructures (ESI); CAdES digital signatures; Part 1: building blocks and CAdES baseline signatures"

[2] ETSI EN 319 122 Part 2: "Electronic Signatures and Infrastructures (ESI); CAdES digital signatures; Part 2: extended CAdES signatures"

[3] Bruce Eckel: “Thinking in Patterns”

3 CAdES conformance testing tool user guide

3.1 Running the tool

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

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

java –jar CAdESConformanceChecker –in <inputFile> -outFolder <outputFolder> -testSpec <referenceCAdESSpecification>

Where

<inputFile> contains the pathname of the file containing the CAdES 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.

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

"EN31912201v010100" for testing CAdES signatures against CAdES baseline signatures as specified within ETSI EN 319 122 Part 1 [1] v1.1.1.

"EN31912202v010100" for testing CAdES signatures against extended CAdES baseline signatures as specified within ETSI EN 319 122 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 CAdES conformance testing tool when the CAdES structure contains one CAdES signature.

Figure 1: An example of contents of output folder generated by CAdES 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 parallel CAdES signatures (one per each instance of SignerInfo type) that a CAdES structure may contain. If only one CAdES signature is present, only one link is shown.

   - 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).

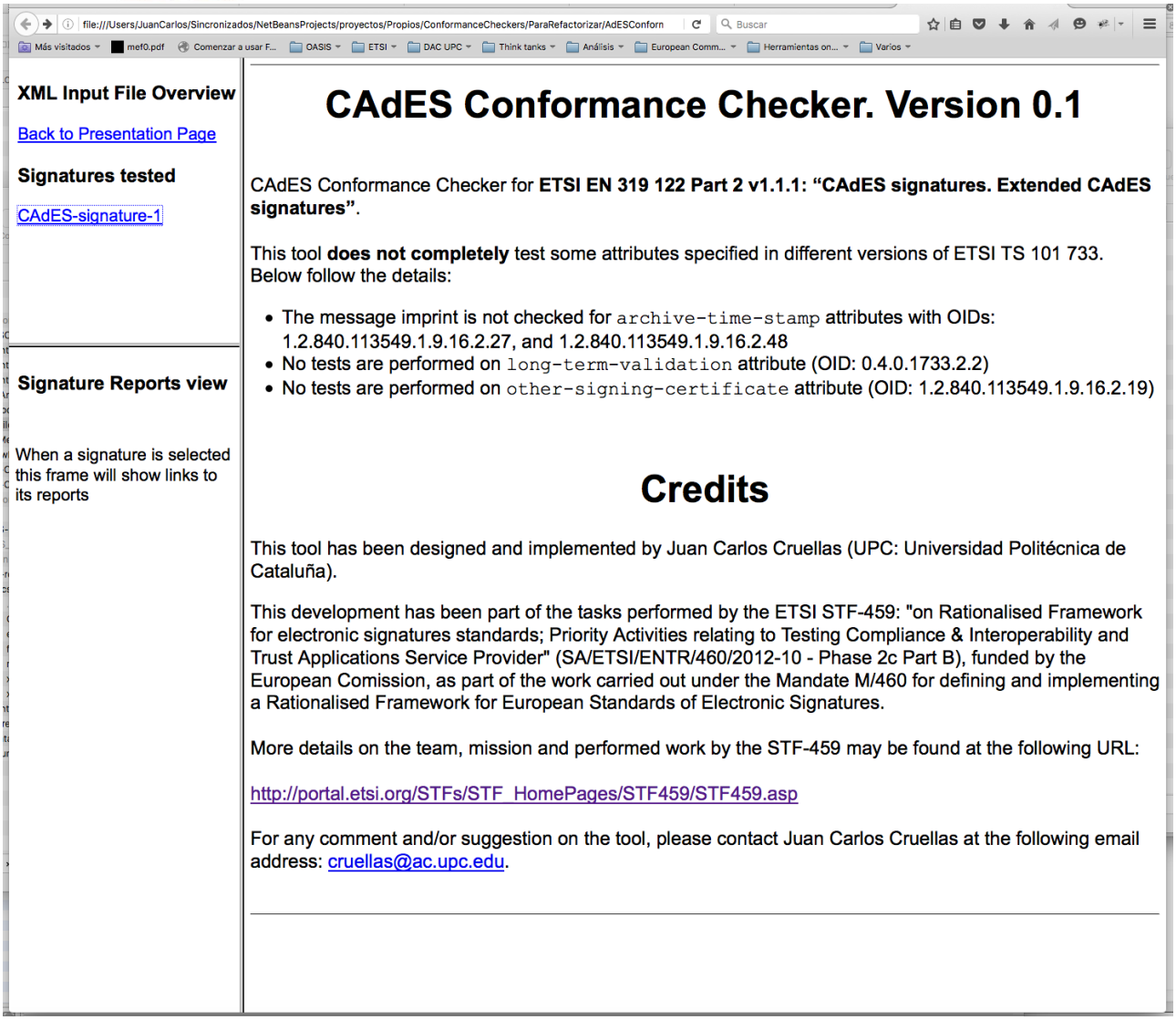


Figure 2: File "index.html" viewed in a web browser.

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 different CAdES signatures found within the CAdES structure.

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 CAdES signatures within the CAdES structure 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 CAdES conformance-testing tools as explained below. Figure 3 Shows an example of what is shown by the web browser after the link "CAdES-signature-1" is selected in the page shown in Figure 2.

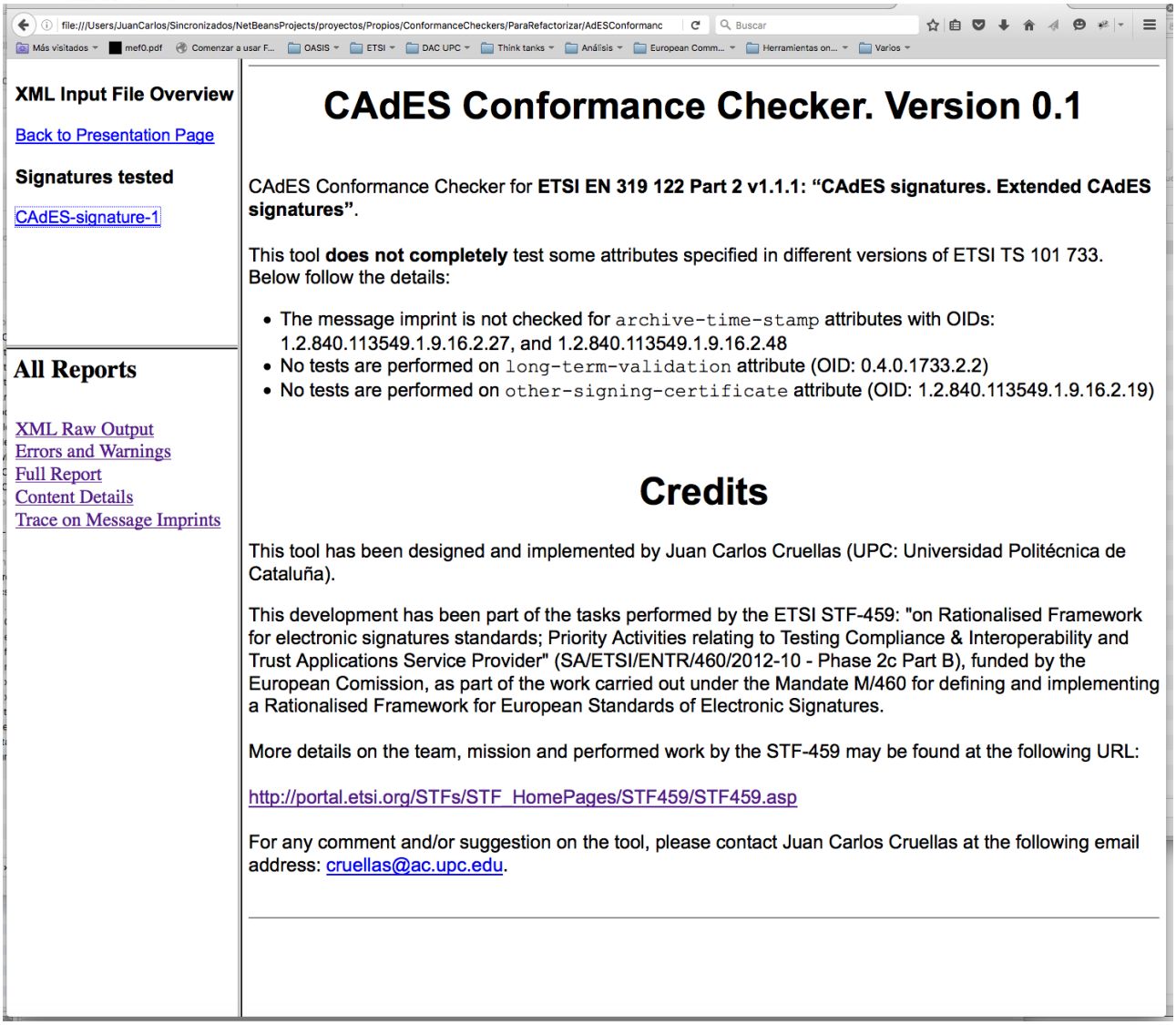


Figure 3: View of file "index.html" after selecting one of the CadES signatures in the "XML Input File Overview" frame.

5) **One or more folders with names "CAdES-signature-"<i>** (i has integer values starting in 1): One folder per each instance of SignerInfo type found within the CAdES structure (one per each CAdES signature found). Each folder will contain the following elements:

- **File "all-reports-frame.html"**: This page contains links to the different reports generated by the tool (5 reports for CAdES signatures). 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 one 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.

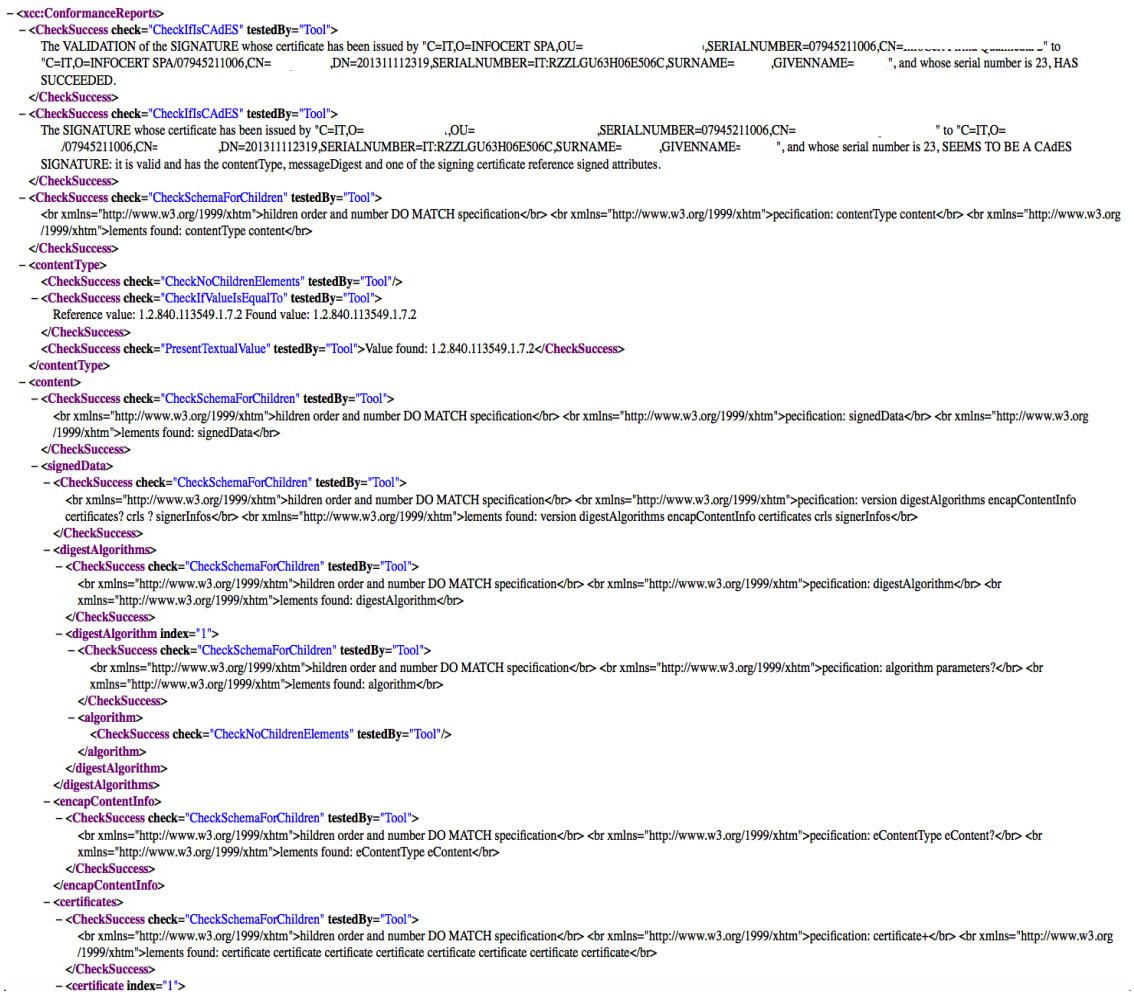
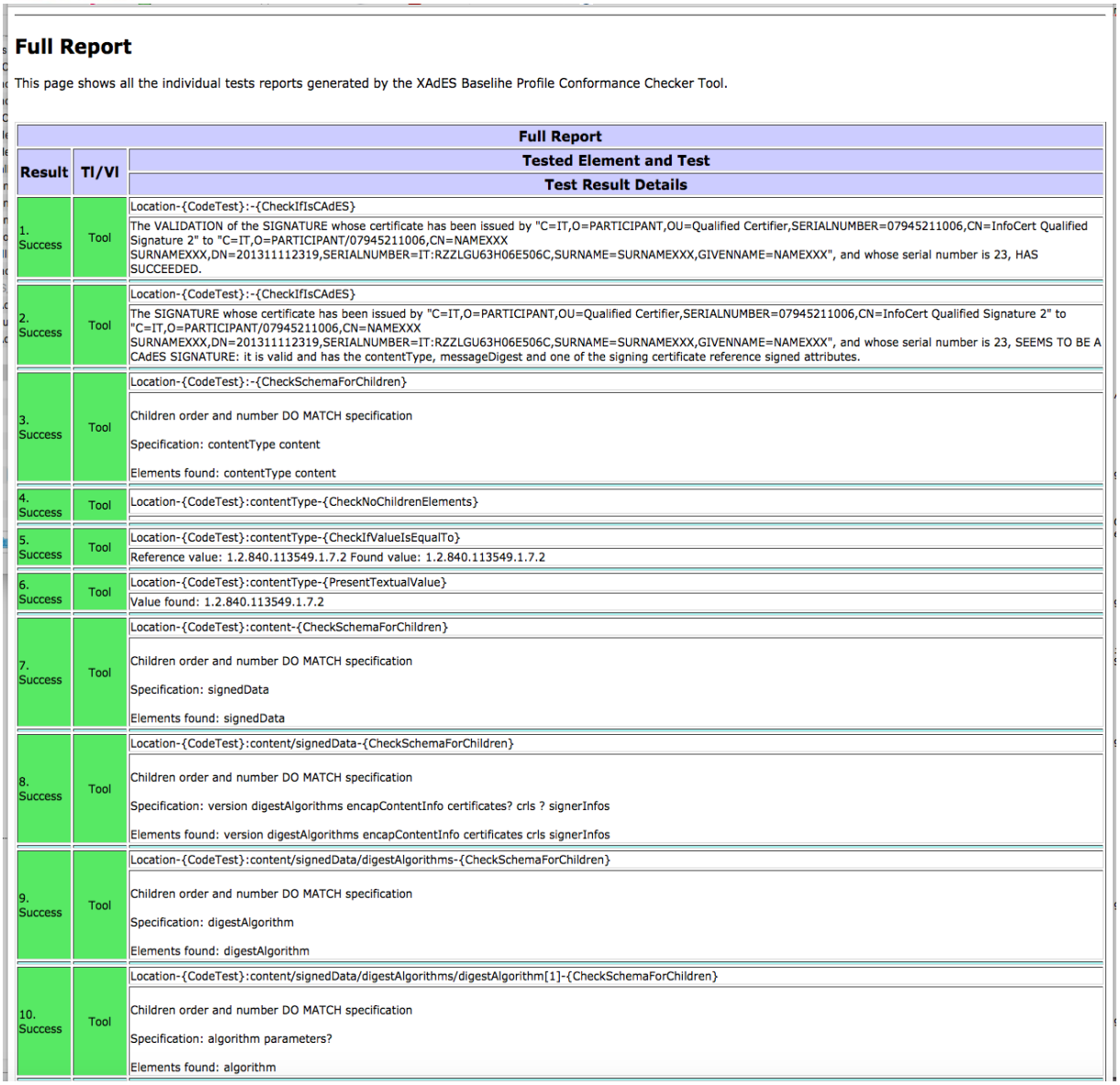


Figure 4: View of file "xmlrawoutputashtml.xml" corresponding to one CAdES signature.

- **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.



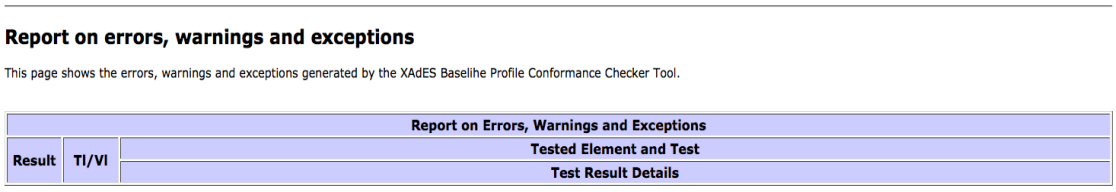
Full Report

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}:-{CheckIfIsCAdES} The VALIDATION of the SIGNATURE whose certificate has been issued by "C=IT,O=PARTICIPANT,OU=Qualified Certifier,SERIALNUMBER=07945211006,CN=InfoCert Qualified Signature 2" to "C=IT,O=PARTICIPANT/07945211006,CN=NAMEXXX SURNAMEXXX,DN=201311112319,SERIALNUMBER=IT:RZZLGU63H06E506C,SURNAME=SURNAMEXXX,GIVENNAME=NAMEXXX", and whose serial number is 23, HAS SUCCEEDED.
2. Success	Tool	Location-{CodeTest}:-{CheckIfIsCAdES} The SIGNATURE whose certificate has been issued by "C=IT,O=PARTICIPANT,OU=Qualified Certifier,SERIALNUMBER=07945211006,CN=InfoCert Qualified Signature 2" to "C=IT,O=PARTICIPANT/07945211006,CN=NAMEXXX SURNAMEXXX,DN=201311112319,SERIALNUMBER=IT:RZZLGU63H06E506C,SURNAME=SURNAMEXXX,GIVENNAME=NAMEXXX", and whose serial number is 23, SEEMS TO BE A CAdES SIGNATURE: it is valid and has the contentType, messageDigest and one of the signing certificate reference signed attributes.
3. Success	Tool	Location-{CodeTest}:-{CheckSchemaForChildren} Children order and number DO MATCH specification Specification: contentType content Elements found: contentType content
4. Success	Tool	Location-{CodeTest}:contentType-{CheckNoChildrenElements}
5. Success	Tool	Location-{CodeTest}:contentType-{CheckIfValueIsEqualTo} Reference value: 1.2.840.113549.1.7.2 Found value: 1.2.840.113549.1.7.2
6. Success	Tool	Location-{CodeTest}:contentType-{PresentTextualValue} Value found: 1.2.840.113549.1.7.2
7. Success	Tool	Location-{CodeTest}:content-{CheckSchemaForChildren} Children order and number DO MATCH specification Specification: signedData Elements found: signedData
8. Success	Tool	Location-{CodeTest}:content/signedData-{CheckSchemaForChildren} Children order and number DO MATCH specification Specification: version digestAlgorithms encapContentInfo certificates? crls ? signerInfos Elements found: version digestAlgorithms encapContentInfo certificates crls signerInfos
9. Success	Tool	Location-{CodeTest}:content/signedData/digestAlgorithms-{CheckSchemaForChildren} Children order and number DO MATCH specification Specification: digestAlgorithm Elements found: digestAlgorithm
10. Success	Tool	Location-{CodeTest}:content/signedData/digestAlgorithms/digestAlgorithm[1]-{CheckSchemaForChildren} Children order and number DO MATCH specification Specification: algorithm parameters? Elements found: algorithm

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 for a signature for which the CAdES conformance testing tool has not detected any errors and warnings.



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**

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 CAdES 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).

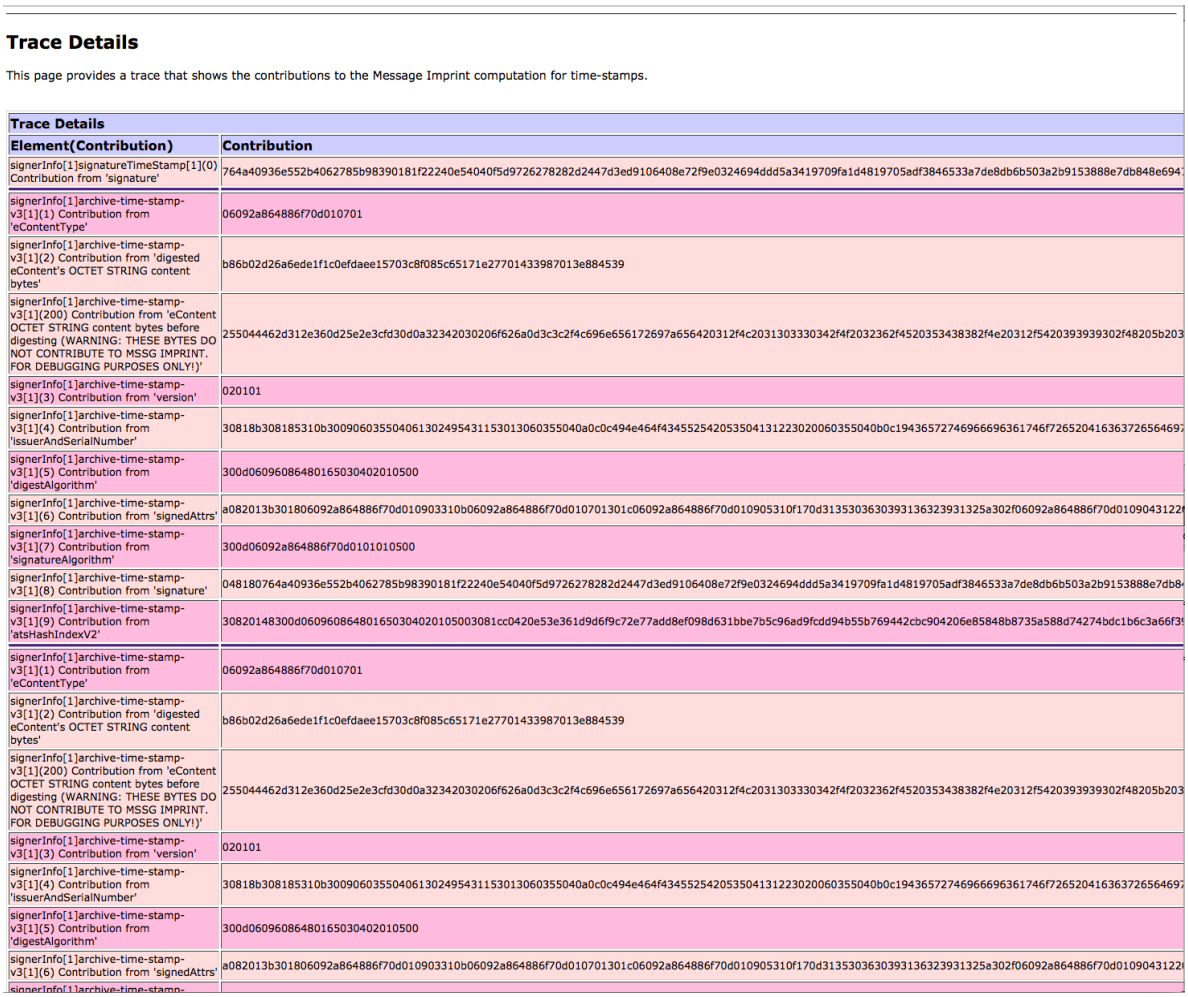
Signature Content Details

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

Signature Content Details		
Element	**Field**	**Value**
SignedData/certs/cert[1]	Issuer	C=FR,O=ETSI,OU=Plugtests_2015-2016,CN=RootCAOK
	SerialNumber	721978627550032864283856493130131814808534949 58
	Subject	C=FR,O=ETSI,OU=Plugtests_2015-2016,CN=RootCAOK
	NotBefore	Wed Apr 29 14:35:57 CEST 2015
	NotAfter	Mon Apr 29 14:35:57 CEST 2019
SignedData/certs/cert[2]	Issuer	C=IT,O=ParticipantNameXXX SPA,OU=TSA,SERIALNUMBER=07945211006,CN=ParticipantNameXXX Time Stamping Authority 2
	SerialNumber	8
	Subject	C=IT,O=ParticipantNameXXX SpA,OU=Qualified Certifier,CN=OCSP Responder Time Stamping Authority 02
	NotBefore	Tue Oct 15 10:50:33 CEST 2013
	NotAfter	Thu Oct 15 11:50:33 CEST 2015
SignedData/certs/cert[3]	Issuer	C=IT,O=ParticipantNameXXX SPA,OU=Qualified Certifier,SERIALNUMBER=07945211006,CN=ParticipantNameXXX Qualified Signature 2
	SerialNumber	19
	Subject	C=IT,O=ParticipantNameXXX SpA,OU=Qualified Certifier,CN=Ocsp Responder Qualified Signature 02
	NotBefore	Thu Oct 03 09:50:49 CEST 2013
	NotAfter	Sat Oct 03 10:50:49 CEST 2015
SignedData/certs/cert[4]	Issuer	C=IT,O=ParticipantNameXXX SPA,OU=TSA,SERIALNUMBER=07945211006,CN=ParticipantNameXXX Time Stamping Authority 2
	SerialNumber	3
	Subject	C=IT,O=ParticipantNameXXX SPA,OU=TSA,SERIALNUMBER=07945211006,CN=ParticipantNameXXX Time Stamping Authority 2
	NotBefore	Fri Apr 19 16:30:33 CEST 2013
	NotAfter	Thu Apr 19 17:30:33 CEST 2029
SignedData/certs/cert[5]	Issuer	C=IT,O=ParticipantNameXXX SPA,OU=Qualified Certifier,SERIALNUMBER=07945211006,CN=ParticipantNameXXX Qualified Signature 2
	SerialNumber	1
	Subject	C=IT,O=ParticipantNameXXX SPA,OU=Qualified Certifier,SERIALNUMBER=07945211006,CN=ParticipantNameXXX Qualified Signature 2
	NotBefore	Fri Apr 19 16:26:15 CEST 2013
	NotAfter	Thu Apr 19 17:26:15 CEST 2029
SignedData/certs/cert[6]	Issuer	C=IT,O=ParticipantNameXXX SPA,OU=TSA,SERIALNUMBER=07945211006,CN=ParticipantNameXXX Time Stamping Authority 2
	SerialNumber	58695
	Subject	C=IT,O=ParticipantNameXXX SPA,OU=TSA,CN=ICEDTS03201503
	NotBefore	Wed Mar 18 16:50:49 CET 2015
	NotAfter	Mon Mar 18 01:00:00 CET 2019
SignedData/certs/cert[7]	Issuer	C=FR,O=ETSI,OU=Plugtests_2015-2016,CN=RootCAOK
	SerialNumber	393802904868580
	Subject	C=FR,O=ETSI,OU=Plugtests_2015-2016,CN=TSA
	NotBefore	Wed Apr 29 19:36:25 CEST 2015
	NotAfter	Sat Apr 29 19:36:25 CEST 2017
SignedData/certs/cert[8]	Issuer	C=IT,O=ParticipantNameXXX SPA,OU=Qualified Certifier,SERIALNUMBER=07945211006,CN=ParticipantNameXXX Qualified Signature 2
	SerialNumber	23
	Subject	C=IT,O=ParticipantNameXXX SPA/07945211006,CN=CommonNameXXX,DN=201311112319,SERIALNUMBER=IT:RZZLGU63H06E506C,SURNAME=SurNameXXXX,GIVENNAME=NameXXX
	NotBefore	Fri Oct 11 10:05:22 CEST 2013
	NotAfter	Tue Oct 11 02:00:00 CEST 2016
signedData/cris/other[1] /ocspResponse /basicOCSPResponse[1]	responderID	C=FR,O=ETSI,OU=Plugtests_2015-2016,CN=RootCAOK
	producedAt	Mon Jul 06 10:42:50 CEST 2015
	resp[1].certID.serialNumber	393802904868580
	resp[1].certID.hashAlgorithm	1.3.14.3.2.26
	resp[1].certID.issuerNameHash	65138c95723d00bcd0698512a71b77ebc8686f82
signedData/cris/other[2] /ocspResponse /basicOCSPResponse[1]	responderID	C=IT,O=ParticipantNameXXX SpA,OU=Qualified Certifier,CN=OCSP Responder Time Stamping Authority 02
	producedAt	Mon Jul 06 10:42:50 CEST 2015
	resp[1].certID.serialNumber	58695
	resp[1].certID.hashAlgorithm	1.3.14.3.2.26
	resp[1].certID.issuerNameHash	9b9d1f65f5fb3577ef0963316c82d61ec91186f0
signedData/cris/other[3] /ocspResponse /basicOCSPResponse[1]	responderID	C=IT,O=ParticipantNameXXX SpA,OU=Qualified Certifier,CN=Ocsp Responder Qualified Signature 02
	producedAt	Mon Jul 06 10:42:50 CEST 2015
	resp[1].certID.serialNumber	23
	resp[1].certID.hashAlgorithm	1.3.14.3.2.26
	resp[1].certID.issuerNameHash	28c0cd90e9872ca0cd0f8026a293449e8b5a297a
SignedData/signerInfos /signerInfo[1] /signatureTimeStamp[1]	sid.Issuer	C=IT,O=ParticipantNameXXX SPA,OU=TSA,SERIALNUMBER=07945211006,CN=ParticipantNameXXX Time Stamping Authority 2
	sid.SerialNumber	58695
	genTime	4: C=IT,O=ParticipantNameXXX SPA,OU=TSA,CN=ICEDTS03201503
	genTime	Tue Jun 09 18:29:16 CEST 2015
	accuracy.seconds	1
	accuracy.mileseconds	0
	accuracy.microseconds	0
	ordering	false
	policy	1.3.76.36.1.1.1
	serialNumber	137277362
SignedData/signerInfos /signerInfo[1] /signatureTimeStamp[1] /content/signedData/certs /cert[1]	Issuer	C=IT,O=ParticipantNameXXX SPA,OU=TSA,SERIALNUMBER=07945211006,CN=ParticipantNameXXX Time Stamping Authority 2
	SerialNumber	58695
	Subject	C=IT,O=ParticipantNameXXX SPA,OU=TSA,CN=ICEDTS03201503
	NotBefore	Wed Mar 18 16:50:49 CET 2015
	NotAfter	Mon Mar 18 01:00:00 CET 2019
SignedData/signerInfos /signerInfo[1] /archiveTimeStampV3[1]	sid.Issuer	C=FR,O=ETSI,OU=Plugtests_2015-2016,CN=RootCAOK
	sid.SerialNumber	393802904868580
	genTime	Mon Jul 06 10:42:50 CEST 2015
	accuracy.seconds	1
	accuracy.mileseconds	1
	accuracy.microseconds	1
	ordering	false
	policy	1.3.6.1.4.1.2706.2.2.5.2.1.1.1
	serialNumber	14361721703173772
	nonce	11443196242837592146
SignedData/signerInfos /signerInfo[1] /archiveTimeStampV3[1] /content/signedData/certs /cert[1]	Issuer	C=FR,O=ETSI,OU=Plugtests_2015-2016,CN=RootCAOK
	SerialNumber	721978627550032864283856493130131814808534949 58
	Subject	C=FR,O=ETSI,OU=Plugtests_2015-2016,CN=RootCAOK
	NotBefore	Wed Apr 29 14:35:57 CEST 2015
	NotAfter	Mon Apr 29 14:35:57 CEST 2019
SignedData/signerInfos /signerInfo[1] /archiveTimeStampV3[1] /content/signedData/certs /cert[2]	Issuer	C=FR,O=ETSI,OU=Plugtests_2015-2016,CN=RootCAOK
	SerialNumber	393802904868580
	Subject	C=FR,O=ETSI,OU=Plugtests_2015-2016,CN=TSA
	NotBefore	Wed Apr 29 19:36:25 CEST 2015
	NotAfter	Sat Apr 29 19:36:25 CEST 2017
SignedData/signerInfos /signerInfo[1] /archiveTimeStampV3[1]	sid.Issuer	C=FR,O=ETSI,OU=Plugtests_2015-2016,CN=RootCAOK
	sid.SerialNumber	393802904868580
	genTime	Mon Jul 06 10:42:50 CEST 2015
	accuracy.seconds	1
	accuracy.mileseconds	1
	accuracy.microseconds	1
	ordering	false
	policy	1.3.6.1.4.1.2706.2.2.5.2.1.1.1
	serialNumber	14361721704843773
	nonce	976245696324079539
SignedData/signerInfos /signerInfo[1] /archiveTimeStampV3[1] /content/signedData/certs /cert[1]	Issuer	C=FR,O=ETSI,OU=Plugtests_2015-2016,CN=RootCAOK
	SerialNumber	721978627550032864283856493130131814808534949 58
	Subject	C=FR,O=ETSI,OU=Plugtests_2015-2016,CN=RootCAOK
	NotBefore	Wed Apr 29 14:35:57 CEST 2015
	NotAfter	Mon Apr 29 14:35:57 CEST 2019
SignedData/signerInfos /signerInfo[1] /archiveTimeStampV3[1] /content/signedData/certs /cert[2]	Issuer	C=FR,O=ETSI,OU=Plugtests_2015-2016,CN=RootCAOK
	SerialNumber	393802904868580
	Subject	C=FR,O=ETSI,OU=Plugtests_2015-2016,CN=TSA
	NotBefore	Wed Apr 29 19:36:25 CEST 2015
	NotAfter	Sat Apr 29 19:36:25 CEST 2017

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 CAdES 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. This is the document presented by the tool when the user selects the "Trace on Message Imprints" link in the "Signature Reports view" frame. Figure 8 shows an example of part of the contents of this file.



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'	764a40936e552b4062785b98390181f22240e54040f5d9726278282d2447d3ed9106408e72f9e0324694ddd5a3419709fa1d4819705adf3846533a7de8db6b503a2b9153888e7db848e694
signerInfo[1]archive-time-stamp-v3[1](1) Contribution from 'eContentType'	06092a864886f70d010701
signerInfo[1]archive-time-stamp-v3[1](2) Contribution from 'digested eContent's OCTET STRING content bytes'	b86b02d26a6ede1f1c0efdaee15703c8f085c65171e27701433987013e884539
signerInfo[1]archive-time-stamp-v3[1](200) Contribution from 'eContent OCTET STRING content bytes before digesting (WARNING: THESE BYTES DO NOT CONTRIBUTE TO MSSG IMPRINT. FOR DEBUGGING PURPOSES ONLY!)'	255044462d312e360d25e2e3cfd30d0a32342030206f626a0d3c3c2f4c696e656172697a656420312f4c2031303330342f4f2032362f4520353438382f4e20312f5420393939302f48205b203
signerInfo[1]archive-time-stamp-v3[1](3) Contribution from 'version'	020101
signerInfo[1]archive-time-stamp-v3[1](4) Contribution from 'issuerAndSerialNumber'	30818b308185310b3009060355040613024954311530130603550a0c0c494e464f4345525420535041312230200603550 40b0c194365727469666963617 46f726520416372656469
signerInfo[1]archive-time-stamp-v3[1](5) Contribution from 'digestAlgorithm'	300d060960864801650304020105 00
signerInfo[1]archive-time-stamp-v3[1](6) Contribution from 'signedAttrs'	a082013b301806092a864886f70d010903310b06092a864886f70d010701301c06092a864886f70d010905310f170d313530363039313632393132 5a302f06092a864886f70d0109043122
signerInfo[1]archive-time-stamp-v3[1](7) Contribution from 'signatureAlgorithm'	300d06092a864886f70d0101010500
signerInfo[1]archive-time-stamp-v3[1](8) Contribution from 'signature'	048180764a40936e552b4062785b98390181f22240e54040f5d9726278282d2447d3ed9106408e72f9e0324694ddd5a3419709fa1d4819705adf3846533a7de8db6b503a2b9153888e7db8
signerInfo[1]archive-time-stamp-v3[1](9) Contribution from 'atsHashIndexV2'	30820148300d060960864801650304020105003081cc0420e53e361d9d6f9c72e77add8ef098d631bbe7b5c96ad9fcdd94b55b769442cbc904206e85848b8735a588d74274bdc1b6c3a66f3
signerInfo[1]archive-time-stamp-v3[1](1) Contribution from 'eContentType'	06092a864886f70d010701
signerInfo[1]archive-time-stamp-v3[1](2) Contribution from 'digested eContent's OCTET STRING content bytes'	b86b02d26a6ede1f1c0efdaee15703c8f085c65171e27701433987013e884539
signerInfo[1]archive-time-stamp-v3[1](200) Contribution from 'eContent OCTET STRING content bytes before digesting (WARNING: THESE BYTES DO NOT CONTRIBUTE TO MSSG IMPRINT. FOR DEBUGGING PURPOSES ONLY!)'	255044462d312e360d25e2e3cfd30d0a32342030206f626a0d3c3c2f4c696e656172697a656420312f4c2031303330342f4f2032362f4520353438382f4e20312f5420393939302f48205b203
signerInfo[1]archive-time-stamp-v3[1](3) Contribution from 'version'	020101
signerInfo[1]archive-time-stamp-v3[1](4) Contribution from 'issuerAndSerialNumber'	30818b308185310b3009060355040613024954311530130603550a0c0c494e464f4345525420535041312230200603550 40b0c194365727469666963617 46f726520416372656469
signerInfo[1]archive-time-stamp-v3[1](5) Contribution from 'digestAlgorithm'	300d060960864801650304020105 00
signerInfo[1]archive-time-stamp-v3[1](6) Contribution from 'signedAttrs'	a082013b301806092a864886f70d010903310b06092a864886f70d010701301c06092a864886f70d010905310f170d313530363039313632393132 5a302f06092a864886f70d0109043122

Figure 8: View of file "mssgimprintsdetails.html" corresponding to the time-stamp tokens found within one CAdES signature.

4 Contents of the reports generated by CAdESCC.

4.1 Introduction

This clause provides a detailed explanation of the reports that CAdESCC may generate when checking a certain CAdES signature. The contents of the full report are classified in contents that may also be contents of the full report generated by other conformance-testing tools and contents that are specific to the CAdESCC.

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 component 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: `content/signedData/digestAlgorithms/digestAlgorithm[1]`, indicates that the tested component is the first `digestAlgorithm` child component of `digestAlgorithms`, child of `signedData`, chil of `content` root component of the CAdES signature.

- 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

4.2.3.1 CheckSchemaForChildren test

This entry of the table reports on a test that checks if the children of a certain component appear in the order and with the cardinality specified by the syntax defined by the reference specification (ASN.1 definition in the case of CAdES signatures, XML Schema in the case of XAdES signatures, the definition of the PDF dictionary in the case of PAdES signatures).

The additional information row provides the details of what the tool expects to find according to the syntax and what it actually appears in the signature. The text in this row uses special characters for indicating the syntax defined in the reference specification:

? Indicates that a certain child is optional

* Indicates that 0 or several children with the same name may appear

+ Indicates that at least 1 child with the indicated name has to appear.

|| Indicates a choice between a child with the name that appears at its left, or a child with the name that appears at its right.

EXAMPLE:

Location-{CodeTest}:content/signedData/signerInfos/signerInfo[1]-{CheckSchemaForChildren}
Children order and number DO MATCH specification Specification: version (issuerAndSerialNumber || subjectKeyIdentifier) digestAlgorithm signedAttrs signatureAlgorithm signature unsignedAttrs? Elements found: version issuerAndSerialNumber digestAlgorithm signedAttrs signatureAlgorithm signature unsignedAttrs

These two rows indicate that the test has checked if the contents of the signerInfo component are conformant to the ASN.1 specification (test CheckSchemaForChildren).

The second row first indicates that the contents match the specification: both the order and the cardinality of the children are correct. Furthermore the tool provides information of the ASN.1 specification and also of the children actually found within the component. In case of error, this allows quickly identify the cause of the error.

4.2.3.2 CheckNoChildrenElements test

This entry of the table reports on a test that checks that a certain component does not contain any children. The additional information row only appears in case the component does contain any children (this would indicate that the signature is not conformant against the reference specification).

4.2.3.3 CheckIfValueIsEqualTo test

This entry of the table reports on a test that checks that a certain component has a certain value. The additional information row shows both values, the one defined by the reference specification and the value of the tested component.

EXAMPLE:

Location-{CodeTest}:contentType-{CheckIfValueIsEqualTo}
Reference value: 1.2.840.113549.1.7.2 Found value: 1.2.840.113549.1.7.2

4.2.3.4 CheckIfValueIsOneOfDefined test

This entry of the table reports on a test that checks that the value of a certain component is one of the members of a set of values. The additional information row shows the set of allowed values according tothe reference specification and the value of the tested component.

EXAMPLE:

Location-{CodeTest}:content/signedData/crls/other[1]/otherRevInfoFormat-{CheckIfValueIsOneOfDefined}
Found value: '1.3.7.1.5.5.7.17.2'. Allowed values: 1.3.7.1.5.5.7.48.1.1,1.3.7.1.5.5.7.17.2

4.2.3.5 ExecuteCommandsInCaseTextValueOfChild test

This entry of the table reports on a test that checks that the value of a certain child of a specific component is one of the members of a set of values. In case this is true, the conformance testing tool executes the tests indicated by the XML-encoded commands that are children of the command that has triggered this test (see clause 5 for a high level overview of the components of the conformance-testing tools, including XML-encoded commands).

The additional information row shows the set of allowed values according tothe reference specification and the value of the tested component.

EXAMPLE:

Location-{CodeTest}:content/signedData/crls/other[1]-{ExecuteComamandsInCaseTextValueOfChild}
Found value is one of the allowed values. Found value 1.3.7.1.5.5.7.17.2. Allowed values: 1.3.7.1.5.5.7.17.2

4.2.3.6 MustNotBeEmpty test

This entry of the table reports on a test that checks that a certain component is not empty. The additional information row does not appear.

EXAMPLE:

Location-{CodeTest}:content/signedData/signerInfos/signerInfo[1]/signedAttrs/attribute[1]/attrValues-{MustNotBeEmpty}

4.2.3.7 PresentTextValue indication

This entry of the table does not report of any test. Instead it only presents the textual value of a certain component of the signature in the additional information row.

EXAMPLE:

Location-{CodeTest}:content/signedData/signerInfos/signerInfo[1]/unsignedAttrs/attribute[1]/attrValues/signatureTimeStamp[1]/contentType-{PresentTextualValue}
Value found: 1.2.840.113549.1.7.2

4.2.3.8 CheckIfX509Certificate test

This entry of the table reports on a test that checks that the content of a certain component is an X.509 certificate. No additional information row is generated for the full report.

EXAMPLE:

Location-{CodeTest}:content/signedData/certificates/certificate[1]-{CheckIfX509Certificate}

The command that implements this test includes within the XML raw report additional elements that contain relevant details of the X.509 certificate itself, so that they can be presented in the content details report. The details included in the report are the its `Issuer`, `serialNumber`, `Subject`, and validity period

EXAMPLE OF CONTENTS GENERATED FOR THE CONTENT DETAILS REPORT:

SignerInfo[1]/CertificateValues/Certificate[1]	Issuer	C=FR,O=ETSI,OU=Plugtests STF-351 2008-2009,CN=RootCAOK
	SerialNumber	1
	Subject	C=FR,O=ETSI,OU=Plugtests STF-351 2008-2009,CN=RootCAOK
	NotBefore	Fri Aug 22 16:46:34 CEST 2008
	NotAfter	Thu Aug 22 16:46:34 CEST 2013

4.2.3.9 CheckIfX509CRL test

This entry of the table reports on a test that checks that the content of a certain component is an X.509 CRL. No additional information row is generated for the full report.

Below follows an example of how the tool reports success when it successfully parses a X509 CRL:

EXAMPLE:

Location-{CodeTest}:content/signedData/crls/crl[1]-{CheckIfX509CRL}

The command that implements this test includes within the XML raw report additional elements that contain relevant details of the X.509 CRL itself, so that they can be presented in the content details report. The details included in the report are the its `version`, `Issuer`, `thisUpdate`, and `NextUpdate`.

EXAMPLE OF CONTENTS GENERATED FOR THE CONTENT DETAILS REPORT:

SignerInfo[1]/RevocationValues/CRL[2]	version	2
	Issuer	CN=LevelACAOK, OU=Plugtests_2013-2014, O=ETSI, C=FR
	thisUpdate	Mon Dec 02 16:38:17 CET 2013
	nextUpdate	Wed Jan 01 16:38:17 CET 2014

4.2.3.10 CheckIsBasicOCSPResponse test

This entry of the table reports on a test that checks that the content of a certain component is an instance of `BasicOCSPResponse` type. No additional information row is generated for the full report.

Below follows an example of how the tool reports success when it successfully parses an instance of `BasicOCSPResponse` type:

EXAMPLE:

Location-{CodeTest}:content/signedData/crls/other[1]/basicOCSPResponse-{CheckIsBasicOCSPResponse}

The command that implements this test includes within the XML raw report additional elements that contain relevant details of the instance of `BasicOCSPResponse` type itself, so that they can be presented in the content details report. The details included in the report are the its `responderId`, `producedfAt`, `certID.serialNumber`, `certID.hashAlgorithm`, and `certID.issuerNameHash`.

EXAMPLE OF CONTENTS GENERATED FOR THE CONTENT DETAILS REPORT:

SignerInfo[1]/ RevocationValues/OCSP[2]	responderID	C=FR,O=ETSI,OU=Plugtests_2013-2014, CN=LevelACAOK
	producedAt	Tue Dec 03 09:45:16 CET 2013

	resp[1].certID.serialNumber	2149705798169604440
	resp[1].certID.hashAlgorithm	2.17.840.1.101.3.4.2.1
	resp[1].certID.issuerNameHash	645f831443e1cdbe81c9e0dbf8ca021a2da481fc86966e806939cbeb6d3a3c96

4.2.3.11 CheckIfOCSPValue test

This entry of the table reports on a test that checks that the content of a certain component is an instance of `OCSPResponse` type. No additional information row is generated for the full report.

Below follows an example of how the tool reports success when it successfully parses an instance of `OCSPResponse` type:

EXAMPLE:

Location-{CodeTest}:content/signedData/signerInfos/signerInfo[1]/unsignedAttrs[1]/attribute[6]/attrValue/revocationValues[1]/ocspVals/ocsps/ocsp[1]-{CheckIfOCSPValue}

The command that implements this test includes within the XML raw report additional elements that contain relevant details of the instance of `OCSPResponse` type itself, so that they can be presented in the content details report. The details included in the report are the its `responderId`, `producedfAt`, `certID.serialNumber`, `certID.hashAlgorithm`, and `certID.issuerNameHash`.

EXAMPLE OF CONTENTS GENERATED FOR THE CONTENT DETAILS REPORT:

SignerInfo[1]/RevocationValues/OCSP[1]	responderID	C=FR,O=ETSI,OU=Plugtests_2013-2014,CN=LevelBCAOK
	producedAt	Mon Dec 02 08:58:44 CET 2013
	resp[1].certID.serialNumber	428364045964074
	resp[1].certID.hashAlgorithm	1.3.14.3.2.26
	resp[1].certID.issuerNameHash	459153f43a314cdf36e74a189e5c8a3f828a0f1e

4.2.3.12 CheckIfTimeStampToken test

This entry of the table reports on a test that checks that the content of a certain component is an instance of `TimeStampToken` type. No additional information row is generated for the full report.

Below follows an example of how the tool reports success when it successfully parses an instance of `TimeStampToken` type:

EXAMPLE:

Location-{CodeTest}:content/signedData/signerInfos/signerInfo[1]/unsignedAttrs[1]/attribute[4]/attrValue/timeStampedCertsCRLsReferences[1]-{CheckIfTimeStampToken}

The command that implements this test includes within the XML raw report additional elements that contain relevant details of the instance of `TimeStampToken` type itself, so that they can be presented in the content details report. The details included in the report are the its `sid.Issuer`, `sid.serialNumber`, `genTime`, , `accuracy.seconds`, `accuracy.miliseconds`, and `accuracy.microseconds`.

EXAMPLE OF CONTENTS GENERATED FOR THE CONTENT DETAILS REPORT:

timeStampedCertsCRLsReferences[1]	sid.Issuer	C=FR,O=ETSI,OU=Plugtests_2013-2014,CN=RootCAOK
	sid.SerialNumber	468690769505377
	genTime	Mon Dec 02 08:58:45 CET 2013
	accuracy.seconds	1

	accuracy.mileseconds	1
	accuracy.microseconds	1
	ordering	false
	policy	1.3.7.1.4.1.2707.2.2.5.2.1.1.1
	serialNumber	1385971125666221
	nonce	523011182614978847100050062058505187369966798853

4.2.3.13 CheckIfMssgImpIsAsInSpec test

This entry of the table reports on a test that checks that the message imprint found in the time-stamp token, encapsulated within an attribute/qualified property, is the message imprint that should be found according the reference specification. For checking this, the conformance testing tool computes the message imprint value of this time-stamp token according to the reference specification and compares with the value found within the time-stamp token.

The additional information row contains the values of the computed and present message imprint values.

EXAMPLE:

Location-{CodeTest}:content/signedData/signerInfos/signerInfo[1]/unsignedAttrs/attribute[2]/attrValues/archiveTimeStampV3[1]-{CheckIfMssgImpIsAsInSpec}
Both the message imprint found in the time-stamp token and the computed one by the tool, have the same value: 4ee813156f462269700d5848efb5bc634c75d23f2b4483fe51caa1de86184db8

Each time this test is executed (except for the time-stamp token encapsulated within a DocumentTimeStamp dictionary in a PAdES signature), the tool incorporates to the XML raw report, all the individual contributions that are concatenated for building the input to the message imprint computation. The tool also indicates what are the sources of these individual contributions.

The conformance-testing tools show these contributions in the “trace on message imprints” report. Implementers may use the details shown in this report for identifying, in case of negative result, the exact point(s) where their tools and the conformance testing tool compute a different contribution for building the input to the message imprint computation. This certainly helps to build common understanding of the specification and speed up building correct implementations.

EXAMPLE OF TRACE OF MESSAGE IMPRINT COMPUTATIONS:

Trace Details	
Element(Contribution)	**Contribution**
signerInfo[1]archive-time-stamp-v3[1](1) Contribution from 'eContentType'	06092a864886f70d010701
signerInfo[1]archive-time-stamp-v3[1](2) Contribution from 'digested eContent's OCTET STRING content bytes'	24966de3536df15b186a13fe5ed8b8ad1def0439147d177a82ab88b65e91e4eb
signerInfo[1]archive-time-stamp-v3[1](200) Contribution from 'eContent OCTET STRING content bytes before digesting (WARNING: THESE BYTES DO NOT CONTRIBUTE TO MSSG IMPRINT. FOR DEBUGGING PURPOSES ONLY!)'	746f42655369676e6564

Trace Details	
Element(Contribution)	**Contribution**
signerInfo[1]archive-time-stamp-v3[1](3) Contribution from 'version'	020101
signerInfo[1]archive-time-stamp-v3[1](4) Contribution from 'issuerAndSerialNumber'	305a304f310b3009060355040613024652310d300b060355040a130445545349311c301a060355040b0c13506c756774657374735f323031332d32303134311330110603550403130a4c6576656c4243414f4b0207034f87ee8aa85c
signerInfo[1]archive-time-stamp-v3[1](5) Contribution from 'algorithmIdentifier'	300d06096086480165030402010500
signerInfo[1]archive-time-stamp-v3[1](6) Contribution from 'signedAttrs'	a082010730180609 2a864886f70d010903310b06092a864886f70d010701301c06092a864886f70d010905310f170d313331323033303831343232 5a302f06092a864886f70d010904312204202496 6de3536df15b186a13fe5ed8b8ad1def0439147d177a82ab88b65e91e4eb30819b060b2a864886f70d010910022f31818b3081883081853081820420 47287703ae54ee756a9f4df3797f6006abdd3b46614b420391a1f59604c241ac305e3053a451304f310b3009060355040613024652310d300b060355040a130445545349311c301a060355040b0c13506c756774657374735f323031332d32303134311330110603550403130a4c6576656c4243414f4b0207034f87ee8aa85c
signerInfo[1]archive-time-stamp-v3[1](7) Contribution from 'algorithmIdentifier'	300d06092a864886f70d0101010500
signerInfo[1]archive-time-stamp-v3[1](8) Contribution from 'signature'	04820100431611c7126e74666041a2e2817a6dc7f6426d688609ce22c58e170d56abc83fa05bc172d756aeacfd5ea24fe69819f9545324167fe44db515bb1385df4f9edc1c435c1c63554c4d6be4f79943cc1568e62d57dcdc941f750bebddb45b9d21eb0d20f4e6c67ebf8cff816e309c84292e1b542e0bb99e831c9341f1e43c73d62c1d9f986772f5124ca7c10687c07bb42969f37e1e0fa9a044d16e8a3466d646c5eb93cdc4b69034dcc8ef7e0e3b7fa8cb931ba130ce942c624984f6eb892f2ef3f3853566522efbf13692b5ed39b2f0bfc66c59227e0ee91f8ccbc9a1b1dc815d4033102b5a210cf4c00f5007c8691c9a99e52d2b83555021c4491538089c043c
signerInfo[1]archive-time-stamp-v3[1](9) Contribution from 'atsHashIndex'	3081de06096086480165030402013022042047287703ae54ee756a9f4df3797f6006abdd3b46614b420391a1f59604c241ac30003081aa04201bcbaf2d87f4658064c6c9f6ec8b0ef76ef296864d2a8289312c4ff1be05ccfb042015b27b09c759998ffb5eba9f40a9231fdeae7b4910792fc413250e098745706c0420bef9ef6d8d570ba07cbf3dc1b235781c54cf2591d2f1f6048f13cb1d2c56ea0004207ff3ca026a57bfdbce4184e575bcc02a93c366194ae0410cf563d4de593a7e7c0420ad82955f65aa0f982842d0c4773b9bb2890b6af59410f97069f88e46d06174ba

The table shows the different contributions (encoded in hexadecimal) that are concatenated for building up the input to the message imprint computation, as generated by the tool. More specifically:

1) Contribution (1) from `eContentType` field is shown in row 1 of the table.

2) Contribution resulting from digesting (with the digest algorithm used for computing the archive time-stamp token's message imprint) the content that has been used for computing the digest value encapsulated within the `message-digest` signed attribute, is shown in row 2 of the table.

3) Row 3 of the table DOES NOT show any contribution to the message imprint computation, but the octets whose digest (using the digest algorithm used for computing the archive time-stamp token's message imprint) is shown in row 2.

4) Contribution of `version` field in `signedData.signerInfo`'s item corresponding to the CAdES signature being checked is shown in row 4 of the table.

5) Contribution of `sid` field in signedData.signerInfo's item corresponding to the CAdES signature being checked is shown in row 5 of the table.

6) Contribution of `digestAlgorithm` field in `signedData.signerInfo`'s item corresponding to the CAdES signature being checked is shown in row 6 of the table.

7) Contribution of `signedAttrs` field in `signedData.signerInfo`'s item corresponding to the CAdES signature being checked is shown in row 7 of the table.

8) Contribution of `signatureAlgorithm` field in `signedData.signerInfo`'s item corresponding to the CAdES signature being checked is shown in row 8 of the table.

9) Contribution of `signature` field in `signedData.signerInfo`'s item corresponding to the CAdES signature being checked is shown in row 9 of the table.

10) Contribution from `ats-hash-index-v3` unsigned attribute present within the `archive-time-stamp-v3` is shown in row 10 of the table.

4.2.3.13 CheckIfX509AttributeCertificate test

This entry of the table reports on a test that checks if the content of a certain component of the signature is an X509 attribute certificate. No additional information row is generated for the full report.

EXAMPLE:

Location-{CodeTest}:content/signedData/signerInfos/signerInfo[1]/signedAttrs/attribute[5]/attrValues/signerAttributesV2[1]/certifiedAttributesV2/certifiedAttributesVals/attributeCertificate[1]/attributeCertificateVal-{CheckIfX509AttributeCertificate}

The command that implements this test includes within the XML raw report additional elements that contain relevant details of the X.509 attribute certificate itself, so that they can be presented in the content details report. The details included in the report are the its `version`, `holderCertificate.issuer`, `holderCertificate.serialNumber`, `issuer`, `serialNumber`, `notBefore`, `notAfter`, and `attr`.

4.2.4 Contents specific to CAdES conformance testing

4.2.4.1 Introduction

This clause provides details of components of the full report that are generated when testing conformance of a CAdES signature. All these tests can appear within a full report generated by the CAdES conformance testing tool. Some of these tests can also appear within a full report generated by the PAdES conformance testing tool.

4.2.4.2 CheckIfIsCAdES test

This component indicates whether the signature whose details are provided in within the additional information row is a CAdES signature (i.e., it contains the mandatory signed attributes as specified by ETSI TS 319 122 Part 1). It also performs a cryptographic verification of the digital signature value.

This command also cryptographically verifies the digital signature value for each countersignature incorporated in any of the parallel signatures.

EXAMPLE:

Location-{CodeTest}:-{CheckIfIsCAdES}
The VALIDATION of the SIGNATURE whose certificate has been issued by "C=IT,O=INFOCERT SPA,OU=Certificatore Accreditato,SERIALNUMBER=07945211006,CN=InfoCert Firma Qualificata 2" to "C=IT,O=INFOCERT SPA/07945211006,CN=Luigi Rizzo,DN=201311112319,SERIALNUMBER=IT:RZZLGU63H06E506C,SURNAME=RIZZO,GIVENNAME=LUIGI",

and whose serial number is 23, HAS SUCCEEDED.

The LOCATION part of the first row is empty in this case. The additional information row provides all the required information for identifying the tested signature, which may be one of the CAdES parallel signatures present within the CAdES structure, or a counter-signature.

4.2.4.3 CheckAllowedAttributes test

This component of the report reports the result of the test that checks if the signed or unsigned attributes present within the signature are attributes allowed by the reference specification.

The additional information row indicates what attributes are allowed by the reference specification and which attributes have been found by the tool.

The CheckSchemaForChildren component can not report on this specific aspect because the children element of the `signedAttrs` and `unsignedAttrs` fields within a CAdES signature, are `attrType` and `attrValues` fields.

EXAMPLE:

Location-{CodeTest}:content/signedData/signerInfos/signerInfo[1]/signedAttrs-{CheckAllowedAttributes}
Children order and number DO MATCH specification Specification: (contentType || messageDigest || signingTimeUTCTime || SigningTimeGeneralizedTime || essSigningCertificate || essSigningCertificateV2 || commitmentTypeIndication || contentHints || signerLocation || signerAttributes || signerAttributesV2 || contentTimeStamp || signaturePolicyIdentifier || contentReference || contentIdentifier)* Elements found: contentType signingTimeUTCTime messageDigest essSigningCertificateV2

In this example the first line of the additional information row reports that the check has succeeded. The second line shows that the reference specification allows the incorporation of certain number of signed attributes in any order. The last line shows the signed attributes actually found within the CAdES signature.

4.2.4.4 CheckAttributesCardinality test

This component reports on a complementary test performed after the previous one. Once it has been checked that all the signed or unsigned attributes present within the CAdES signature are allowed by the reference specification , the conformance testing tool checks that the cardinalities of each attribute are conformant with the reference specification.

The additional information row reports on any discrepancy between the cardinalities as defined by the reference specification and the cardinalities of any of the attributes present within the signature.

EXAMPLE:

Location-{CodeTest}:content/signedData/signerInfos/signerInfo[1]/signedAttrs-{CheckAttributesCardinality}
The cardinalities of all the attributes match their respective specifications

4.2.4.5 CheckOnlyOneAttrOfTheListPresent test

In certain occasions the reference specification forces a choice between two or more attributes. This component reports on a test that checks if the set of attributes present within the signature includes one and only one of a set of attributes defined by the reference specification defining the scope of the choice.

The additional information row provides details of both, the attribute actually found within the signature and the different choices forced by the specification.

EXAMPLE:

Location-{CodeTest}:content/signedData/signerInfos/signerInfo[1]/signedAttrs-{CheckOnlyOneAttrOfTheListPresent}
Only one of the attributes in the following list have to be present: essSigningCertificate essSigningCertificateV2 . The attribute essSigningCertificateV2 has been found

In this example the CAdES conformance testing tool reports that only the `essSigningCertificateV2` attribute is found, and that the ETSI EN 319 122 Part 1 specifies that within the set of signed attributes only one occurrence of one of `essSigningCertificate` and `essSigningCertificateV2` shall be present.

4.2.4.6 CheckIsAGivenType test

This component reports on a test that checks that a certain component is of a certain given ASN.1 type. This test is usually performed on types whose contents are defined by the value of an OID.

The additional information row provides information of both, what should be the type according the reference specification and the actual type found by the conformance-testing tool. Sometimes it also reports on the value of this component.

EXAMPLE:

Location-{CodeTest}:content/signedData/signerInfos/signerInfo[1]/signedAttrs/attribute[1]/attrValues/contentType[1]-{CheckIsAGivenType}
This component is a(n) OID. Tag value found: 7. Value found: 1.2.840.113549.1.7.1

4.2.4.7 CheckIfIsDigestOfPKIDataInSignedData test

This component reports on a test that checks if one of the digest values present within one of the set of digest values present within the `ats-hash-index-v3` field is the actual digest value of one of the validation data values present within `signedData` field of the CAdES signature.

If the component being cheked is a one of the components within the `ats-hash-index-v3.certificateHashIndex` it reports if this value is the digest value of one of the X509 certificates present within `signedData.certificates` field of the CAdES signature that contains the `archive-time-stamp-v3` that incorporates the aforementioned `ats-hash-index-v3` attribute.

If the component being cheked is a one of the components within the `ats-hash-index-v3.crlsHashIndex` it reports if this value is the digest value of one of the X509 CRLs, or instances of `OCSPResponse` type, or instances of `BasicOCSPResponse` present within `signedData.crls` field of the CAdES signature that contains the `archive-time-stamp-v3` that incorporates the aforementioned `ats-hash-index-v3` attribute

The additional information row provides information that fully identifies the validation data whose digest value is equal to the checked digest value within `ats-hash-index-v3` child.

The CAdES conformance-testing tool also incorporates code for testing `ats-hash-index` and `ats-hash-index-v2`. These versions of the tool were used in CAdES Plugtests™ events that tested interoperability among tools that implemented previous draft versions of ETSI EN 319 122 Part 1.

EXAMPLE:

Location-{CodeTest}:content/signedData/signerInfos/signerInfo[1]/unsignedAttrs/attribute[2]/attrValues/archiveTimeStampV3[1]/content/signedData/signerInfos/signerInfo[1]/unsignedAttrs/attribute[1]/attrValues/atsHashIndexV2[1]/certificatesHashIndex/

certificateHashIndex[1]-{CheckIfIsDigestOfPKIDataInSignedData}
The digest with value: e53e361d9d6f9c72e77add8ef098d631bbe7b5c96ad9fcdd94b55b769442cbc9 corresponds to the value within SignedData with the following details: Issuer: C=IT,O=INFOCERT SPA,OU=Certificatore Accreditato,SERIALNUMBER=07945211006,CN=InfoCert Firma Qualificata 2; SerialNumber: 23; Subject: C=IT,O=INFOCERT SPA/07945211006,CN=Luigi Rizzo,DN=201311112319,SERIALNUMBER=IT:RZZLGU63H06E506C,SURNAME=RIZZO,GIVENNAME=LUIGI; NotBefore: Fri Oct 11 10:05:22 CEST 2013; NotAfter: Tue Oct 11 02:00:00 CEST 2016

At the time the present document was written no CAdES signatures incorporating `ats-hash-index-v3` had been found for testing conformance.

4.2.4.8 CheckIfIsDigestOfUnsignedAttrs test

This component reports on a test that checks if one of the digest values present within one of the set of digest values present within the `ats-hash-index-v3.unsignedAttrValuesHashIndex` field is the actual digest value of one of the unsigned attributes of the CAdES signature.

The additional information row provides information that fully identifies the unsigned attribute whose digest value is equal to the checked digest value within `ats-hash-index-v3` child.

EXAMPLE:

Location-{CodeTest}:content/signedData/signerInfos/signerInfo[1]/unsignedAttrs/attribute[2]/attrValues/archiveTimeStampV3[1]/content/signedData/signerInfos/signerInfo[1]/unsignedAttrs/attribute[1]/attrValues/atsHashIndexV2[1]/unsignedAttrsHashIndex/unsignedAttrHashIndex[1]-{CheckIfIsDigestOfUnsignedAttrs}
The unsigned attribute digest with value: 6cbc46d408e72c7fdfe9e6330d4f0d804c1f7b9e1e9a2b6ebac0c77c9079ff32 corresponds to the unsigned attribute value within SignedData with the following details: signatureTimeStamp[1] Oct 11 10:05:22 CEST 2013; NotAfter: Tue Oct 11 02:00:00 CEST 2016

4.2.4.9 CheckIfIsDigestOfUnsignedAttrsValuesInstances test

This component reports on a test that checks if one of the digest values present within one of the set of digest values present within the `ats-hash-index-v3.unsignedAttrValuesHashIndex` field is the actual digest value of the concatenation of the `Attribute.attrType` field and one of the instances of `AttributeValue` within the `Attribute.attrValues` within the `unsignedAttrs` field.

The additional information row shows the digest value of the instance of `AttributeValue` type within the unsigned attribute. As the `unsignedAttrs` field is a SET OF it can not be identified by its order of appearance in set.

At the moment this document was written no CAdES signature with an `archive-time-stamp-v3` incorporating an `ats-hash-index-v3` had been yet tested.

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, CAdES signatures have only fields, whereas XAdES signatures, being XML based signatures, have elements and attributes. In PAdES signatures, the components may be fields of CAdES 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 CAdES 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 [3], 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 XML 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.

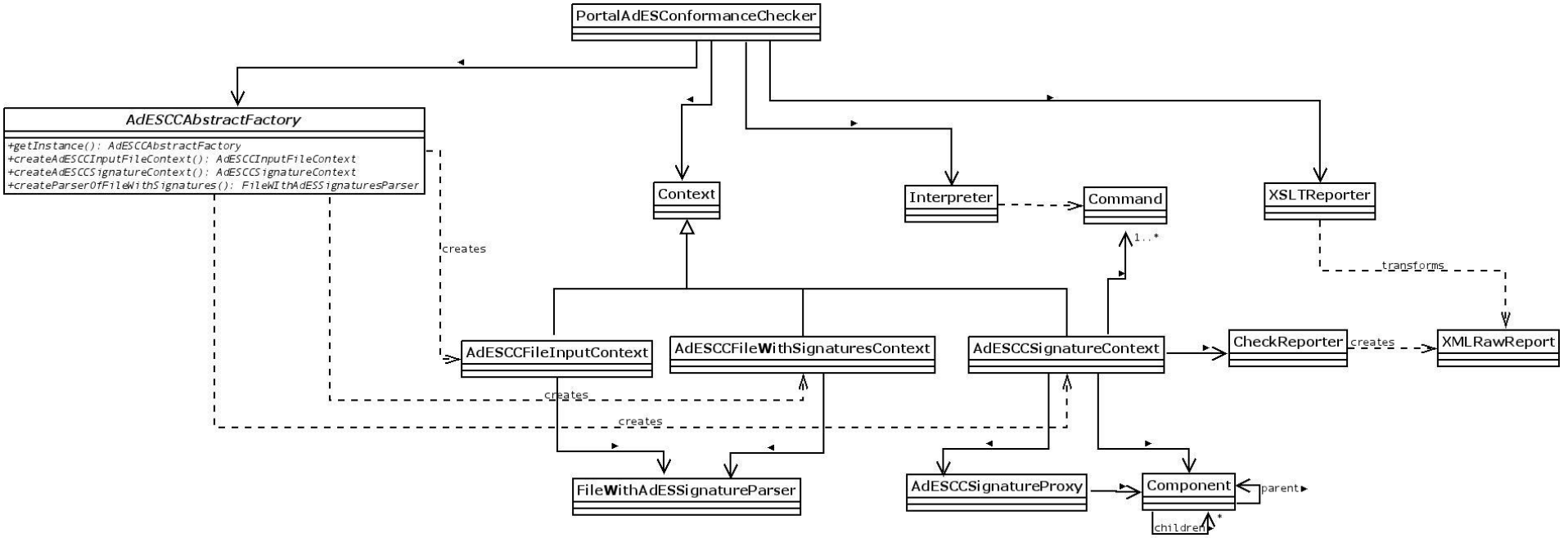


Figure 9: Basic conceptual model of the conformance-testing tools

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 re**presenting 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 CAdES signatures generated by participants in the CAdES interoperability

remote plugtest™ started on 2nd December 2013. 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 Highlights of CAdES conformance-testing tools design

This clause highlights some relevant design aspects that are specific to the CAdES conformance-testing tool.

7.1 Dealing with ASN.1, BER and DER

Following what was established within the STF-459 Terms of Reference ("The STF will use already existing open source tools as much as possible for the production of these tools"), the CAdES conformance-testing tools make use of the **subset of classes present within bouncy castle tools, which provide support to the subset of ASN.1 syntax and BER and DER encodings that are needed for successfully implementing CMS signatures** (on which CAdES is built).

The reasons for this decision have been:

- This is an open source tool, built on two languages, one of which is Java, the selected language for the CAdES conformance-testing tools. No restrictions to its usage are imposed by the authors.
- This tool has been used in a considerable number of implementations of CAdES generation and validation tools.
- Implementations of CAdES based on this tool have proved to achieve good results in interoperability tests.
- The set of classes providing support to ASN.1 syntax, and BER and DER encoding have achieved a good level of stability with time.

However, this tool does not provide all the features required by the CAdES conformance-testing tool, namely:

- The bouncy castle framework does not ensure preservation of the encoding after a cycle of decoding-encoding, if the input ASN.1 data object is BER-encoded. This tool was conceived for managing CMS signatures, and as such is able to properly deal with the signed attributes when computing the digital signature value. However, as CMS does not mandate DER encoding for unsigned attributes and other fields not covered by the digital signature (like `signedData.certificates` and `signedData.crls` fields), the bouncy castle tool does not ensure encoding preservation of some of these fields. In CAdES signatures, this is a major drawback when one has to compute the input to the message imprint of archive time-stamps, as not only the signed attributes contribute to the message imprint computation input, but also unsigned attributes and other fields within the CMS object that are not covered by the digital signature value contribute to their message imprints.
- The bouncy castle framework does not assign identifiers to the fields after parsing an input ASN.1 data object. This tool merely parses a ASN.1 data object. For assigning names to the different fields, information on the ASN.1 specification should be inputted and additional processing is required. It is true that bouncy castle shipment includes a framework of Java classes dealing with CMS signatures, but unfortunately they do not include attributes with the names of the different fields. On its side, the CAdES conformance-testing tool work (as the work of other conformance-testing tools) is driven by a set of commands written in the XML driving instructions file; most of these commands point to the relevant fields to be checked using textual identifiers.

The CAdES conformance-testing tools, in consequence, incorporate code for:

- Preserving the encoding of any ASN.1 data object in a cycle encoding-decoding even if this data object is initially encoded in BER.
- Assigning an identifier to each field found within the CAdES signature field. These identifiers are taken from the ASN.1 definitions of: CMS, RFCs that define attributes (signed and/or unsigned) incorporated within CAdES signatures, and CAdES specification itself.

Sub-clauses below provide a high level overview on how these two features are incorporated to the CAdES conformance-testing tools.

7.1.1 Preservation of encoding

This feature is achieved by using the decorator and observer patterns. The decorator object captures each byte read from the CAdES signature under test before sending it to the ASN.1 parsing classes within bouncy castle package, and sends it an observer object that keeps it as a byte of the tag, length or value fields of the ASN.1 data object. When the conformance-testing tool completes the parsing of the CAdES signature to be tested, it has also built a tree of ASN.1 data objects with their original encodings.

7.1.2 Assigning identifiers to CAdES fields

In order to assign identifiers to CAdES fields, the CAdES conformance-testing tool uses annotations on the ASN.1 specification of CMS, CAdES and other RFCs defining attributes present in CAdES signatures.

The CAdES conformance-testing tool processes the aforementioned annotations and the ASN.1 data objects tree for assigning one name to each ASN.1 data object.

7 Specification of commands that are common to all the conformance-testing tools

7.1 Introduction

This clause presents the details of those commands that any conformance-testing tool runs for performing its tasks.

Commands in clause 7.2 are commands that perform specific checks on the values of the components under test.

Commands in clause 7.3 are commands that implement control flow (moves to a certain child, iterates on all the children, etc).

Commands in clause 7.4 are commands for managing variables.

7.2 Commands for checking contents of components

7.2.1 CheckSchemaForChildren

This command checks that the children of the signature component tested appear in the order established by the reference specification.

For doing this the command builds up a string concatenating the names of the children in their order of appearance checks if this string matches the regular expression defined by the reference specification.

Command arguments:

<regex> child element contains the regular expression defined by the reference specification

<spec> child element contains a string that represents the contents of the reference specification using the notation shown in clause 4.2.3.1 of the present document.

EXAMPLE:

```
<ExecuteCommand name="CheckSchemaForChildren" specifiedBySchema="true">
   <regex>(otherRevInfoFormat)\s*((ocspResponse)|(basicOCSPResponse))\s*</regex>
   <spec>otherRevInfoFormat (ocspResponse || basicOCSPResponse)</spec>
</ExecuteCommand>
```

The <spec> element shows that the suitable contents for the component under test are one child named `otherRevInfoFormat` and one child named either `ocspResponse` or `basicOCSPResponse`.

7.2.2 CheckNoChildrenElements

This command checks if the component under test has no children.

It has no command arguments.

7.2.3 CheckIfValueIsEqualTo

This command checks if the component under test has a certain value.

Command arguments:

<*value*> *child element* contains the value that the reference specification establishes as value of the component under test.

7.2.4 CheckIfValueIsOneOfDefined

This command checks if the value of the component under test has is one of the values present within a set of values.

Command arguments:

<*values*> *child element* contains the list of potential values as for the reference specification, separated by a comma.

7.2.5 CheckAtLeastOneOfTheFollowingIsPresent

This command checks that the component under test has one child whose name is one of a list of names passed as argument.

Command arguments:

<*children*> *child element* contains the list of names passed as argument, separated by one whitespaces character.

7.2.6 MustNotBeEmpty

This command checks that the component under test is not empty.

This command does not have command arguments.

7.2.7 PresentTextValue

This command instructs the conformance-testing tool to add to the XML raw report a special element containing the value of the component under test. This element will become part of the full report generated by the conformance-testing tool.

7.2.8 CheckIfX509Certificate

This command checks if the component under test contains a X509 certificate.

This command does not have command arguments.

7.2.9 CheckIfX509CRL

This command checks if the component under test contains a X509 CRL.

This command does not have command arguments.

7.2.10 PresentNotImplementedMessage

This command instructs the conformance-testing tool to include in the XML raw report a message notifying that the tool does not incorporate code for testing a certain component.

This command does not have command arguments.

7.3 Commands for implementing flow control

7.3.1 CursorToChild

This command moves the cursor that determines which is the component under test to the first child found whose name is equal to the one specified in a command argument. Once the cursor has been moved to the corresponding child component, it executes the commands encoded in its XML children.

It can also check the number of children whose names are equal to the name specified in the aforementioned command argument, and check if this number matches the cardinality defined in the reference specification.

Command arguments:

'cursorTo' mandatory attribute contains the name of the child where the cursor will be moved.

'noCheckInstances' optional attribute contains a boolean value. If “false” then checks if the number of children with the name indicated in argument ‘cursorTo’ matches the cardinality defined in the reference specification. If “false” then this check is not performed.

'noCheckInstances' optional attribute contains an indication of the cardinality defined in the reference specification. The notation for the cardinality is the notation used in UML class diagrams for the cardinalities in the associations between classes. This command argument has to be present if the command argument ‘noCheckInstances’ is set to “false”.

'specifiedBySchema' optional attribute contains a boolean value. It indicates whether the cardinality for the named children is defined in the reference specification by a formal syntax specification (like ASN.1 in CAdES signatures or XML Schema in XAdES signature) or not.

7.3.2 ForAllChildrenNamedAsIndicatedDo

This command iterates the cursor on any children components whose names are equal to the one specified in a command argument. Once the cursor is moved to one of the selected children components, it executes the commands encoded in its XML children.

This command can also check the number of children whose names are equal to the name specified in the aforementioned command argument, and check if this number matches the cardinality defined in the reference specification.

Command arguments:

'cursorTo' mandatory attribute contains the name of the children upon which the control will iterate.

'noCheckInstances' optional attribute contains a boolean value. If “false” then checks if the number of children with the name indicated in argument ‘cursorTo’ matches the cardinality defined in the reference specification. If “false” then this check is not performed.

'noCheckInstances' optional attribute contains an indication of the cardinality defined in the reference specification. The notation for the cardinality is the notation used in UML class diagrams for the cardinalities in the associations between classes. This command argument has to be present if the command argument ‘noCheckInstances’ is set to “false”.

'specifiedBySchema' optional attribute contains a boolean value. It indicates whether the cardinality for the named children is defined in the reference specification by a formal syntax specification (like ASN.1 in CAdES signatures or XML Schema in XAdES signature) or not.

7.3.3 ForAllTheChildrenDo

This command iterates the cursor on all the children components. Once the cursor is moved to one of the selected children components, the conformance-testing tool executes all the commands encoded in its XML children.

This command does not have any command argument.

7.3.4 CaseThis

This command checks if the name of the component under test is equal to a certain value. If this is the case, then the conformance-testing tool executes all the XML-encoded commands that are the XML children of the `CaseThis` XML element.

Command arguments:

'this' mandatory attribute contains the name that the component under test has to have for executing the XML-encoded commands that are XML children of the `CaseThis` XML element.

7.3.5 ExecuteCommandsInCaseTextValueOfChild

This command checks that the value of one of the children of the component under test is one of the members of a set of values. In case this is true, this command executes the commands indicated in its XML children.

Command arguments:

<childName> child element contains the name of the child whose value is tested.

<values> child element contains the list of potential values that the child element may have according to the reference specification, separated by a comma.

7.3.6 ExecuteCommandsGroup

This command instructs the conformance-testing tool to execute a sequence of commands that are encoded as XML elements that are children of an XML element named `CommandsGroup` whose attribute name has the same value as the argument passed to the `ExecuteCommandsGroup` command.

This [`CommandsGroup`, `ExecuteCommandsGroup`] pair builds a mechanism that simulates the definition and the invocation of a function.

Command arguments:

'name' mandatory attribute contains the name of the commands group to be executed.

7.4 Commands for managing variables

7.4.1 NewContextVar

This command creates a new global variable with a name, a type and an initial value. This variable is added to a map of global variables formed by pairs [name,variable].

Command arguments:

'varName' mandatory attribute contains the name of the variable to be created.

'varType' mandatory attribute indicates the name of the type of the variable.

'varVal' optional attribute contains the initial value of the variable to be created.

7.4.1 IfTrueDo

If the value of the variable whose name is passed as an argument to the command is “true” then this command executes all the commands indicated in its XML children.

Command arguments:

‘conditionVariable’ mandatory attribute indicates the name of the variable whose boolean value will be checked.

7.4.2 IfFalseDo

If the value of the variable whose name is passed as an argument to the command is “false” then this command executes all the commands indicated in its XML children.

Command arguments:

‘conditionVariable’ mandatory attribute indicates the name of the variable whose boolean value will be checked.

8 Specification of commands that are applicable only for CAdES conformance testing

8.1 Introduction

This clause presents the details of those commands that are specific to CAdES signatures and consequently are used only by CAdES conformance testing tool.

8.2 CheckIsBasicOCSPResponse

This command checks if the component pointed by the cursor contains an instance of `BasicOCSPResponse` type. The command manages two command arguments whose details are shown below.

Comand arguments:

'detailsTo' identifies a local variable where the details of the checked instance of `BasicOCSPResponse` will be stored for being used by o*ther commands.*

'isEncapsulated' if true indicates that the cursor points to an ASN.1 object encapsulating the BasicOCSPResponse and that its bytes are the content bytes of the aforementioned ASN.1 object; if false, it indicates that the cursor points to the BasicOCSPResponse itself and that the bytes of the BasicOCSPResponse object are the bytes of the component.

8.3 CheckIsOCSPResponse

This command checks if the component pointed by the cursor contains an instance of `OCSPResponse` type.

Comand arguments:

'detailsTo' identifies a local variable where the details of the checked instance of `OCSPResponse` type will be stored for being used by other commands.

'isEncapsulated' if true indicates that the cursor points to an ASN.1 object encapsulating the instance of `OCSPResponse` type and that its bytes are the content bytes of the aforementioned ASN.1 object; if false, it indicates that the cursor points to the instance `OCSPResponse` itself and that the bytes of the instance of `OCSPResponse` are the bytes of the component.

8.4 CheckIsOneOfTheseTypes

This command checks if the component pointed by the cursor is an instance of one of a set of types identified within the command argument.

Comand arguments:

'typeName' contains the identifiers of a set of ASN.1 types. Each name shall be separated from the rest by a whitespace character ' '.

8.5 CheckOnlyOneAttrOfTheListPresent

This command is called when the cursor points to a container of ASN.1 attributes (i.e. when it points to a `signedAttrs` or `unsignedAttrs` attributes of a CAdES signature or an RFC 3161 time-stamp token).

Comand arguments:

list of <Attribute> children elements. Each element contains the name of a specific CAdES or CMS signed or unsigned attribute.

This command is used, for instance, for checking that a certain CAdES signature contains either `essSigningCertificate` attribute or `essSigningCertificateV2`.

8.6 CountAndListAttributes

This command is invoked when the cursor points to a container of ASN.1 attributes (when it points to a `signedAttrs` or `unsignedAttrs` attributes of a CAdES signature or an RFC 3161 time-stamp token).

This command generates a map that maps the name of the attributes found with the number of occurrences of each attribute within the container. It also generates a String that contains the number of the attributes found separated by a whitespace character ' ', so that other commands may use both data.

Comand arguments:

'countIn' contains the name of a map local variable where the command will store map entries [name of the attribute, number of occurrences of the attribute], where the key of each entry is the name of the attribute and the value the number of occurrences of this attribute within the container.

'listIn' contains the name of a String local variable where the command will store the names of the attributes found separated by whitespace character ' '.

8.7 CheckAllowedAttributesAndCardinality

This command is invoked when the cursor points to a container of ASN.1 attributes (when it points to a `signedAttrs` or `unsignedAttrs` attributes of a CAdES signature or an RFC 3161 time-stamp token).

This command checks that the actual attributes present within the container are the attributes whose names appear within one of the command arguments, and that the number of occurrences of each attribute is as specified in other command arguments.

This command is used for checking presence and cardinality requirements of both signed and unsigned attributes within CAdES signatures and RFC 3161 time-stamp tokens.

Comand arguments:

'attrsCount' contains the name of a map local variable where the command will find the map generated by command CountAndListAttributes, which as mentioned above contains entries [name of the attribute, number of occurrences of

the attribute], where the key of each entry is the name of the attribute and the value the number of occurrences of this attribute within the container.

*' **attrsList**'* contains the name of a String local variable where the command will find the String generated by command CountAndListAttributes, which contains the names of the attributes found within a container separated by whitespace character ' '.

list of <Cardinality> children elements. These elements are empty. Each element has two attributes. Attribute named *'**attr**'* contains one name of a certain CAdES or CMS attribute. Attribute named *'**card**'* contains an indication of the cardinality associated to this attribute by the reference specification. The notation for the cardinality indication is the UML notation for expressing cardinalities in associations.

<spec> child element has as value a regex expression indicating what are the attributes allowed by the reference specification within this container.

EXAMPLE:

```
<ExecuteCommand name="CheckAllowedAttributesAndCardinality" specifiedBySchema="true"
attrsCount="unSignedAttributesCount" attrsList="unSignedAttributesList">
   <regex>(\s*((completeCertificateRefs\s*))|((completeRevocationRefs\s*))|((certificateV
alues\s*))|((revocationValues\s*)))*</regex>
   <spec>(completeCertificateRefs  || completeRevocationRefs  || certificateValues ||
revocationValues)*</spec>
   <Cardinality attr="completeCertificateRefs" card="0..1"></Cardinality>
   <Cardinality attr="completeRevocationRefs" card="0..1"></Cardinality>
   <Cardinality attr="certificateValues" card="0..1"></Cardinality>
   <Cardinality attr="revocationValues" card="0..1"></Cardinality>
</ExecuteCommand>
```

In this example the children may have as name `completeCertificateRefs`, `completeRevocationRefs`, `certificateValues`, or `revocationValues`. Any of them may be absent, but if present only one instance of each is allowed.

8.8 IsComponentOfTimeStampToken

This command checks if the component pointed by the cursor is a component of one of the attributes that encapsulate time-stamp tokens. If this is the case it sets to true a Boolean local variable, otherwise it sets it to false.

Comand arguments:

*'**var**'* contains the name of a Boolean local variable that the command sets to true if the component pointed by the cursor is a component of one of the attributes that encapsulate time-stamp tokens, or to false if this is not the case.

The value of this variable is used by other commands for performing or not certain checks depending on whether certain components pertain or not to one of the attributes that encapsulate time-stamp tokens.

8.9 IsComponentOfArchiveTimeStampV3

This command checks if the component pointed by the cursor is a component of an `archive-time-stamp-v3` attribute. If this is the case it sets to true a Boolean local variable, otherwise it sets it to false.

Comand arguments:

*'**var**'* contains the name of a Boolean local variable that the command sets to true if the component pointed by the cursor is a component of an `archive-time-stamp-v3` attribute, or to false if this is not the case.

The value of this variable is used by other commands for performing or not certain checks depending on whether certain components pertain or not to `archive-time-stamp-v3` attributes.

8.10 IsComponentOfRootCMS

This command checks if the component pointed by the cursor is a component of a CMS structure corresponding to a CMS or to a CAdES signature (by opposition to components that appertain to `archive-time-stamp-v3` attributes or to any of the attributes encapsulating time-stamp tokens). If this is the case it sets to true a Boolean local variable, otherwise it sets it to false.

Comand arguments:

'var' contains the name of a Boolean local variable that the command sets to true if the component pointed by the cursor is a component of a CMS structure corresponding to a CMS or to a CAdES signature, or to false if this is not the case.

The value of this variable is used by other commands for performing or not certain checks depending on whether certain components pertain or not a CMS structure corresponding to a CMS or to a CAdES signature.

8.11 KeepComponentIn

This command takes the value of the component pointed by the cursor and adds this value to a list local variable passed as command argument.

Comand arguments:

'where' contains the name of a List local variable where the command adds the value of the component pointed by the cursor.

The value added to this list can be used by other commands.

8.12 CheckIfCAdESBaselineAttributes

This command checks if the set of signed and unsigned attributes is correct according to CAdES baseline signatures specification.

8.13 CheckIfIsCAdES

This command checks if the parallel signatures within a CAdES structure are CAdES signatures. For each parallel signature the command:

- checks that it incorporates the mandatory signed attributes according to ETSI EN 319 122 Part 2, including one of the attributes that contain a reference of the signing certificate.
- Cryptographically verifies its digital signature value.

This command also cryptographically verifies the digital signature value for each countersignature incorporated in any of the parallel signatures.

This command does not check whether the countersignatures within a CAdES CMS structure are CAdES or not.

8.14 CheckIfIsCAdESAndBaselineEN319122_1

This command checks if the parallel signatures within a CAdES structure contain the set of attributes allowed by ETSI EN 319 122 Part 1 for CAdES baseline signatures. For each parallel signature the command:

- checks that it incorporates the mandatory signed attributes for CAdES baseline signatures by ETSI EN 319 122 Part 1, including one of the attributes that contain a reference of the signing certificate.
- Cryptographically verifies its digital signature value.

This command also cryptographically verifies the digital signature value for each countersignature incorporated in any of the parallel signatures.

This command does not check whether the countersignatures within a CAdES CMS structure are CAdES baseline signature or not.

8.15 CheckIfIsDigestOfPKIDataInSignedData

This command is executed when the cursor points to either one of the `certificateHashIndex` or one of the `crlHashIndex` fields that are grandchildren of an `ats-hash-index-v3` attribute incorporated into an `archive-time-stamp-v3` attribute.

If the cursor points to one `certificateHashIndex` component, this command checks if the value of the component is the digest value of one of the certificates present in the `certs` field of the CAdES signature incorporating the `archive-time-stamp-v3`.

If the cursor points to one `crlHashIndex` component, this command checks if the value of the component is the digest value of one of the revocation present in the `crls` field of the CAdES signature incorporating the `archive-time-stamp-v3`.

Comand arguments:

'digests2PKIData' contains the name of a map local variable. This map is actually a map of maps, because it contains entries [String, Map]. The key of each entry is a String identifying a digest algorithm. The value of each entry is another Map containing entries [ByteArray, PKIObjectBytesAndDetails]. The key of each entry is an object encapsulating a byte array that contains the digest value of a certain PKI object (X509 certificate,X509 CRL, etc) computed with the digest algorithm indicated in the key of the previous map. The value of each entry is an object that encapsulates a byte array resulting from encoding the PKI object and a String reporting certain details of the PKI object (in the case of X509 details like issuer, subject, serial number, etc). If this variable does not exist, the command creates it. This map stores digest values on PKI objects computed by the command that do not match the PKI object encapsulated by the component pointed by the cursor. This avoids to repeat the computation when this command is invoked and the cursor is pointing to another component.

'sigDataPKIData' contains the name of a list local variable. This list contains instances of class PKIObjectBytesAndDetails, each one encapsulating the byte array resulting of encoding a certain PKI object, and a String reporting certain details of the aforementioned PK object.

8.16 CheckIfIsDigestOfUnsignedAttrs

This command is executed when the cursor points to one of the `unsignedAttrHashIndex` field that is grandchild of an `ats-hash-index-v2` attribute incorporated into an `archive-time-stamp-v3` attribute.

This command checks if the value of the component is the digest value of one of the unsigned attributes present in the `unsignedAttrs` field of the CAdES signature incorporating the `archive-time-stamp-v3`.

Comand arguments:

'unsAttrsVar' contains the name of a list local variable. This list is a list of maps. Each map in the list is actually a map of maps, because it contains entries [String, Map]. The key of each entry is a String identifying a digest algorithm. The value of each entry is another Map containing entries [ByteArray, PKIObjectBytesAndDetails]. The key of each entry is an object encapsulating a byte array that contains the digest value of a certain PKI object (X509 certificate,X509 CRL, etc) computed with the digest algorithm indicated in the key of the previous map. The value of each entry is an object that encapsulates a byte array resulting from encoding the PKI object and a String reporting certain details of the PKI object (in the case of X509 details like issuer, subject, serial number, etc). If this variable does not exist, the command creates it. This map stores digest values on PKI objects computed by the command that do not match the PKI object encapsulated by the component pointed by the cursor. This avoids to repeat the computation when this command is invoked and the cursor is pointing to another component.

'unsAttrs' contains the name of a list local variable. This list contains instances of class PKIObjectBytesAndDetails, each one encapsulating the byte array resulting of encoding one unsigned attribute of the CAdES signature, and a String reporting certain details of the aforementioned unsigned attribute.

Although the ETSI TS 119 122 Part 1 forbids using the `ats-hash-index-v2` attribute, this command has been left within the source for being able to apply checks to signatures that were generated with this attribute.

8.17 CheckIfIsDigestOfUnsignedAttrsValuesInstances

This command is executed when the cursor points to one of the `unsignedAttrHashIndex` field that is grandchild of an `ats-hash-index-v3` attribute incorporated into an `archive-time-stamp-v3` attribute.

This command checks if the value of the component is the digest value of one of the unsigned attributes present in the `unsignedAttrs` field of the CAdES signature incorporating the `archive-time-stamp-v3`.

Comand arguments:

'unsAttrsVar' contains the name of a list local variable. This list is a list of maps. Each map in the list is actually a map of maps, because it contains entries [String, Map]. The key of each entry is a String identifying a digest algorithm. The value of each entry is another Map containing entries [ByteArray, PKIObjectBytesAndDetails]. The key of each entry is an object encapsulating a byte array that contains the digest value of a certain PKI object (X509 certificate,X509 CRL, etc) computed with the digest algorithm indicated in the key of the previous map. The value of each entry is an object that encapsulates a byte array resulting from encoding the PKI object and a String reporting certain details of the PKI object (in the case of X509 details like issuer, subject, serial number, etc). If this variable does not exist, the command creates it. This map stores digest values on PKI objects computed by the command that do not match the PKI object encapsulated by the component pointed by the cursor. This avoids to repeat the computation when this command is invoked and the cursor is pointing to another component.

'unsAttrs' contains the name of a list local variable. This list contains instances of class PKIObjectBytesAndDetails, each one encapsulating the byte array resulting of encoding one of the instances of AttributeValue type within one unsigned attribute of the CAdES signature, and a String reporting certain details of the aforementioned instance.

8.18 CheckIfX509AttributeCertificate

This command checks if the component pointed by the cursor is a X509 attribute certificate.

8.19 CheckIsAGivenType

This command checks if the component pointed by the cursor is an ASN.1 object of the type indicated by the command argumnent.

Comand arguments:

'typeName' contains the name of an ASN.1 type. The following list shows the possible values for this argument: "OID", "BOOLEAN", "INTEGER", "REAL", "UTF8String", "NumericString", "PrintableString", "TeletexString", "T61String", "VideotexString", "IA5String", "UTCTime", "GeneralizedTime", "GraphicString", "VisibleString", "IO646String", "GeneralString", "UniversalString", "CHARACTERSTRING", "BMPString","OCTET STRING".