XAdES baseline and XAdES extended signatures
Conformance Testing Tools Documentation

Version 1.0 June 2016

Abstract

This document provides a high level overview of the XAdES baseline signatures and extended XAdES 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 XAdES remote Plugtests™ event started on 1st October 2015, organized and supported by ETSI CTI (Centre for Testing and Interoperability).
Contents

1 Introduction .................................................................................................................. 4
2 References .................................................................................................................. 4
3 XAdES conformance testing tool user guide ............................................................... 4
  3.1 Running the tool ..................................................................................................... 4
  3.2 Results generated .................................................................................................... 5
4 Contents of the reports generated by XAdESCC ....................................................... 12
  4.1 Introduction ............................................................................................................. 12
  4.2 Full report contents ............................................................................................... 12
    4.2.1 Introduction ........................................................................................................ 12
    4.2.2 Structure of the table ....................................................................................... 12
    4.2.3 Contents common to all the conformance-testing tools .................................... 13
      4.2.3.1 CheckSchemaForChildren test ................................................................. 13
      4.2.3.2 CheckNoChildrenElements test ............................................................... 14
      4.2.3.3 CheckIfValueIsEqualTo test .................................................................... 14
      4.2.3.4 CheckIfValueIsOneOfDefined test ............................................................ 14
      4.2.3.5 ExecuteCommandsInCaseTextValueOfChild test .................................... 15
      4.2.3.6 MustNotBeEmpty test .............................................................................. 15
      4.2.3.7 PresentTextValue indication ...................................................................... 15
      4.2.3.8 CheckIfSigningCertsIsReferenced test ...................................................... 15
      4.2.3.9 CheckIfSignsTheRightSignatureValue test .............................................. 15
      4.2.3.10 MustBeBase64 test .................................................................................. 15
      4.2.3.11 CheckIfValueIsBoolean test ................................................................... 16
      4.2.3.12 CheckIfValueIsEmpty test ..................................................................... 16
      4.2.3.13 MustBeURI test ....................................................................................... 16
      4.2.3.14 CheckPKIDataRefAgainstPKIDataValuesIfRequired test ....................... 16
      4.2.3.15 CheckIfMsgImpIsAsInSpec test ............................................................... 17
  4.2.4 Contents specific to XAdES conformance testing ............................................... 24
    4.2.4.1 Introduction .................................................................................................. 24
    4.2.4.2 CheckIfsXAdESSignature test .................................................................... 24
    4.2.4.3 CheckDigestValueInReferenceWithinSignedManifest test ....................... 25
    4.2.4.4 CheckEncapsulatesOCSPValueAndAddToVariable test ............................. 25
    4.2.4.5 CheckEncapsulatesX509CertificateAndAddToVariable test ................... 25
    4.2.4.6 CheckEncapsulatesX509CRLAndAddToVariable test ............................... 26
    4.2.4.7 CheckIDOFFPointsToReferenceToSignedObject test .............................. 26
    4.2.4.8 CheckIDOFFsPointToAllReferencesToSignedObjects test ....................... 27
    4.2.4.9 CheckIfSignerCertIsInKeyInfo test ............................................................. 27
    4.2.4.10 CheckIfValueIsURIWithoutNoFragmentPart test ..................................... 27
    4.2.4.11 CheckIfValueXMLDateTime test ............................................................... 27
    4.2.4.12 CheckNoAttributes test ............................................................................. 27
    4.2.4.14 CheckPointsTodsReference test ................................................................. 28
    4.2.4.15 CheckSchemaForAttributes test ............................................................... 28
    4.2.4.16 CheckSignedInfoReference test ................................................................. 28
    4.2.4.17 DecodeAndShowIssuerSerialInFullReport test ....................................... 28
    4.2.4.18 IsNotDistributedXAdES test ................................................................... 29
    4.2.4.19 TestIfPointsToDOTSTOrADOTST test ................................................. 29
5 High level overview of the conformance-testing tools software components ................. 29
  5.1 Basic conformance-testing tools’ conceptual model ................................................ 29
  5.2 Commands and XML driving instructions file ......................................................... 32
  5.3 Overview of the operation of the testing tools ......................................................... 33
6 Specification of commands that are common to all the conformance-testing tools ............ 34
  6.1 Introduction ............................................................................................................. 34
  6.2 Commands for checking contents of components .................................................. 34
    6.2.1 CheckSchemaForChildren .............................................................................. 34
6.2.2 CheckNoChildrenElements ................................................................. 34
6.2.3 CheckIfValueIsEqualTo ................................................................. 34
6.2.4 CheckIfValueIsOneOfDefined ......................................................... 35
6.2.5 CheckAtLeastOneOfTheFollowingIsPresent ....................................... 35
6.2.6 MustNotBeEmpty ........................................................................... 35
6.2.7 PresentTextValue ........................................................................... 35
6.2.8 PresentNotImplementedMessage ...................................................... 35
6.2.9 CheckIfSomeChildrenPresent .......................................................... 35
6.2.10 CheckIfValueIsBase64 .................................................................... 36
6.2.11 CheckIfValueIsBoolean ................................................................. 36
6.2.12 CheckIfValueIsEmpty ..................................................................... 36
6.3 Commands for implementing flow control ............................................. 36
6.3.1 CursorToChild ................................................................................. 36
6.3.2 ForAllChildrenNamedAsIndicatedDo ............................................... 37
6.3.3 ForAllTheChildrenDo ...................................................................... 37
6.3.4 CaseThis ......................................................................................... 37
6.3.5 CaseAncestor .................................................................................. 37
6.3.6 ExecuteCommandsInCaseTextValueOfChild .................................... 38
6.3.7 ExecuteCommandsGroup .................................................................. 38
6.4 Commands for managing variables ....................................................... 38
6.4.1 NewContextVar .............................................................................. 38
6.4.2 IfTrueDo ......................................................................................... 38
6.4.2 IfFalseDo ........................................................................................ 38

7 Specification of commands that are applicable only for XAdES conformance testing .......................... 39
7.1 Introduction ....................................................................................... 39
7.2 CheckIfIsXAdESSignature ................................................................. 39
7.3 CheckDigestValueInReferenceWithinSignedManifest ................................ 39
7.4 CheckEncapsulatesOCSPValueAndAddToVariable ................................ 39
7.5 CheckEncapsulatesX509CertificateAndAddToVariable ......................... 39
7.6 CheckEncapsulatesX509CRLAndAddToVariable ................................... 40
7.7 CheckIfDOFPointsToReferenceToSignedObject .................................... 40
7.8 CheckIfDOFPointsToAllReferencesToSignedObjects ......................... 40
7.9 CheckIfSignerCertificateInKeyInfo ..................................................... 40
7.10 CheckIfValuesURIWithoutNoFragmentPart ....................................... 40
7.11 CheckIfValuesXMLDateTime ............................................................. 40
7.12 CheckNoAttributes .......................................................................... 41
7.14 CheckPointsTodsReference .............................................................. 41
7.15 CheckSchemaForAttributes ............................................................... 41
7.16 CheckSignedInfoReference ............................................................... 41
7.17 DecodeAndShowIssuerSerialInFullReport ......................................... 41
7.18 IsNotDistributedXAdES .................................................................... 41
7.19 GetExCanIDOTSTsAndADOTSTs ......................................................... 41
7.20 TestIfPointsToDOTSTorDOTST ................................................................ 42
7.21 CursorToXMLAttribute .................................................................... 42
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

Previously, a conformance-testing tool was developed for checking conformance of XAdES signatures against ETSI TS 103 171: “XAdES Baseline Profile”. This tool has been taken as the starting point for the development of the rest of conformance-testing tools mentioned above. This tool has been developed in Java language.

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


2 References


3 XAdES conformance testing tool user guide

3.1 Running the tool

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

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

```
java -jar XAdESConformanceChecker -in <inputFile> -outFolder <outputFolder> -testSpec <referenceXAdESSpecification>
```

Where

- `<inputFile>` contains the pathname of the file containing the XAdES 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.
<referenceXAdESSpecification> identifies the ETSI specification against which the XAdES structure has to be tested. At present the tool admits the following values:


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 XAdES conformance-testing tool when a XML file contains one XAdES signature.

![Figure 1: An example of contents of output folder generated by XAdES conformance testing tool](image)

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 XAdES signatures (one per each instance of SignerInfo type) that a XAdES structure may contain. If only one XAdES 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 XAdES signatures found within the XAdES 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 XAdES signatures within the XAdES 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 XAdES conformance-testing tools as explained below. Figure 3 Shows an example of what is shown by the web browser after the link “XAdES-signature-1” is selected in the page shown in Figure 2.
**XAdES Conformance Checker for ETSI pre EN 319 132 Part 1 v1.1.1: "XAdES signatures. Building blocks and XAdES baseline signatures".**

This tool tests conformance of XAdES signatures against N 319 132 Part 1 v1.01.1: "XAdES signatures. Building blocks and XAdES baseline signatures".

This tool has been designed and implemented by Juan Carlos Cruellas (Universidad Politécnica de Cataluña).

The first version of the tool was produced as part of the tasks performed by the ETSI STF-428: "Quick fixes to testing of electronic signatures standards" (SA/ETSI/ENTR/460/2010-13), funded by the European Commission.

This update has been carried out as part of the tasks performed by the ETSI STF-459: "on Rationalised Framework for electronic signatures standards; Priority Activities relating to Testing Compliance & Interoperability and Trust Applications Service Provider" (SA/ETSI/ENTR/460/2012-10 - Phase 2c Part B), funded by the European Commission.

STF-428 and STF-459 were part of the work carried out under the Mandate M/460 for defining and implementing a Rationalised Framework for European Standards of Electronic Signatures.

More details on the team, mission and performed work by the STF-459 may be found at the following URL: [http://portal.etsi.org/STF/STF_HomePages/STF459/STF459.asp](http://portal.etsi.org/STF/STF_HomePages/STF459/STF459.asp)

More details on the team, mission and performed work by the STF-428 may be found at the following URL: [http://portal.etsi.org/STF/STF_HomePages/STF428/STF428.asp](http://portal.etsi.org/STF/STF_HomePages/STF428/STF428.asp)

For any comment and/or suggestion on the tool, please contact Juan Carlos Cruellas at the following email address: cruellas@ac.upc.edu.

---

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

5) **One or more folders with name "XAdES-signature-"<i>i</i>**: (i has integer values starting in 1): One folder per each per each ds:Signature element representing a signature found within the input XML file, except those ones that are children of a xades:CounterSignature element (countersignatures). 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 XAdES signatures). Its contents appear within the "Signature Reports view" frame once the user selects one 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 XAdES 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 XAdES signature. In this figure and in all the subsequent ones some of the data...
identifying the generator of the signature or the issuer of a certificate have been occulted or changed for preserving their anonymity whenever it has been required.

![Diagram](image.png)

**Figure 4:** View of file "xmlrawoutputashtml.xml" corresponding to one XAdES 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.
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 XAdES conformance testing tool has not detected any errors and warnings. Figure 6 shows the errors reported by the XAdEESCC when checking a XAdES signature conformant with ETSI TS 101 903 against the ETSI EN 319 132 Part 1. It may be seen that the tool reports that the signature does not contain the xades:SigningCertificateV2.
Report on errors, warnings and exceptions

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

<table>
<thead>
<tr>
<th>Result</th>
<th>Code</th>
<th>Test Result Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Error</td>
<td>Tool Error</td>
<td>Location: (CodeTest) - KeyInfo/X509Data(1)- (CheckIfSignedCertIsReferenced) The signature does not incorporate the proper signed property referencing the signing certificate.</td>
</tr>
<tr>
<td>2. Error</td>
<td>Tool Error</td>
<td>Location: (CodeTest) - KeyInfo/X509Data(1)- (CheckIfSignedRevocation) The signature is not a XAdES signature. Neither the SigningCertificateV2 property references the signer's certificate AND SignedProperties is covered by the signature itself NOR (the signer's cert is within ds:KeyInfo AND it is signed by the signature itself)</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>Element</th>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid.issuer</td>
<td>C=FR,O=ETSI,OU=Plugtests_STF-428_2011-2012,CN=RootCAOK</td>
<td></td>
</tr>
<tr>
<td>sid.SerialNumber</td>
<td>107689462004955</td>
<td></td>
</tr>
<tr>
<td>gentime</td>
<td>Mon 26 13:50:15 CET 2012</td>
<td></td>
</tr>
<tr>
<td>accuracy.seconds</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>accuracy.milliseconds</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ordering</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>policy</td>
<td>1.3.6.1.4.1.2706.3.2.5.2.1.1</td>
<td></td>
</tr>
<tr>
<td>serialNumber</td>
<td>13327626150174110</td>
<td></td>
</tr>
<tr>
<td>nonce</td>
<td>1288903612923056977</td>
<td></td>
</tr>
</tbody>
</table>

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

- **File "mssgimprintsdetails.html"**: This is a document that provides, for each time-stamp token found in the XAdES 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.
4 Contents of the reports generated by XAdESCC.

4.1 Introduction

This clause provides a detailed explanation of the reports that XAdESCC may generate when checking a certain XAdES 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 XAdESCC.

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>
<thead>
<tr>
<th>Result</th>
<th>TI/VI</th>
<th>Tested Element and Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Test Result details</td>
</tr>
</tbody>
</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 XML elements that must be visited in the signature tree structure for going from the root element of the signature to the checked component. Components of the path name are separated by "/". When there are several sibling components with the same name, then integer indexes are used. XML attributes are indicated by inserting the add character '@' before their names (@Algorithm indicates the XML attribute whose name is Attribute).

Example:

UnsignedSignatureProperties/TimeStampValidationData[1]/RevocationValues[1]/OCSPValues/EncapsulatedOCSPValue[1], indicates that the tested component is the first EncapsulatedOCSPValue child of OCSPValues, child of the first RevocationValues, child of the first TimeStampValidationData, child of UnsignedSignatureProperties container.

- 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.
EXAMPLE:

<table>
<thead>
<tr>
<th>Location-[CodeTest]:SignedProperties-[CheckSchemaForChildren]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Children order and number DO NOT MATCH specification</td>
</tr>
<tr>
<td>Elements found: xadesv132:SignedSignatureProperties</td>
</tr>
<tr>
<td>Error indication (^ appears at the end of the last correct child): xadesv132:SignedSignatureProperties^</td>
</tr>
</tbody>
</table>

These two rows indicate that the test has checked if the contents of the xades:SignedProperties are conformant to the XML Schema 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 XML Schema specification and 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).

EXAMPLE: }

| Location-[CodeTest]:SignedSignatureProperties/SigningTime-[CheckNoChildrenElements] |

### 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.

### 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 to the reference specification and the value of the tested component.

EXAMPLE:

<table>
<thead>
<tr>
<th>Location-[CodeTest]:UnsignedSignatureProperties/SignatureTimeStamp[1]/CanonicalizationMethod/@Algorithm-[CheckIfValueIsOneOfDefined]</th>
</tr>
</thead>
</table>
### 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 to the reference specification and the value of the tested component.

### 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:**

<table>
<thead>
<tr>
<th>Location</th>
<th>CodeTest:UnsignedSignatureProperties/TimeStampValidationData[1]-[MustNotBeEmpty]</th>
</tr>
</thead>
</table>

### 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.

### 4.2.3.8 CheckIfSigningCertIsReferenced test

This entry of the table reports on a test that checks that the signing certificate is referenced by a XAdES signed qualifying property that contains references to X509 certificates (xades:SigningCertificateV2 in ETSI EN 319 132 or xades:SigningCertificate in ETSI TS 101 903). The additional information row shows either the position of the signing certificate if the check has succeeded or the reasons for the check failure.

**EXAMPLE:**

<table>
<thead>
<tr>
<th>Location</th>
<th>CodeTest:KeyInfo/X509Data[1]/X509Certificate[1]-[CheckIfSigningCertIsReferenced]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digest</td>
<td>found in Cert element number 1</td>
</tr>
</tbody>
</table>

### 4.2.3.9 CheckIfSignsTheRightSignatureValue test

This entry of the table reports on a test that checks that the countersignature actually signs the \( \text{ds:SignatureValue} \) element of the countersigned XAdES signature.

**EXAMPLE:**

<table>
<thead>
<tr>
<th>Location</th>
<th>CodeTest:UnsignedSignatureProperties/CounterSignature[1]/</th>
</tr>
</thead>
<tbody>
<tr>
<td>The countersignature number: 1 actually signs its enveloping XAdES's ( \text{ds:SignatureValue} ) element through its ( \text{ds:Reference} ) number 1</td>
<td></td>
</tr>
</tbody>
</table>

### 4.2.3.10 MustBeBase64 test

This entry of the table reports on a test that checks if the content of the component under test is the result of base-64 encoding some binary data. The additional information row does not appear.

**EXAMPLE:**
4.2.3.11 CheckIfValuesIsBoolean test
This entry of the table reports on a test that checks if the content of the component under test is a Boolean value.

EXAMPLE:

```
Location-{CodeTest}:SignedInfo/Reference[1]/DigestValue-{MustBeBase64}
```

4.2.3.12 CheckIfValuesIsEmpty test
This entry of the table reports on a test that checks if the value of the component under test is an empty value (an empty XML element in XAdES is the xades: AllSignedDataObjects component of the xades:CommitmentTypeIndication qualifying property).

4.2.3.13 MustBeURI test
This entry of the table reports on a test that checks if the content of the component under test is a valid URI reference.

EXAMPLE:

```
Location-{CodeTest}:UnsignedSignatureProperties/SignatureTimeStamp[1]/CanonicalizationMethod/@Algorithm-{MustBeUri}
Value found: 'http://www.w3.org/2001/10/xml-exc-c14n#'
```

4.2.3.14 CheckPKIDataRefAgainstPKIDataValuesIfRequired test
This entry of the table reports on a test that checks, when required by the specification, whether a certain XAdES qualifying property whose goal is to reference a certain PKI data value (X509 certificate, X509 attribute certificate, OCSP response or CRL), actually references one of the PKI data values incorporated into the signature. The additional information row provides details of the referenced PKI data.

EXAMPLE:

```
CERTIFICATE digest references certificate in CertificateValuesX509Certs. Details: Issuer: CN=RootCAOK, OU=Plugtests_STF-428_2011-2012, O=ETSI, C=FR; Serial Number: 6536134721675829178; Subject: CN=LevelACAOK, OU=Plugtests_STF-428_2011-2012, O=ETSI, C=FR; Public Key's SHA1 digest: 'c3c3baebb782e3f538b9be84831940ae7c47253
```

---

**ETSI**
4.2.3.15 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: UnsignedSignatureProperties/SignatureTimeStamp[1]/EncapsulatedTimeStamp[1]-CheckIfMssgImpIsAsInSpec |
|-----------|------------------------------------------------------------|
| Both the message imprint found in the time-stamp token and the computed one by the tool, have the same value: | 12cc0ffa6ccdac1bc48bac2748c61a0402911c82 |

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:**

<table>
<thead>
<tr>
<th>Trace Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element(Contribution)</td>
</tr>
</tbody>
</table>
ETSI XAdES baseline and extended XAdES signatures
Conformance Testing Tool Documentation

Trace Details

Element(Contribution) Contribution

T0NTUD9J YT1MZXXlbeJDQU9LMcGuAiUdEOQgMB6BHgIvZXouYmVubWJcmt
hQGInYxB02x2vXj9M20wHwYDFR0jBBgwFoAU+kUiOu+axqxoNbaXa6e9epgzk
DQYJ1KOzIvhcNAQFEBADqgMA39GW0sOlVfBhG5UQeowA7335xPH14Q1O
vZ1xKvC+VBlW5hd7cZDZJiDCEMAD3pmu2/IL0oRK+CznOYm4Bwl6fmmHuYBiiss
Zc91YmIEIHghuzeg1P7mC9jZiMn4fWbkbVwpV5v5FrgFavIT0UKL87q5Hg44ZAA18MC
GCaFxBDbNiXgQaAqQhmcUKgcOJj82wkurl6NjcfLeJrObvAsAtIO9cp90VWPWfEg
Dzrm+4/7TClybFyDHQTjAI2WQA1qXgqruT4fxcQYsLbLO7lX/KXZSRZ7hZMNzy
du+lIsaY2vs6yR4CRStx4aclNLGqMzID93dQfPQ43WRSI="/d:x:509Certificate="/d:

E/sCAQMxCzAI89UqDgB1AgMCAGA1UdEQQgMB6BHgIvZXouYmVubWJcmt

ArchiveTimeStamp[4]/EncapsulatedTimeStamp(5)

<xs:X509Data><ds:KeyInfo>
</xs:X509Certificate></d:

ArchiveTimeStamp[4]/EncapsulatedTimeStamp(5)
### Trace Details

<table>
<thead>
<tr>
<th>Element(Contribution)</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>3Rlc3RzLm5ldC9cw90ZWQyY2Fw28v7Y3Jscy9TQ09L1Jv9b3RDQL9LLNm</td>
<td>bDCASGgCCsGAQUFBwEBBIBGgCCARYwdQYIKwYBBQHMAKGAwRkYXAnLj9y4WYRl3Nygd5w5hVd0hVdmhvaV0LN0P5jv3bRDQQU9L9PPV9Wd0ZXNC919TVEYND4XZwMTM5AmMxPPU9UV0kaQz1GJ9jQUCnRzPzmzJXRR102JphmFvTFRBggrBEBFQwAoZFaHR0cDvL3hihZGVzc3J2LnsdWd0dZ5XN0c5 uZQXvCvhJvGvJvGkL2NkHNC9L2NcRz1NDT0vUs9v9ENB10toY2VyE0MoCSgAQUFBwEBB1j5oHHR0i8veGFkZXNwYcYkGZ3Rlc3RlzLm5d9cw90ZWQyY2Fw28v7Y3Jscy9TQ09L1Jv9b3RDQL9LLNm</td>
</tr>
</tbody>
</table>
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) Row 1 of the table shows the contribution from the result of processing the ds:Reference element referencing the signed document.
2) Row 2 shows the contribution from ds:SignedInfo element.
3) Row 3 shows the contribution from ds:SignatureValue element.
4) Row 4 shows the contribution from ds:KeyInfo element.
5) Row 5 shows the contribution from xades:SignatureTimeStamp element.
6) Row 6 shows the contribution from xades:CertificateValues element.
7) Row 7 shows the contribution from xades:RevocationValues element.

4.2.4 Contents specific to XAdES 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 XAdES signature. All these tests can appear within a full report generated by the XAdES 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 CheckIfIsXAdESSignature test

This component indicates whether the signature under test is a XAdES signature as specified by ETSI TS 319 132 Part 1. More specifically it checks that either:

a) The signature incorporates the xades:SigningCertificateV2 qualifying property and that it references the signer's certificate AND that the xades:SignedProperties container is covered by the signature itself OR

b) the signer's cert is within ds:KeyInfo AND it is signed by the signature itself.

EXAMPLE:

```
Location-{CodeTest}::{CheckIfIsXAdESSignature}
```

The signature is a XAdES signature. The SigningCertificateV2 property references the signer's certificate AND SignedProperties is signed by the signature itself.
4.2.4.3 CheckDigestValueInReferenceWithinSignedManifest test

This entry of the table reports on a test that checks the correctness content of \texttt{ds:DigestValue} child element of a \texttt{ds:Reference} child element of a signed \texttt{ds:Manifest} element. The XAdES conformance-testing tool processes the \texttt{ds:Reference} following the processing model specified in XMLDSIG [3] and compares the obtained value with the value present within the XAdES signature.

**EXAMPLE:**

<table>
<thead>
<tr>
<th>Location</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CodeTest:Object[2]/Manifest[1]/Reference[2]-[CheckDigestValueInReferenceWithinSignedManifest]</td>
<td></td>
</tr>
</tbody>
</table>

Computed digest value is the same as the present digest value: \texttt{'beEzfWukV/qSGGQpwZd4VkJpwhQz'}

4.2.4.4 CheckEncapsulatesOCSPValueAndAddToVariable test

This entry of the table reports on a test that checks that the XML element under test encapsulates an OCSP value.

**EXAMPLE:**

<table>
<thead>
<tr>
<th>Location</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CodeTest:UnsignedSignatureProperties/RevocationValues[1]/OCSPValues/EncapsulatedOCSPValue[2]-[CheckEncapsulatesOCSPValueAndAddToVariable]</td>
<td></td>
</tr>
</tbody>
</table>

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

**EXAMPLE OF CONTENTS GENERATED FOR THE CONTENT DETAILS REPORT:**

<table>
<thead>
<tr>
<th>xadesv141:TimeStampValidationData[1]/RevocationValues[1]/OCSPValues/EncapsulatedOCSPValue[1]</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>responderID</td>
<td>C=FR,O=ETSI,OU=Plugtests_2013-2014,CN=LevelBCAOK</td>
</tr>
<tr>
<td>producedAt</td>
<td>Mon Dec 02 08:58:44 CET 2013</td>
</tr>
<tr>
<td>resp[1].certID.serialNumber</td>
<td>428364045964074</td>
</tr>
<tr>
<td>resp[1].certID.hashAlgorithm</td>
<td>1.3.14.3.2.26</td>
</tr>
<tr>
<td>resp[1].certID.issuerNameHash</td>
<td>459153f43a314cdf6e74a189e5c8a3f828a0f1e</td>
</tr>
</tbody>
</table>

4.2.4.5 CheckEncapsulatesX509CertificateAndAddToVariable test

This entry of the table reports on a test that checks that the XML element under test encapsulates an X509 certificate value.

**EXAMPLE:**

<table>
<thead>
<tr>
<th>Location</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CodeTest:KeyInfo/X509Data[1]/X509Certificate[1]-[CheckEncapsulatesX509CertificateAndAddToVariable]</td>
<td></td>
</tr>
</tbody>
</table>

This component encapsulates a X509 certificate.
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:**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SerialNumber</td>
<td>3342904847548966778</td>
</tr>
<tr>
<td></td>
<td>Subject</td>
<td>C=FR, O=ETSI, OU=Plugtests_STF-428_2011-2012, CN=RootCA2OK</td>
</tr>
<tr>
<td></td>
<td>NotBefore</td>
<td>Tue Nov 08 16:43:35 CET 2011</td>
</tr>
<tr>
<td></td>
<td>NotAfter</td>
<td>Sat Nov 08 16:43:35 CET 2014</td>
</tr>
</tbody>
</table>

### 4.2.4.6 CheckEncapsulatesX509CRLAndAddToVariable test

This entry of the table reports on a test that checks that the XML element under test encapsulates an X509 CRL value.

**EXAMPLE:**

```
Location-{CodeTest}:UnsignedSignatureProperties/RevocationValues[1]/CRLValues/EncapsulatedCRLValue[1]/{CheckEncapsulatesX509CRLAndAddToVariable}
```

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:**

<table>
<thead>
<tr>
<th>RevocationValues[3]/CRLValues/EncapsulatedCRLValue[1]</th>
<th>version</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Issuer</td>
<td>CN=RootCAOK, OU=Plugtests_STF-428_2011-2012, O=ETSI, C=FR</td>
</tr>
<tr>
<td></td>
<td>thisUpdate</td>
<td>Tue Mar 13 17:58:28 CET 2012</td>
</tr>
<tr>
<td></td>
<td>nextUpdate</td>
<td>Thu Apr 12 18:58:28 CEST 2012</td>
</tr>
</tbody>
</table>

### 4.2.4.7 CheckIfDOFPointsToReferenceToSignedObject test

This entry of the table reports on a test that checks that the `xades:DataObjectFormat` qualifying property points to one `ds:Reference` that references a signed data object.

**EXAMPLE:**

```
Location-{CodeTest}:SignedDataObjectProperties/DataObjectFormat[1]//@ObjectReference-{CheckIfDOFPointsToReferenceToSignedObject}
```

The `xades:DataObjectFormat` points to one `ds:Reference` referencing a signed data object. Value found for attribute...
4.2.4.8 CheckIfDOFsPointToAllReferencesToSignedObjects test

This entry of the table reports on a test that checks that all the `xades:DataObjectFormat` qualifying properties incorporated into the XAdES signature point to the `ds:Reference` elements that reference all the signed data objects.

**EXAMPLE:**

<table>
<thead>
<tr>
<th>Location</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{CodeTest}</code>:SignedDataObjectProperties-{CheckIfDOFsPointToAllReferencesToSignedObjects}</td>
<td>All the <code>ds:Reference</code> elements referencing a signed data object are pointed by one <code>xades:DataObjectFormat</code> AND each <code>xades:DataObjectFormat</code> points to one <code>ds:Reference</code> element referencing a signed data object.</td>
</tr>
</tbody>
</table>

4.2.4.9 CheckIfSignerCertIsInKeyInfo test

This entry of the table reports on a test that checks that the signer’s certificate is present within the `ds:KeyInfo` element.

**EXAMPLE:**

<table>
<thead>
<tr>
<th>Location</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{CodeTest}</code>:KeyInfo-{CheckIfSignerCertIsInKeyInfo}</td>
<td></td>
</tr>
</tbody>
</table>

4.2.4.10 CheckIfValueIsURIWithoutNoFragmentPart test

This entry of the table reports on a test that checks that a certain URI reference value does not have the fragment part.

**EXAMPLE:**

<table>
<thead>
<tr>
<th>Location</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{CodeTest}</code>:SignedDataObjectProperties/IndividualDataObjectsTimeStamp[1]/Include[1]/@URI-{CheckIfValueIsURIWithoutNoFragmentPart}</td>
<td></td>
</tr>
</tbody>
</table>

4.2.4.11 CheckIfValueIsXMLDateTime test

This entry of the table reports on a test that checks that the content of the XML element under test is of type `dateTime`.

**EXAMPLE:**

<table>
<thead>
<tr>
<th>Location</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{CodeTest}</code>:SignedSignatureProperties/SigningTime-{CheckIfValueIsXMLDateTime}</td>
<td></td>
</tr>
</tbody>
</table>

4.2.4.12 CheckNoAttributes test

This entry of the table reports on a test that checks that the XML element under test does not contain any XML attribute.

**EXAMPLE:**

<table>
<thead>
<tr>
<th>Location</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{CodeTest}</code>:SignedSignatureProperties/SigningCertificate-{CheckNoAttributes}</td>
<td></td>
</tr>
</tbody>
</table>
No attributes were found for this component, as specified by the schema

4.2.4.14 CheckPointsTodsReference test
This entry of the table reports on a test that checks that the XML element under test points to one \texttt{ds:Reference} referencing one signed data object.

EXAMPLE:

<table>
<thead>
<tr>
<th>Location</th>
<th>CheckPointsTodsReference</th>
</tr>
</thead>
<tbody>
<tr>
<td>{CodeTest}:IndividualDataObjectsTimeStamp[1]/Include[1]/@URI</td>
<td>This component points to a \texttt{ds:Reference} element.</td>
</tr>
</tbody>
</table>

4.2.4.15 CheckSchemaForAttributes test
This entry of the table reports on a test that checks that the set of attributes present within the XML element under test matches the XML Schema.

EXAMPLE:

<table>
<thead>
<tr>
<th>Location</th>
<th>CheckSchemaForAttributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes found: ObjectReference</td>
<td></td>
</tr>
</tbody>
</table>

4.2.4.16 CheckSignedInfoReference test
This entry of the table reports on a test that processes the \texttt{ds:Reference} XML element under test as specified in XMLDSIG and checks that the digest value of the obtained result matches the digest value present within this element.

EXAMPLE:

<table>
<thead>
<tr>
<th>Location</th>
<th>CheckSignedInfoReference</th>
</tr>
</thead>
<tbody>
<tr>
<td>{CodeTest}:SignedInfo/Reference[1]/DigestValue</td>
<td>The digest computed value is the same as the present digest value:</td>
</tr>
<tr>
<td>\texttt{'5UyKB9ht94y6CZNvLdO1C7Z3MXaYc2Qol3Dt3Qp4Ajq='}</td>
<td></td>
</tr>
</tbody>
</table>

4.2.4.17 DecodeAndShowIssuerSerialInFullReport test
This entry of the table reports on a test that decodes the contents of the \texttt{xades:IssuerSerialV2} XML element and shows its string representation.

EXAMPLE:

<table>
<thead>
<tr>
<th>Location</th>
<th>DecodeAndShowIssuerSerialInFullReport</th>
</tr>
</thead>
</table>
4.2.4.18 IsNotDistributedXAdES test

This entry of the table reports on a test that checks that the XAdES signature is not a distributed one.

**EXAMPLE:**

| Location-{|CodeTest|-|IsNotDistributedXAdES} |
|---|---|---|

4.2.4.19 TestIfPointsToIDOTSTOrADOTST test

This entry of the table reports on a test that checks that the URI attribute of xadesv141:TimeStampValidationData qualifying property points to a xades:IndividualDataObjectsTimeStamp or xades:AllDataObjectsTimeStamp qualifying properties.

**EXAMPLE:**

| Location-{|CodeTest|:UnsignedSignatureProperties/TimeStampValidationData[1]/@URI-|TestIfPointsToIDOTSTOrADOTST} |
|---|---|---|
| This element points to one `xadesv132:IndividualDataObjectsTimeStamp` element. |

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

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

5.1 Basic conformance-testing tools’ conceptual model

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

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

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

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

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

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

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

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

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

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

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

Figure 9 below shows the basic conceptual model that groups the aforementioned concepts.
Figure 9: Basic conceptual model of the conformance-testing tools
The instance of PortalAdESConformanceChecker is responsible of coordinating the creation and work of the rest of the objects so that each one properly cooperate with others and jointly performs the suitable set of conformance checks on each component of the AdES signature under test.

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

5.2 Commands and XML driving instructions file

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

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

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

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

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

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

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

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

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

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

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

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

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

5.3 Overview of the operation of the testing tools

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

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

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

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

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

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

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

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

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

The conformance-testing tool generates each HTML file by applying a specific XSLT transformation designed for such purpose, to the XMLRawReport. The XSLTReporter object is in charge of applying these XSLT transformations to the XMLRawReport. Attached to this document, a couple of examples of the output HTML documents framework are delivered. These reports corresponds to XAdES signatures generated by participants in the XAdES interoperability remote plugtest™ started on 1st October 2015. The details table is manipulated in order to preserve the anonymity of the generators of the signatures, according to the NDA signed by the participants in the interoperability event.
6 Specification of commands that are common to all the conformance-testing tools

6.1 Introduction

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

Commands in clause 6.2 are commands that perform specific checks on the values of the components under test.

Commands in clause 6.3 are commands that implement control flow (moves to a certain child, iterates on all the children, etc).

Commands in clause 6.4 are commands for managing variables.

6.2 Commands for checking contents of components

6.2.1 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:

```xml
<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`.

6.2.2 CheckNoChildrenElements

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

It has no command arguments.

6.2.3 CheckIfValuesEqualTo

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.

6.2.4 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.

6.2.5 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.

6.2.6 MustNotBeEmpty

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

This command does not have command arguments.

6.2.7 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.

6.2.8 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.

6.2.9 CheckIfSomeChildrenPresent

This commands instructs the conformance-testing tool check whether certain children elements are present within the component under test.

**EXAMPLE of usage:** when checking against XAdES baseline signature a signature may contain xadesv141:CompleteCertificateRefsV2 qualifying properties if this signature is not a XAdES-B-LTA, but it can not contain this property if it is a XAdES-B-LTA. Consequently if the signature contains the xadesv141:ArchiveTimeStamp qualifying property and the xadesv141:CompleteCertificateRefsV2, then it is not conformant with the specification of XAdES-B-LTA signatures.

Command arguments:

<children> child element contains the list of children names whose presence wants to be detected, separated by a comma.

'resultTo' identifies a Boolean local variable where the command puts the result of the check.

'detailsTo' identifies a String local variable that contains the names of those searched children that have actually been found.
6.2.10 CheckIfValueIsBase64

This command instructs the conformance-testing tool check whether the value of the component under test is a legal base64 value.

Command arguments:

'specifiedBySchema' is an optional Boolean attribute argument. It indicates whether this requirement 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.

'showInFullReport' is an optional Boolean attribute argument. If true then the AdESCC will provide details of the found value within the full report.

6.2.11 CheckIfValueIsBoolean

This command instructs the conformance-testing tool check whether the value of the component under test is a legal Boolean value.

Command arguments:

'specifiedBySchema' is an optional Boolean attribute argument. It indicates whether this requirement 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.

'showInFullReport' is an optional Boolean attribute argument. If true then the AdESCC will provide details of the found value within the full report.

6.2.12 CheckIfValueIsEmpty

This command instructs the conformance-testing tool check whether the value of the component under test is an empty value. This command does not have any command argument.

6.3 Commands for implementing flow control

6.3.1 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.

'numInstances' 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.

6.3.2 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.

6.3.3 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.

6.3.4 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.

6.3.5 CaseAncestor

This command checks if the component under test has one ancestor whose name is passed as a command argument. If this is the case, then the conformance-testing tool executes all the XML-encoded commands that are the XML children of the CaseAncestor XML element.

Command arguments:
‘ancestor’ mandatory attribute contains the name of the searched ancestor.
6.3.6 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.

6.3.7 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.

6.4 Commands for managing variables

6.4.1 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.

6.4.1 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.

6.4.2 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.
7 Specification of commands that are applicable only for XAdES conformance testing

7.1 Introduction

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

7.2 CheckIfIsXAdESSignature

This command checks whether the signature under test is a XAdES signature as specified by ETSI TS 319 132 Part 1. More specifically it checks that either:

a) The signature incorporates the xades:SigningCertificateV2 qualifying property and that it references the signer's certificate AND that the xades:SignedProperties container is covered by the signature itself OR

b) the signer's cert is within ds:KeyInfo AND it is signed by the signature itself.

Command arguments:

'isSignerCertReferenced' input argument that contains the name of a Boolean local variable that indicates whether the signer's certificate of the XAdES signature is referenced by the signing certificate qualifying property.

'result' output argument that contains the name of a Boolean local variable indicating whether the signature can be considered a XAdES signature.

7.3 CheckDigestValueInReferenceWithinSignedManifest

This command checks the correctness content of ds:DigestValue child element of a ds:Reference child element of a signed ds:Manifest element. This command does not require the usage of any argument.

7.4 CheckEncapsulatesOCSPValueAndAddToVariable

This command checks if the XML element pointed by the cursor encapsulates an instance of OCSPResponse type. The command manages two command arguments whose details are shown below.

Command arguments:

'detailsTo' identifies a local variable that is a set of byte arrays, each containing the encoded instance of OCSPResponse type.

7.5 CheckEncapsulatesX509CertificateAndAddToVariable

This command checks if the XML element pointed by the cursor encapsulates an X509 certificate. The command manages one command argument whose details are shown below.

Command arguments:

'detailsTo' identifies a local variable that is a set of byte arrays, each containing the encoded instance of OCSPResponse type.
7.6 CheckEncapsulatesX509CRLAndAddToVariable

This command checks that the XML element under test encapsulates an X509 CRL value. The command manages one command argument whose details are shown below.

Command arguments:

'detailsTo' identifies a local variable that is a set of byte arrays, each containing the encoded X509 CRL.

7.7 CheckIfDOFPointsToReferenceToSignedObject

This command checks that the xades:DataObjectFormat qualifying property under test points to one ds:Reference that references a signed data object. This command manages one command argument whose details are shown below.

Command arguments:

'addTo' identifies a local variable that is a map whose keys are the result of applying the exclusive canonicalization of the ds:Reference elements pointed xades:DataObjectFormat qualifying properties and whose values are references to the aforementioned xades:DataObjectFormat qualifying properties. This command generates a new pair within the map each time it is invoked.

7.8 CheckIfDOFsPointToAllReferencesToSignedObjects

This command checks that all the xades:DataObjectFormat qualifying properties incorporated into the XAdES signature point to the ds:Reference elements that reference all the signed data objects. This command is invoked when all the xades:DataObjectFormat qualifying properties have been individually tested for ensuring that the set of all the xades:DataObjectFormat qualifying properties satisfies the requirements specified by ETSI EN 319 123 Part 1 [1].

Command arguments:

'varIn' identifies a local variable that is a map whose keys are the result of applying the exclusive canonicalization of the ds:Reference elements pointed xades:DataObjectFormat qualifying properties and whose values are references to the aforementioned xades:DataObjectFormat qualifying properties. This is the map filled by the CheckIfDOFPointsToReferenceToSignedObject command each time one xades:DataObjectFormat qualifying property is tested.

7.9 CheckIfSignerCertIsInKeyInfo

This command checks that the signer’s certificate is present within the ds:KeyInfo element. This command does not have any command argument.

7.10 CheckIfValueIsURIWithoutNoFragmentPart

This command checks that a certain URI reference value does not have the fragment part. This command does not have any command argument.

7.11 CheckIfValueIsXMLDateTime

This command checks that the content of the XML element under test is of type dateTime. This command does not have any command argument.
7.12 CheckNoAttributes

This command checks that the XML element under test does not contain any XML attribute.

Command arguments:

'specifiedBySchema' is a Boolean input argument. Its value indicates whether the fact that the XML element under test must not have any attribute is specified by the XML Schema (value will be "true") or if the XML Schema allows that the element has some attribute but the ETSI EN 319 132 in some of its parts imposes the absence of any attribute for the reference XAdES baseline signature or the specific extended XAdES signature against which the XAdES conformance-testing tool checks the signature under test.

7.14 CheckPointsTodsReference

This command checks that the XML element under test points to one ds:Reference referencing one signed data object. This command does not have any command argument.

7.15 CheckSchemaForAttributes

This command checks that the set of attributes present within the XML element under test matches the XML Schema.

Command arguments:

'spec' is a string that enumerates the names of the attributes that the XML element under test may have. The optional attributes are qualified with the '?' character. Those attributes whose names are not qualified with the aforementioned character are required attributes that must be present in the XML element under test.

7.16 CheckSignedInfoReference

This command processes the ds:Reference XML element under test as specified in XMLDSIG and checks that the digest value of the obtained result matches the digest value present within this element.

Command arguments:

'iteratorName' is the name of an integer local variable that holds the number of the checked reference. Its value is incremented each time one ds:Reference XML element is tested.

7.17 DecodeAndShowIssuerSerialInFullReport

This command decodes the contents of the xades:IssuerSerialV2 XML element and shows its string representation. This command does not have any command argument.

7.18 IsNotDistributedXAdES

This command checks that the XAdES signature is not a XAdES distributed signature.

Command arguments:

'result' is the name of a Boolean local variable whose value is set to true if the signature is a XAdES distributed signature and it is set to false otherwise.

7.19 GetExCanIDOTSTsAndADOTSTs

This command takes each xades:IndividualDataObjectsTimeStamp and xades:AllDataObjectsTimeStamp qualifying properties incorporated into the XAdES signature, computes
their exclusive canonicalization transformations, and stores them into a list local variable whose name is passed as a command argument.

Command arguments:

'result' is the name of an local variable that holds a list. This list contains the result of computing the exclusive canonicalization to all the xades:IndividualDataObjectsTimeStmp and xades:AllDataObjectsTimeStmp qualifying properties incorporated into the signature.

7.20 TestIfPointsToIDOTSTOrADOTST

This command checks that the URI attribute of xadesv141:TimeStampValidationData qualifying property points to a xades:IndividualDataObjectsTimeStmp or xades:AllDataObjectsTimeStmp qualifying properties. This command processes retrieves the object referenced by the value of the aforementioned attribute computes the exclusive canonicalization of the retrieved object if it is a node set, and checks whether the resulting value is present in a list that is hold in a variable whose name is passed as a command argument.

Command arguments:

$input' is the name of an local variable that holds a list. This list contains the result of computing the exclusive canonicalization to all the xades:IndividualDataObjectsTimeStmp and xades:AllDataObjectsTimeStmp qualifying properties incorporated into the signature.

7.21 CursorToXMLAttribute

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

Command arguments:

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

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

'numInstances' 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 XML attribute is defined in the reference specification by the XML Schema or not.