ETSI XAdES PLUGTESTS™

Final Report

Martin Centner, Juan Carlos Cruellas, Peter Lipp

Draft

Table of Content

Gelöscht: 1 Introduction 3¶
2 Test Scenario . 4¶
2.1 General Assumptions . 4¶
2.2 Test Data 5¶
3 Test Matrix 5¶
3.1 XAdES Test cases 6¶
3.2 XADES-T-Test cases . 6¶
3.3 XAdES-C and XAdES-X-Testcases 6¶
3.4 XADES-X-L-Testcases 7¶
3.5 XAdES-A test cases . 7¶
4 Participating Implementations 7¶
4.1 Baltimore 7¶
4.2 IAIK . 9¶
4.3 Kopint–Datorg Rt. 10¶
4.4 Microsoft 11¶
4.5 UPC 12¶
4.6 Sertifitseerimiskeskus . 13¶
5 Feature Matrix . 15¶
6 Interoperability Results . 19¶
6.1 Verification results for signatures created by Baltimore 20¶
6.2 Verification results for signatures created by Kopdat . 21¶
6.3 Verification results for signatures created by IAIK 22¶
6.4 Verification results for signatures created by Microsoft 23¶
6.5 Verification results for signatures created by UPC 24¶
6.6 Verification results for signatures created by SK 25¶
7 Input for the XAdES Maintenance Process 26¶
7.1 Issue #1 – <EncapsulatedOCSPValues> 26¶
7.2 Issue #2 – <TimeStampType> Data Type . 27¶
7.3 Issue #3 – <ArchiveTimeStamp> 29¶
7.4 Issue #4 – Requirement Levels (RFC2119) 30¶
7.5 Issue #5 – <QualityingProperties> 31¶
7.6 Issue #6 – ASN.1 Encoding . 31¶
7.7 Issue #7 – Trust Status Lists . 32¶
7.8 Issue #8 – <SigningCertificate> 32¶
7.9 Issue #9 – XAdES forms 32¶
7.10 Issue #10 – archival forms 32¶
7.11 Issue #11 – <AnyType> Data Type 33¶
7.12 Issue #12 – <CertID> 33¶
7.13 Issue #13 – .NET validating parser 33¶
7.14 Issue #14 – XAdES schema 34¶
7.15 Issue #15 – <QualifyingPropertiesReferenceType> data type . ... [1]

1 Introduction

ETSI has hosted the XAdES-PLUGTESTS™ interoperability event in Sophia Antipolis, France from the 3rd to the 7th of November, 2003. The interoperability event was intended to support software developers that have implemented the XAdES specification [1] to create interoperable implementations and to get feedback from the implementers as input for the XAdES maintenance process and future versions of the XAdES specification.

The participants of the event were:

- **Agencia Catalana de Certificació - CATCert**
  Marta Cruellas (mcruellas@catcert.net)
- **Baltimore Technologies**
  Vivekanand Sakaram (vsakaram@baltimore.com)
- **Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology**
  Martin Centner (mcentner@iaik.tugraz.at)
  Peter Lipp (plipp@iaik.tugraz.at)
- **Kopint–Datorg Rt. (Kopdat)**
  Balás Doházs Andrányos (balazs.dohanyos@kopdat.hu)
- **Microsoft**
  Eddy Rubens (eddyrube@microsoft.com)
  Stefan Santesson (stefans@microsoft.com)
- **AS Sertifitseerimiskeskus (SK)**
  Tarvi Martens (tarvi.martens@sk.ee)
- **Universitat Politecnica de Catalunya (UPC)**
  Joan Arnedo (joanar@ac.upc.es)
  Juan Carlos Cruellas (cruellas@ac.upc.es)

XAdES implementations were provided by:

- **Baltimore**
- **IAIK**
- **Kopint–Datorg** (partial implementation)
- **Microsoft**
- **SK** (partial implementation)
- **UPC**

2 Test Scenario

XAdES interoperability event in November concentrated on a number of test cases (XAdES signatures containing different qualifying properties). This section describes the rules provides to participants prior to the event as a technical guidance to enable smooth testing at the event

Participants were expected to be able to generate XAdES signatures to at least a subset of the test cases specified below, ideally most or all of the test cases. The result of these were to be stored in a form specified below, zipped and made available to other participants for evaluation.

Each participant was expected to be able to parse and process XAdES signatures provided in this way by the other participants and to provide a report on the results of the verification.

Signatures generated by participants as described above were expected to be valid signatures and should be verifiable by other implementations. We expected that during the tests several implementations will produce signatures that were in fact invalid for all kinds of reasons. Such cases were added to the list of test cases dynamically and verification by implementations is expected to fail with an appropriate reason code. The interop specification did not specify the form an implementation reports failed or even successful verifications. Checking the correctness of the validation process was expected to happen manually at the event.

2.1 General Assumptions

This section provides information on general assumptions made by the core team when facing the test cases for this first event.

1. Participants will bring their own equipment sufficient to be able to conduct the tests and change or fix their software on site. There will be wired and wireless internet connectivity provided. Nothing else should be assumed, if not mentioned below. Participants require some extra feature need to ensure its availability in time.
2. All the signatures produced will be enveloping signatures.
3. It is out of the scope of the event to deal with trust issues (including trusted certificates, etc) and certificate chain validation “obscurities” (like policy issues or chain- versus shell-.verification model etc). This means: any correct certificate and complete certificate chain provided by another participant must be accepted and cannot lead to a failure of verification of the signature.
4. Each participants signature creation or extension environment should be self contained, i.e. each participant is responsible for providing all the elements required in its signature (time-stamps, CRLs and OCSP responses). Participants or third parties may provide online services (like OCSP, LDAP etc.) but the success of the event should not be jeopardized by interoperability problems with such services. IAIK will provide demo-services allowing for creation of certificates and corresponding revocation information. Participants may make use of these services and should prepare using those services beforehand.

5. Creation of signatures will require online use of a time stamping service. IAIK is providing such a service currently for testing purposes and makes this available at the event. Participants should check if they can use that service properly (tsa.iaik.at)

2.2 Test Data

All participating implementations must

- be able to accept test data according to the following specifications and
- be able to store the results of signature creation for subsequent verification of other implementations according to the following specifications.

It is the decision of the implementers on how to achieve these.

2.2.1 Test case data specification

All data that is required to be able to verify a signature MUST be available in one ZIP-file. This allows us to make test data for a special test case available as one file on the web or elsewhere.

Unpacking the file must result in one directory, containing all required files. The following files must exist in all cases:

- ***./readme.txt***: a file identifying the test case supported by the test data set
- ***./Signature.xml*** containing the signature to be verified.
- ***./SignerCertificate.der*** containing the certificate corresponding to the signers private key in DER-encoding.
- ***./CertificateChain.der*** containing the certificate chain starting from the signer certificate up to whatever trusted (root) certificate applicable.
- ***Any other elements*****,** like CRLs, OCSP-Responses or Data referenced in properties or the signed document itself, which are required for verifying the signature must be placed within the ZIP-file using any unambiguous filename. The references within the signature must be file URLs relative to the location of the signature document.

3 Test Matrix

In the preparation for the interoperability event a number of test cases had been developed, each test case representing a different signature format. During the interoperability event each participant had to create the defined test signatures and had to verify the test signatures created by all the other participants. In the original document test case ‘XAdESX#4’ with `<RefsOnlyTimeStamp>` that built on test case ‘XAdESC#2’ had been defined. This test case has been removed because it was not conforming to the specification. Consequently the test cases ‘XAdESXL#4’ and ‘XAdESA#4’ have been removed as well. The test case ‘XAdESA#5’ was redefined to build upon ‘XAdESA#3’.

3.1 XAdES Test cases

test case	<Signing Time>	<Signing Certificate>	<Signature Policy Identifier>	additional property
XAdES#1	●	●	●	
XAdES#2	●	●	●	<SignatureProductionPlace>
XAdES#3	●	●	●	<SignerRole>
XAdES#4	●	●	●	<CommitmentTypeIndication>
XAdES#5	●	●	●	<DataObjectFormat>
XAdES#6	●	●	●	<CounterSignature>
XAdES#7	●	●	●	<AllDataObjectsTimeStamp>
XAdES#8	●	●	●	<IndividualDataObjectsTimeStamp>

Table 3-1: XAdES-Test cases

3.2 XADES-T-Test cases

	PROPERTIES WITHIN SIGNATURE	
TEST CASE CODE ↓	**Properties present in test case**	**Signature Time Stamp**
XAdES-T#1	XAdES#1	●

Table 3-2: XAdES-T-Test cases

3.3 XAdES-C and XAdES-X-Testcases

	PROPERTIES WITHIN SIGNATURE					
TEST CASE CODE ↓	**Properties present in test case**	**Complete Certificate Refs**	**Complete Revocation Refs -using CRLRefs-**	**Complete Revocation Refs -using OCSPRefs-**	**Sig And Refs Time Stamp**	**Refs Only Time Stamp**
XAdES-C#1	XAdES-T#1	X	X			
XAdES-C#2	XAdES-T#1	X		X		
XAdES-X#1	XAdES-T#1	X	X		X	
XAdES-X#2	XAdES-T#1	X	X			X
XAdES-X#3	XAdES-T#1	X		X	X	

Table 3-3: XAdES-C and XAdES-X -Test cases

3.4 XADES-X-L-Testcases

TEST CASE CODE ↓	PROPERTIES WITHIN SIGNATURE			
	Properties present in test case	**Certificate Values**	**Revocation Values -using CRLValues-**	**Revocation Values -using OCSPValues-**
XAdES-X-L#1	XAdES-X#1	X	X	
XAdES-X-L#2	XAdES-X#2	X	X	
XAdES-X-L#3	XAdES-X#3	X		X

Table 3-4: XAdES-X-L-Test cases

3.5 XAdES-A test cases

TEST CASE CODE ↓	PROPERTIES WITHIN SIGNATURE	
	Properties present in test case	**Archive Time Stamp**
XAdES-A#1	XAdES-X-L#1	X
XAdES-A#2	XAdES-X-L#2	X
XAdES-A#3	XAdES-X-L#3	X
XAdES-A#5	XAdES-A#3	X

Table 3-5: XAdES-A-Test cases

4 Participating Implementations

4.1 Baltimore

The implementation by Baltimore Technologies was the perhaps most complete implementation of the XAdES specification so far. Most of the XAdES features had been implemented. `<QualifyingPropertiesReferences>` had not been implemented yet (but they are not implemented by any participating party, so far). The implementation was

Implementation by	Baltimore
implementation language	Java
base crypto toolkit used	KeyTools Pro
base xml-signature toolkit used	KeyTools XML Java
XAdES implementation will be available in what form	Xades implementation from Baltimore will be a toolkit
conditions of availability	Probably *only* under commercial licence
source code availability	Source licenses are negotiated on an individual basis.
estimated time of release	Sometime after the spec is finalised.
person to contact	Ulrich Brell
any other information that may be of interest	n/a

Table 4-1: Implementation by Baltimore

able to parse and verify all test signatures produced by the other participants which were aligned with the XAdES schema. The implementation was not able to verify the test cases 'XAdES-A#1' to 'XAdES-A#5' produced by IAIK because the Baltimore was not aware of a agreement on a slight change of the XAdES schema that was discussed on the xades-plugtests@list.etsi.org mailing list in advance of the interoperability tests (see below). However, the implementation by Baltimore was correct and produced signatures that were aligned with the XAdES specification.

4.2 IAIK

Like the implementation by Baltimore Technology, the implementation by IAIK supported most of the XAdES features except for the `<QualifyingPropertiesReference>`. The signed data object properties `<CommitmentTypeIndication>`, `<DataObjectFormat>` and `<CounterSignature>` had not been implemented so far. Therefore, the test cases 'XAdES#4', 'XAdES#5' and 'XAdES#6' were not produced by the IAIK implementation.

Implementation by	**IAIK**
implementation language	Java
base crypto toolkit used	IAIK JCE toolkit
base xml-signature toolkit used	IAIK IXSIL
XAdES implementation will be available in what form	XAdES implementation from IAIK will be a toolkit
conditions of availability	free educational and research licenses, commercial licence
source code availability	yes, for commercial licenses
estimated time of release	Sometime after the spec is finalised.
person to contact	Peter Lipp
any other information that may be of interest	n/a

Table 4-2: Implementation by IAIK

IAIK had already adopted a change of the XAdES schema that was agreed on in advance of the interoperability event by subscribers of the XAdES-plugtests@list.etsi.org-mailing list. Therefore, the IAIK implementation produced signatures that where not completely aligned with the latest XAdES schema version used at the interoperability event. Baltimore Technologies was, however, sticking to the XAdES schema and was therefore not able to verify the XAdES-A signature test cases produced by IAIK.

During the interoperability tests it was agreed that changes will have to be made to the XAdES specification any way, which will make this issue obsolete. Therefore, it was agreed to leave both implementations untouched and accept the fail of those test cases.

Some of the verifications for completeness of the different XAdES forms had not been implemented so far, while all cryptographic verifications had been implemented. The verification of the input for the different time-stamps used in XAdES had been omitted, because it had been assumed that changes in the XAdES specification are necessary to make this verification feasible in the general case.

4.3 Kopint–Datorg Rt.

The implementation by Kopint–Datorg Rt. used a different schema and supported just some parts of the XAdES specification. As both, the XAdES and the XMLDSig schema, where violated at different points it was difficult for the other participants to verify the signatures produced by the implementation of Kopint–Datorg Rt.

Implementation by	Kopint-Datorg Rt
implementation language	C++ (compiled in VC6.0)
base crypto toolkit used	MS Crypto API
base xml-signature toolkit used	MSXML DOM only
XAdES implementation will be available in what form	commercial XAdES + xml package handler API sold in an SDK
conditions of availability	contact sales
source code availability	no
estimated time of release	sept. 2003
person to contact	Sales: Mr. Zsolt Hevesi Technical: Balazs Andras Dohanyo
any other information that may be of interest	

Table 4-3: Implementation by Kopdat

Since just some parts of the XAdES specification had been implemented and a different signature schema had been used, Kopint–Datorg Rt. was not able to verify XAdES signatures produced by the other participants.

4.4 Microsoft

Microsoft's XAdES implementation was in a relatively early stage of the development. The .NET XMLDSig implementation was for any reasons not able to verify the underlying XMLDSig signature of the XAdES signatures provided by the other participants. This problem could not be solved during the PLUGTESTS™ event.

Beside this issue, Microsoft had no implementation of the time-stamp protocol (TSP – RFC3161 [2]) and the online certificate status protocol (OCSP – RFC2560 [3]) for the .NET frame work. So, only a very limited part of the XAdES features could be created and tested by Microsoft's XAdES implementation.

Implementation by	Microsoft
implementation language	C#
base crypto toolkit used	Microsoft .NET Framework 1.1
base xml-signature toolkit used	.NET Framework 1.1 (SignedXml class)
XAdES implementation will be available in what form	Toolkit (other forms of distribution are discussed)
conditions of availability	Free
source code availability	Yes
estimated time of release	Early 2004
person to contact	Ronny Bjones
any other information that may be of interest	The library is implemented as a derivation of the SignedXml class of the Microsoft .NET Framework

Table 4-4: Implementation by Microsoft

The basic XAdES structures produced by Microsoft's implementation could be parsed and verified by the implementation of Baltimore Technologies, IAIK and UPC. However, the underlying XMLDSig signature-verification failed in all test cases provided by Microsoft—most likely for the same reason that Microsoft was not able to verify the signatures provided by the other participants.

4.5 UPC

UPC provided a rather complete implementation of the XAdES specification, as well. The `<QualifyingPropertiesReference>` had not been implemented either. The UPC software made usage of a time-stamp server and client software that resulted in the generation of time-stamps that were not correctly verified by the rest of the implementations. It was necessary then to incorporate a new time-stamp client for requesting time-stamps from the time-stamp server provided by IAIK. A part of the interoperability event was spent trying to fix these problems. Once they were fixed, the

Implementation by	1.1 UPC
implementation language	Java
base crypto toolkit used	Java SDK, SUN cryptographic provider
base xml-signature toolkit used	UPC-xslib (a XMLDSIG tool implemented by UPC)
XAdES implementation will be available in what form	XAdES implementation from UPC will be a toolkit
conditions of availability	probably commercial licence
source code availability	no
estimated time of release	short time after the spec. review will be finalized
person to contact	Juan Carlos Cruellas
any other information that may be of interest	

Table 4-5: Implementation by UPC

tool was able to correctly parse and verify XAdES signatures generated by IAIK and Baltimore incorporating different kinds of time-stamps. All signatures created by UPC could by verified by Baltimore and IAIK, beside the verification of the included time-stamps. Signatures with revocation values using OCSP responses were not provided by UPC during the interoperability event.

4.6 Sertifitseerimiskeskus

SK provided an already deployed implementation of the XAdES specification. However, only a limited subset of the defined qualifying properties had been implemented. The different time-stamp properties had not implemented. Instead, the OCSP protocol is 'abused' to provide time-stamping functionality. The SK implementation sends a hash of the signature value as nonce to the OCSP server. The nonce is intentionally used in the

By	Sertifitseerimiskeskus
implementation language	C and Java (2 implementations)
base crypto toolkit used	C - OpenSSL, Java - Bounty-Castle
base xml-signature toolkit used	none
XAdES implementation will be available in what form	• C library toolkit for Win32/Linux/FreeBSD/... (CDigiDoc) • Java library toolkit (JDigiDoc) • Windows COM library • Windows end-user application • Linux Portal application
conditions of availability	• C, Java library toolkits - LGPL • Windows COM library and end-user application - freeware • Portal - free for use, source code - commercial
source code availability	libraries are available today www.openxades.org
estimated time of release	International Windows Client will be released in 1Q2004
person to contact	Tarvi Martens, tarvi@sk.ee
any other information that may be of interest	www.openxades.org, www.id.ee

Table 4-6: Implementation from Estonia

OCSP protocol as counter measure against replay attacks.

During the interoperability event the approach that SK used to provide time-stamping functionality has been discussed. It was understood by most of the participants that, while it is principally possible to use the nonce to provide time-stamp functionality, it is generally not a good idea to create proprietary solutions by abusing mechanisms of standard protocols that are intended for different purposes.

Therefore, the solution of SK will most likely not find a recommendation in future versions of the XAdES specification. It was stated that a specification can hardly specify or recommend the abuse of another specification to achieve a certain purpose.

During the interoperability tests some violations of the XAdES schema were identified and fixed. Additionally it turned out, that the XAdES specification is not precise enough when defining what the actual contents of the `<EncapsulatedOCSPValue>` should be. While the specification was interpreted by Baltimore and IAIK to define the use of the `BasicOCSPResponse` as OCSP value, SK and UPC interpreted the specification in a different way, namely to use the `OCSPResponse` which wraps the `BasicOCSPResponse`. Therefore, it was agreed to improve the specification in this point, to be more precise about the contents of the `<EncapsulatedOCSPValue>`.

5 Feature Matrix

The following matrix shows the verifications performed or planned to be performed by the different implementations. This is important to properly understand the interoperability-results and also gives a better feeling for the status of different implementations.

The icons have the following meaning:

- ● implemented
- ○ planned to implement
- ⊙ not applicable
- × not to be implemented

Verification	Balti.	IAIK	Kopdat	MS	UPC
checks if the document uses the correct namespace URI http://uri.etsi.org/01903/v1.1.1#	●	●	○	●	●
Verifies if document is schema-valid	●	●	●	●[1]	●
if schema is not checked, verifies if document contains `SigningTime`	⊙	⊙	⊙	●	●
if schema is not checked, verifies if document contains `SigningCertificate`	⊙	⊙	⊙	●	●
if schema is not checked, verifies if document contains `SignaturePolicyIdentifier`	⊙	⊙	⊙	●	●
verifies, if `SignedProperties` are the only elements defined within XAdES that are covered by the signature	×	×	●	●	○
verifies that the `<SigningCertificate>` element contains a certificate corresponding to the public key that has been used to verify the signature	●	○	●	○	○
verifies the signature of any CRL it encounters (cryptographically, no trust decisions!!!!)	●	●	●	○	○

[1] The published schema doesn't work with the Microsoft .NET XmlValidatingReader class - a modified schema is used. Validating against the schema is an option of the library

verifies the signature of any OCSP-response it encounters (cryptographically, no trust decisions!!!!)	●	●	●	○	○
verifies the signature of any timestamp it encounters (cryptographically, no trust decisions!!!!)	●	●	●	○	●
verifies the `AllDataObjectsTimeStamp` covers all `ds:Reference` elements within `ds:SignedInfo` except `SignedProperties`	●	○	○	○	○
verifies that any `IndividualDataObjectsTimestamp` refers to one of the `ds:Reference` elements to `ds:SignedInfo`	●	○	○	○	○
the implementation verifies if the digest within the `SignaturePolicyIDentifier`-element corresponds to the data received when dereferencing the SPURI-Qualifier	●	●	○	○	●
any Countersignatures encountered are verified using XAdES mechanisms	●	○	○	●	○
any Countersignatures encountered are checked if they correctly reference the original signature value	●	○	○	○	○
checks if all ObjectReferences in the `CommitmentTypeIndication` element reference an element of the signature	●	×	○	○	×
checks if at least one of `ClaimedRoles` or `CertifiedRoles` must be present, if the `SignerRole` element is present	●	○	○	●	●
ensures that the input to the `SignatureTimeStamp` is the `ds:SignatureValue` element	●	○	○	○	○
checks if the `SigAndRefsTimestamp` element contains a sequence of `HashDataInfo`-elements that indeed refer to the elements they are supposed to refer to according to the standard	●	○	○	○	○
checks if the `RefsOnlyTimestamp` element contains a sequence of `HashDataInfo`-elements that indeed refer to the elements they are supposed to refer to according to the standard	●	○	○	○	○

checks that for each certificate in the `CompleteCertificateRefs`-element there is one entry in the `CompleteRevocationRefs`-element (except for self-signed-certs)	●	×[2]	○	○	●
checks, if all cerificates referenced in the `CompleteCertificateRefs`-element and the `SigningCerificate` are present in the `CertificateValues` element	●	×[2]	○	○	●
checks, if all revocation information referenced in the `CompleteRevocationRefs`-element are present in the `RevocationValues` element	●	×[2]	○	○	●
check if hashes found in `CompleteCertificateRefs` and `CompleteRevocationRefs` correspond to the certificates and revocation information used for validating the signature	×	×[2]	○	○	●
checks, if the certificate corresponding to the private key used for signing has not been revoked (or any of the certificates in the chain).	●	×[2]	○	○	●
checks if the `ArchiveTimestamp` element contains a sequence of `HashDataInfo`-elements that indeed refer to the elements they are supposed to refer to according to the standard	●	○	○	○	○
checks, if the signature seems to be of form XADES-C (contains complete certificate and revocation references) it also contains all elements required for XAdES-T (time stamp over digital signature)	●	○	●	○	○
checks, if the signature seems to be of form XADES-XL (contains complete certificate and revocation values) it also contains all elements required for XAdES-X	●	○	○	○	○
checks if the target within the `QualifyingProperties` element refers to the XML signature they are associated with	●	○	○	○	○
checks if any existing `QualifyingProperties`	×	○	○	○	○

Gelöscht: [1]

[2] will be implemented in the signature policy module

elements and `QualifyingProperties` Reference elements occur within a single `ds:Object` element					
checks if at most one `QualifyingProperties` element exists within the single `ds:Object`-element	×	○	○	○	×
checks if all signed properties occur within a single `QualifyingProperties` element	×	○	○	○	●

Table 5-1: Implementation features

6 Interoperability Results

The following tables show the results of the verification of the signatures produced by one participant through the implementations of all other participants. The shown verification results represent of course only a snap shot of the implementation's capabilities at the time of the XAdES-PLUGTEST™ event. All implementations are still in development. The capabilities will change and the interoperability matrices are therefore subject to changes. As can be seen from the interoperability matrices below the only applications that were interoperable in almost all cases, were the implementation by Baltimore and the implementation by IAIK—aside from the namespace issue of the Transforms element in the `<HashDataInfo>` elements of the different time-stamps. However, the situation should improve rather fast as work is in progress and the interoperability event has been used to identify the problems of the different implementations. It has been agreed to continue the interoperability tests after the XAdES-PLUGTESTS™ event and to maybe have a further interoperability event in 2004.

6.1 Verification results for signatures created by Baltimore

	Verified by				
Testcase-ID	IAIK	KOPDAT	Microsoft	UPC	remarks
XAdES#1	✓		✓[3]	✓	
XAdES#2	✓	×	✓[3]	✓	
XAdES#3	✓		✓[3]	✓	
XAdES#4	✓[4]	×	✓[3]		
XAdES#5	✓[4]		✓[3]		
XAdES#6	✓[4]		✓[3]		
XAdES#7	✓		✓[3]		
XAdES#8	✓		✓[3]		
XAdES-T#1	✓	×	✓[3]	✓	
XAdES-C#1	✓		✓[3]	✓	
XAdES-C#2	✓		✓[3]		
XAdES-X#1	✓		✓[3]	✓	
XAdES-X#2	✓		✓[3]	✓	
XAdES-X#3	✓		✓[3]		
XAdES-X-L#1	✓		✓[3]	✓	
XAdES-X-L#2	✓		✓[3]	✓	
XAdES-X-L#3	✓		✓[3]		
XAdES-A#1	×[5]		✓[3]	×[6]	

Gelöscht: [2]
Gelöscht: [2]
Gelöscht: [2]
Gelöscht: [3]
Gelöscht: [2]
Gelöscht: [3]
Gelöscht: [2]
Gelöscht: [2]
Gelöscht: [2]
Gelöscht: [2]
Gelöscht: [2]
Gelöscht: [2]
Gelöscht: [2]
Gelöscht: [2]
Gelöscht: [2]
Gelöscht: [2]
Gelöscht: [2]

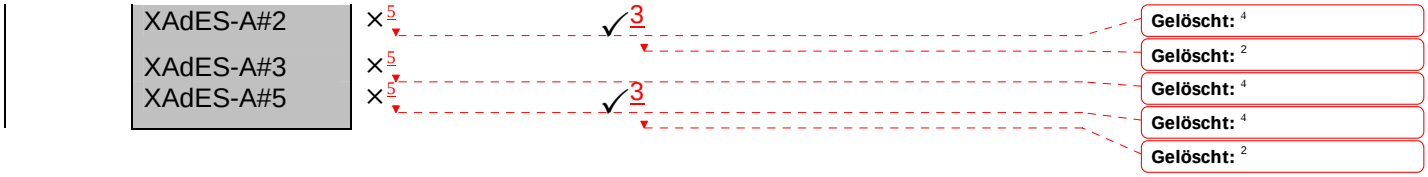
[3] only partial verification:

- the XMLDSig signature was not verified due to problems with the underlying XMLDSig implementation
- TSP time-stamps and OCSP responses were not verified

[4] The `<CommitmentTypeIndication>`, `<DataObjectFormat>` and `<CounterSignature>` properties had not been implemented by IAIK. Therefore, they have not been verified.

[5] caused by different namespaces for the Transforms element in the `<HashDataInfo>` elements (see text for details)

[6] The computed digest in the Archival Timestamp does not match with the digest computed by the verifier after processing the HashDataInfo. The rest of the signature is OK (even the former time-stamps).



6.2 Verification results for signatures created by Kopdat

	Verified by				
Testcase-ID	Baltimore	IAIK	Microsoft	UPC	remarks
XAdES#5	×[7]	×			
XAdES-T#1	×	×			

[7] signatures provided were not aligned with the XAdES schema and were therefore not verifiable

6.3 Verification results for signatures created by IAIK

	Verified by				
Testcase-ID	Baltimore	KOPDAT	Microsoft	UPC	remarks
XAdES#1	✓		✓[8]	✓	
XAdES#2	✓	×	✓[8]	✓	
XAdES#3	✓		✓[8]	✓	
XAdES#4	-	-	-	-	Not provided
XAdES#5	-	-	-	-	Not provided
XAdES#6	-	-	-	-	Not provided
XAdES#7	✓		✓[8]	✓	
XAdES#8	✓		✓[8]	✓	
XAdES-T#1	✓	×	✓[8]	✓	
XAdES-C#1	✓		✓[8]	✓	
XAdES-C#2	✓		✓[8]		
XAdES-X#1	✓		✓[8]	✓	
XAdES-X#2	✓		✓[8]	✓	
XAdES-X#3	✓		✓[8]		
XAdES-X-L#1	✓		✓[8]	✓	
XAdES-X-L#2	✓		✓[8]	✓	
XAdES-X-L#3	✓		✓[8]		
XAdES-A#1	×[9]		✓[8]	✓	
XAdES-A#2	×[9]		✓[8]		
XAdES-A#3	×[9]		✓[8]		
XAdES-A#5	×[9]		✓[8]		

Gelöscht: [7]
Gelöscht: [7]
Gelöscht: [7]
Gelöscht: [7]
Gelöscht: [7]
Gelöscht: [7]
Gelöscht: [7]
Gelöscht: [7]
Gelöscht: [7]
Gelöscht: [7]
Gelöscht: [7]
Gelöscht: [8]
Gelöscht: [7]
Gelöscht: [8]
Gelöscht: [7]
Gelöscht: [8]
Gelöscht: [7]

[8] only partial verification:

- the XMLDSig signature was not verified due to problems with the underlying XMLDSig implementation
- TSP time-stamps and OCSP responses were not verified

[9] caused by different namespaces for the Transforms element in the `<HashDataInfo>` elements (see text for details)

6.4 Verification results for signatures created by Microsoft

	Verified by				
Testcase-ID	Baltimore	IAIK	Kopdat	UPC	remarks
XAdES#1	×[10]		×[10]		
XAdES#2	×[10]		×[10]	×[10]	
XAdES#3	×[10]		×[10]		
XAdES#4	×[10]		×[10]	×[10]	
XAdES#5	×[10]		×[10]		
XAdES#6	×[10]		×[10]		

Gelöscht: [9]
Gelöscht: [9]
Gelöscht: [9]
Gelöscht: [9]
Gelöscht: [9]
Gelöscht: [9]
Gelöscht: [9]
Gelöscht: [9]
Gelöscht: [9]
Gelöscht: [9]
Gelöscht: [9]
Gelöscht: [9]

[10] the verification of the underlying XMLDSig failed

6.5 Verification results for signatures created by UPC

	Verified by				
Testcase-ID	Baltimore	IAIK	KOPDAT	Microsoft	remarks
XAdES#1	✓	✓			
XAdES#2	✓	✓			
XAdES#3	✓[11]	✓[10]			
XAdES#4	-	-	-	-	Not provided
XAdES#5	-	-	-	-	Not provided
XAdES#6	-	-	-	-	Not provided
XAdES#7	-	-	-	-	Not provided
XAdES#8	-	-	-	-	Not provided
XAdES-T#1		✓			
XAdES-C#1		✓			
XAdES-C#2					
XAdES-X#1		✓			
XAdES-X#2		✓			
XAdES-X#3					
XAdES-X-L#1		✓			
XAdES-X-L#2		✓			
XAdES-X-L#3					
XAdES-A#1		✓			
XAdES-A#2		✓			
XAdES-A#3					
XAdES-A#5					

[11] This signature does not contain an attribute certificate within the CertifiedRole elements

6.6 Verification results for signatures created by SK

	Verified by				
Testcase-ID	Baltimore	IAIK	Microsoft	UPC	remarks
DigiDoc_#1		✓		✓	
DigiDoc_#1		✓		✓	

SK provided their own document format, because they were not able to produce the defined test cases with their implementation.

7 Input for the XAdES Maintenance Process

In the preparation of the XAdES-PLUGTESTS™ event some issues of the XAdES specification were brought up by different implementers. These issues were discussed during the interoperability event and have been incorporated into a document giving proposals for the maintenance process of the XAdES specification.

In the following sections the different issues are discussed in detail.

7.1 Issue #1 – `<EncapsulatedOCSPValues>`

7.1.1 Problem Description

In the section 7.6.2 of the XAdES specification [1] it says:

> *OCSP Responses (`OCSPValues`) consist of a sequence of at least one OCSP Response. The* `<EncapsulatedOCSPValue>` *element contains the base64 encoding of a DER-encoded OCSP Response.* [1, section 7.6.2]

During the XAdES-PLUGTESTS™ it turned out that this section has been interpreted differently by the participating implementers in terms of what the actual content of the `<EncapsulatedOCSPValue>` has to bee. Some implementers included the whole `OCSPResponse` others have just included the `BasicOCSPResponse` (contained in the `ResponseBytes` of the `OCSPResponse` as defined in RFC2560 [3]). Therefore, the specification should be more explicit about what to include into the `<EncapsulatedOCSPValue>` element.

7.1.2 Resolution Proposal

Since the additional information that is provided by the `OCSPResponse` is not needed to be archived, it was first suggested to include the `BasicOCSPResponse`. The different possibilities are:

- `OCSPResponse`: On the one hand, the additional information provided by the `OCSPResponse`—an integer value indicating if the request was successful—is not needed to be archived, however, this is how the actual version of the specification is to be interpreted most likely. On the other hand, the information provided by the `<OCSPReferences>` element reflects the content of the `BasicOCSPResponse`. Therefore, any other OCSP response type than the `BasicOCSPResponse` has to be referenced by a `<OtherRef>` element, most likely.Thus, an OCSP response containing a different response type will have to be included into a `<OtherValue>` element.
- `ResponseBytes`: The `ResponseBytes` are already in DER-encoded format. They include an additional object identifier indicating the type of the included OCSP response. The Response Bytes may again contain OCSP responses of different types. Therefore, the same arguments apply, as for the `OCSPResponse` stated in the paragraph above.

- `BasicOCSPResponse`: The `BasicOCSPResponse` contains exactly the data that needs to be archived and corresponds to the information provided by the `<OCSPRef>` element.

At the interop the participants agrred to use `OCSPResponse`, since this is basically what the standards said, and furthermore the only deployed implementation in Estonia uses that interpretation.

7.2 Issue #2 – `<TimeStampType>` Data Type

This problem was identified by most implementers throughout the implementation process and already discussed in advance of the XAdES-PLUGTESTS™ event.

7.2.1 Problem Description

The specification of the `<TimeStampType>` data type is broken in two ways:

1. While it is easy to verify the time-stamp by processing all `<HashDataInfo>` elements and comparing the resulting hash value to the hash value stored in the time-stamp, it is difficult, time-consuming and possibly even infeasible in the general case to verify, if the time-stamp is applied exactly on the data that is claimed by the XAdES specification. That is, to verify if the time-stamp is applied on the elements that are claimed to be time-stamped.
2. For the `<AllDataObjectsTimeStamp>`, `<IndividualDataObjects-TimeStamp>` and the `<ArchiveTimeStamp>` `<HashDataInfo>` elements have to be composed that resolve to exactly the same data as the corresponding `<ds:Reference>`s in the `<ds:SignedInfo>` do. In the general case it is difficult or probably infeasible to compose such a reference, because the result of resolving depends on the context (e.g. the node it is contained in).

7.2.2 Remarks

The input for the different time-stamps used in the current XAdES version is formed by means of `<HashDataInfo>` elements. These `<HashDataInfo>` elements have to be processed according to the reference processing model specified in the XMLDSig specificaion [4]. This is, in short, resolving the provided URI in the URI-attribute of the `<HashDataInfo>` element, applying the transforms that are specified by the optional `<Transforms>` child element of the `<HashDataInfo>` element and finally canonicalizing the result, if the output of the last transform (or the result of resolving the URI, if there is no transform at all) is a node list. This means that the result of processing one `<HashDataInfo>` element is octet data in any case. The resulting octets of all the included `<HashDataInfo>` elements are then concatenated in the order the `<HashDataInfos>` appear in the document to form the input for the time-stamp. These resulting octets are in fact the information that is time-stamped.

The current version of XAdES specification therefore mandates what the result of processing an `<HashDataInfo>` elements has to be. In the definition of the `<SignatureTimeStamp>` property it says for instance:

> *The* <ds:SignatureTimeStamp> *element contains a single* <HashDataInfo> *element that refers to the* <ds:SignatureValue> *element of the XMLDSig signature. That is, the input for the time-stamp hash computation is the* <ds:SignatureValue> *XML element. [1, section 7.3.1]*

A verifying application has to make sure that the time-stamp has been applied on the proper input data. This is, to verify somehow that processing the <HashDataInfo> element results in the data that is claimed by the XAdES specification. In case of the <SignatureTimeStamp> for instance, this is the <ds:SignatureValue> element. Thus, the verifying application has to check that the octets that are being time-stamped are a valid representation of the <ds:SignatureValue> element.

As an URI and an arbitrary number of transforms can be used to compose such a <HashDataInfo> element, it is infeasible to deduce from the specified URI and the given transforms to the result, in the general case. Thus, the only way to verify what has been time-stamped is to process the <HashDataInfo> element and analyze the result.

As one XML structure can have any number of different octet data representations that bear the same information, canonicalization has been introduced. Thus, the only practical way to verify the timestamp input is to compare the canonicalized form of the data that has to be time-stamped according To the specification with the data that results from processing the corresponding <HashDataInfo> element. In this case it would be sufficient to simply create the required input for the time-stamp, compute the digest value and compare it with the digest value in the time-stamp. However, the <HashDataInfo> element was introduced to identify the input of a given time-stamp in cases where the input is ambiguous, but it does not serve this purpose anyway because the actual input can not be identified by just looking at the URI and the transforms included in the <HashDataInfo> element, as has been shown above

Therefore, a new solution has to be found to identify the input-data of a given time-stamp in cases were this input cannot be unambiguously defined by the XAdES specification.

7.2.3 Resolution Proposal

During the interoperability event the following resolution proposal was discussed and agreed on:

The <TimeStampType> data type should be redefined to use an ID-list to identify the elements that have been time-stamped. An optional <ds:CanonicalizationMethod> element should indicate which canonicalization method to use for canonicalizing XML elements. If no canonicalization method is specified the standard canonicalization method as specified by the actual XMLDSig specification MUST be used.

In the case of included <ds:Reference> elements an additional referencedData-attribute indicates if the <ds:Reference> element itself or the data resulting from processing the <ds:Reference> should be included. If the referencedData-attribute is omitted or the attribute value is false the element identified by the included

URI is included. If the `referencedData`-attribute value is true the `<ds:Reference>` has to be processed according to the reference processing model of the XMLDSig specification. The result is then used as input for the time-stamp. The result of the processing must be exactly the same data as that was used in the computation of the `<ds:Reference>` digest value.

```
<xsd:element name="TimeStamp" type="TimeStampType"/>
<xsd:complexType name="TimeStampType">
      <xsd:sequence>
            <xsd:element name="Include" type="IncludeType" maxOccurs="unbounded"/>
            <xsd:element ref="ds:CanonicalizationMethod" minOccurs="0"/>
            <xsd:choice>
                  <xsd:element name="EncapsulatedTimeStamp">
                  type="EncapsulatedPKIDataType"/>
                  <xsd:element name="XMLTimeStamp" type="AnyType"/>
            </xsd:choice>
      </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="IncludeType">
      <xsd:attribute name="uri" type="xsd:anyURI" use="required"/>
      <xsd:attribute name="referencedData" type="xsd:boolean" use="optional"/>
</xsd:complexType>
```

7.3 Issue #3 – <ArchiveTimeStamp>

7.3.1 Problem Description

The `<ArchiveTimeStamp>` definition is broken in two ways:

1. The `<ArchiveTimeStamp>` includes the `<SignedPropertiesElement>` twice.
2. The references to the `<SignedSignatureProperties>` and the `<SignedDataObjectProperties>` cannot be composed using ID-references, because these elements do not have an `xsd:ID`-attribute.

In section 7.7.1 of the XAdES specification [1] it says:

> *The XAdES* `<ArchiveTimeStamp>` *element contains the following sequence of Hash-DataInfo elements:*
>
> - *One <HashDataInfo> element for each data object signed by the XMLDSIG signature The result of application of the transforms specified each <HashDataInfo> must be exactly the same as the octet stream that was originally used for computing the digest value of the corresponding* `<ds:Reference>`*.*
> - *One <HashDataInfo> element for the <ds:SignedInfo> element. The result of application of the transforms specified in this <HashDataInfo> must*

> *be exactly the same as the octet stream that was originally used for computing the signature value of the XMLDSIG signature.*

- *One <HashDataInfo> element for the <SignedSignatureProperties> element.*
- *One <HashDataInfo> element for the <SignedDataObjectProperties> element.*
- *…*

In the first paragraph it says to include a `<HashDataInfo>` element for each `<ds:Reference>` in the XMLDSig signature. This obviously includes the reference to the `<SignedProperties>`. In the third and the fourth paragraph it says to include a `<HashDataInfo>` element for the `<SignedSignatureProperties>` and the `<SignedDataObjectProperties>`. These elements are already included by the reference to the `<SignedProperties>`. Additionally these two elements have no xsd:ID-attribute specified, thus they cannot be referenced using ID-references.

7.3.2 Resolution Proposal

Omit the `<HashDataInfo>` elements for the `<SignedSignatureProperties>` and the `<SignedDataObjectProperties>`. Additionally,

- either add an `<HashDataInfo>` element for the `<SignedProperties>` and omit the `<ds:Reference>` to the `<SignedProperites>`,
- or simply leave the `<ds:Reference>` to the signed properties included.

Add xsd:ID-attributes to the `<SignedSignatureProperties>` and the `<SignedDataObjectProperties>` elements as well as to the `<UnsigendSignatureProperties>` and the `<UnsignedDataObjectProperties>` elements.

7.4 Issue #4 – Requirement Levels (RFC2119)

Within the current version of the XAdES specification, the word “must” is used to indicate a requirement at several places and should therefore say “MUST” according to RFC2119 [5]. The RFC2119 defines how the key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” are to be interpreted in the sense of requirement level. Therefore, the specification should use these key words wherever a requirement is stated.

XAdES specification [1], section 5, first paragraph:

> *The XML namespace URI that **must** be used by implementations of the present document . . . [1, section 5]*

XAdES specification [1], section 6.2, second paragraph:

> *. . . The <SignedProperties> **must** be covered by a Reference element of the XML signature. Alignment with the present document mandates that one <SignedProperties> element MUST exist. [1, secion 6.2]*

XAdES specification [1], section 6.3, second paragraph:

> *However, the following restrictions apply for using <ds:Object>, <QualifyingProperties> and <QualifyingPropertiesReference>:*
>
> *• . . .*
>
> *• All signed properties **must** occur within a single <QualifyingProperties> element. This element can either be a child of the <ds:Object> element (direct incorporation), or it can be referenced by a <QualifyingPropertiesReference> element. See clause 6.3.1 for information how to sign properties.*
>
> *• . . .*

XAdES specification [1], section 7.2.5, last paragraph:

> *At least one element of <Description>, <ObjectIdentifier> and xmlMimeType **must** be present within the property. [1, section 7.2.5]*

XAdES specification [1], section 7.2.8, paragraph 8:

> *. . . At least one of the two elements <ClaimedRoles> or <CertifiedRoles> **must** be present. [1, section 7.2.8]*

XAdES specification [1], section 7.7.1, paragraph 10:

> *The <XAdESArchiveTimeStamp> element contains the following sequence of <HashDataInfo> elements:*
>
> *• One <HashDataInfo> element for each data object signed by the XMLDSig signature. The result of application of the transforms specified each <HashDataInfo> **must** be exactly the same as the octet stream that was originally used for computing the digest value of the corresponding <ds:Reference>.*
>
> *• . . .*

7.5 Issue #5 – <QualityingProperties>

Section 6.2 of the XAdES specification [1] says: “The mandatory Target attribute refers to the XML signature.” This should be changed to: “The mandatory Target-attribute :::::: MUST refer to the <Id>-attribute of the corresponding <ds:Signature>.”

7.6 Issue #6 – ASN.1 Encoding

For some ASN.1 PKI elements that are included into the XAdES signature the exact ASN.1 encoding mechanism is not specified (sections 7.1 and 7.2.8 of the XAdES specification [1]). This should be changed to mandate the DER (Distinguished Encoding Rules) encoding mechanism wherever an ASN.1 encoding is required.

7.7 Issue #7 – Trust Status Lists

The following proposal was made by members of the ETSI Technical Committee ESI (Electronic Signatures and Infrastructures):

> XAdES should probably be able to include Trust Status Lists (TSL [6]), beside certification and revocation information in future versions of the specification.

7.8 Issue #8 – <SigningCertificate>

In XAdES specification [1] section 7.2.2, last but one paragraph it says:

> *If the signer uses an attribute certificate to associate a role with the electronic signature, such a certificate MUST be present in the <SignerRole> property. [1, section 7.2.2]*

This sentence should be moved to section 7.2.8 "The `<SignerRole>` element" of the XAdES specification.

7.9 Issue #9 – XAdES forms

The following proposal was made by members of the ETSI Technical Committee ESI (Electronic Signatures and Infrastructures):

> *In future versions of the XAdES it should be possible to have archival versions 'references only', 'values only' and 'mixed'.*

Currently, the XAdES specification mandates to include references to the certification and revocation information as well as the actual certification and revocation values in the XAdES-X-L and XAdES-A forms. For the purpose of archiving all information necessary to validate the signature at a later time it would however be sufficient to just include the actual certification and revocation values and omit the references. Therefore the standard should provide forms to include only the necessary information to avoid redundancies.

7.10 Issue #10 – archival forms

The following proposal was made by members of the ETSI Technical Committee ESI (Electronic Signatures and Infrastructures):

> *It should be possible in future versions of XAdES to have archival versions that build on XMLDSig signatures without the mandatory <SignedProperties>.*

With the current XAdES versions it is not possible to create valid XAdES-A archival versions out of a plain XMLDSig signature, because the mandatory `<SignedProperties>` cannot be added to the signature later. The XAdES specification should therefore provide forms that permit XAdES-A versions without the currently mandatory `<SigningTime>`, `<SigningCertificate>` and `<SignaturePolicyIdentifier>` properties.

7.11 Issue #11 – <AnyType> Data Type

In the actual version of the XAdES specification [1] the `<AnyType>` data type is defined as follows:

```
<xsd:complexType name="AnyType" mixed="true">
      <xsd:sequence>
            <xsd:any namespace="##any"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##any"/>
</xsd:complexType>
```

This definition does not allow content that has no schema associated. Therefore the definition of the `<AnyType>` data type should read like the following:

```
<xsd:complexType name="AnyType" mixed="true">
      <xsd:sequence>
            <xsd:any namespace="##any" processContents="lax"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##any"/>
</xsd:complexType>
```

7.12 Issue #12 – <CertID>

In the current version of the XAdES specification [1] the `<CertID>` element does not have an `URIattribute` for pointing to an archived version of the referenced certificate:

```
<xsd:complexType name="CertIDType">
<xsd:sequence>
<xsd:element name="CertDigest" type="DigestAlgAndValueType"/>
<xsd:element name="IssuerSerial" type="ds:X509IssuerSerialType"/>
</xsd:sequence>
</xsd:complexType>
```

Therefore the definition of the `<CertID>` element should read like the following to allow pointing to an archived version of the certificate:

```
<xsd:complexType name="CertIDType">
      <xsd:sequence>
            <xsd:element name="CertDigest" type="DigestAlgAndValueType"/>
            <xsd:element name="IssuerSerial" type="ds:X509IssuerSerialType"/>
      </xsd:sequence>
      <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
</xsd:complexType>
```

7.13 Issue #13 – .NET validating parser

The Microsoft .NET validating XML parser fails to parse the current version of the XAdES schema, although the schema has been validated using the schema validating

tools provided by the World Wide Web Consortium (W3C). In order to reach a larger community this issue should be fixed in future versions of the XAdES specification.

7.14 Issue #14 – XAdES schema

In the actual version of the XAdES schema which is part of the XAdES specification the import statement for the XMLDSig schema is missing. Since elements from the XMLDSig schema are referenced by the XAdES schema an import statement has to be present. Therefore the XAdES schema should read like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://uri.etsi.org/01903/v1.1.1#"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns="http://uri.etsi.org/01903/v1.1.1#"
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
      elementFormDefault="qualified">

<xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
      schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig
      -core-schema.xsd"/>
```

7.15 Issue #15 – <QualifyingPropertiesReferenceType> data type

The `<QualifyingPropertiesReferenceType>` data type introduces a new `<Transforms>` element in the XAdES namespace for the `<ds:TransformsType>` rather than using a reference to the element type defined in the XMLDSig schema.

The current XAdES schema definition for the `<QualifyingPropertiesReferenceType>` data type is:

```
<xsd:complexType name="QualifyingPropertiesReferenceType">
      <xsd:sequence>
            <xsd:element name="Transforms" type="ds:TransformsType"
            minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="URI" type="xsd:anyURI" use="required"/>
      <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

This should be changed to:

```
<xsd:complexType name="QualifyingPropertiesReferenceType">
      <xsd:sequence>
            <xsd:element ref="ds:Transforms" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="URI" type="xsd:anyURI" use="required"/>
      <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

7.16 Issue #16 – XAdES examples

The XAdES examples in the (non-normative) annex D of the current version of the XAdES specification [1] are not aligned with the specification. These examples should be fixed, or probably replaced by examples produced as test cases for the XAdES-PLUGTESTS™ event.

7.17 Issue #17 – <DataObjectFormat>

In the XAdES specification [1], section 7.2.5, second paragraph it says:

> *. . . This (the <DataObjectFormat>) is a signed property that qualifies one specific signed data object. In consequence, an XML electronic signature aligned with the present document MAY contain more than one <DataObjectFormat> elements, each one qualifying one signed data object. [1, section 7.2.5, second paragraph]*

However, later in the same section the specification speaks about signed data object(s), suggesting that one <DataObjectFormat> applies for more than one signed data object, which it actually does not:

> *This element can convey:*
>
> - *Textual information related to the signed data object(s) in element <Description>;*
> - *An identifier indicating the type of the signed data object(s) in element <ObjectIdentifier>;*
> - *An indication of the MIME type of the signed data object(s), in element <MimeType>;*
> - *An indication of the encoding format of the signed data object(s), in element <Encoding>.*

This should be changed to say “object” wherever it says “object(s)”.

Additionally, in XAdES specification [1], section 7.2.4, fourth paragraph it says:

> *The mandatory ObjectReference attribute refers to the Reference element of the <ds:Signature> corresponding with the data object qualified by this property. [1, section 7.2.5, fourth paragraph]*

This should be changed to say

> *The mandatory QbjectReference attribute MUST reference the <ds:Reference> element of the <ds:Signature> corresponding with the data object qualified by this property.*

in order to indicate that this is a requirement according to RFC2119 [5].

Additionally, the current version of the XAdES specification mandates the <DataObjectFormat> element to be present when the signed data objects have to be presented to the verifier. In the XAdES specification [1] it says:

> *. . . This element (the <DataObjectFormat>) MUST be present when it is mandatory to present the signed data object to human users on verification. . . .[1, section 7.2.5, second paragraph]*

The first question is, does it make any sense to mandate the presentation of the signed data objects on verification, at all? Additionally, if it makes sense to mandate the presentation on verification, the data format may be defined implicitly by the application or desired use case, any way.

This issue needs further discussion.

7.18 Issue #18 – <CertificateValues>

7.18.1 Problem Description

On the one side the XAdES specification [1] says in section 7.6.1, third paragraph:

> *In principle, the <CERTIFICATEVALUES> element contains the full set of certificates that have been used to validate the electronic signature, including the signer's certificate. However, it is not necessary to include one of those certificates into this property, if the certificate is already present in the <ds:KeyInfo> element of the signature. [1, section 7.6.1]*

On the other side the <ds:KeyInfo> element is not covered by the <ArchiveTimeStamp>(s). That is, certificates that are present in the <ds:KeyInfo> and are not included into the <Certificatevalues> are not time-stamped for archiving purposes.

7.18.2 Resolution Proposal

There are two possible solutions to this issue:

- Mandate the inclusion of all certificates in the certificate chain into the <CertificateValues> element.
- Mandate to include the <ds:KeyInfo> element into the <ArchiveTimeStamp>(s).

This issue needs further discussion.

7.18.3 Issue #19 – <CompleteCertificateRefs>

In the section 7.4.1 of the XAdES specification it says:

> *The <CertRefs> element contains a sequence of <Cert> elements already defined in clause 7.2.2, incorporating the digest of each certificate and optionally the issuer and serial number identifier. [1, section 7.4.1, last paragraph]*

However, the XAdES schema mandates the issuer and serial number identifier to be present in the <Cert> element. Therefore the word "optionally" should be removed from the quoted sentence above.

8 Feedback by participants

We have asked the interop-participants to give us their feedback on the event. Here is what they have said:

8.1 Tarvi Martens, Estonia, Sertifitseerimiskeskus

Estonia was happy to have the opportunity to present its implementation of XAdES-based product line called DigiDoc. With over year of free public availability to 330 000 Estonian ID-card users and IT application integrators, the DigiDoc technology base has matured enough to prove consistency of XAdES-based digital signature technology, to be a valuable feedback for XAdES standard development efforts and serve as an example for other XAdES-implementers.

Current XAdES standard implies indirectly certain model of PKI infrastructure for providing for digital signatures for long-term validity. As with Estonian case the PKI infrastructure model is different - it is simpler, but achieves the same goals or even goes beyond them. As of result, some of "required" blocks and elements of XAdES specification are skipped.

The most significant result of the event from Estonia's point of view was to agree on principle for further standard development where the XAdES specification should NOT be dealing with assumptions for PKI Infrastructure, trust model or verification means - XAdES should be dealing with its building blocks and integral integrity of thereof. We took the responsibility to propose constructive criticism and proposals for the XAdES specification development.

From the other side we got valuable feedback about some syntactical glitches in our current DigiDoc/XAdES document format. We will include all the fixes in the next version of our product line to be as compliant to XAdES as possible.

As the maturity of XAdES implementations is currently pretty low, there is excellent opportunity window to achieve common understanding of XML-based digital signatures. The XAdES specification itself is very young and was not actually based on practical implementations - this means that besides implementation interoperability, further development of XAdES specification itself should be seriously addressed based on real-life feedback.

8.2 Juan Carlos Cruellas, UPC

The XAdES Interoperability Event organized and hosted by ETSI will prove to have been a major milestone in the widespread of the usage of XML advanced electronic signatures aligned with the ETSI TS 101 903 technical specification.

The implementers had opportunity to have face to face meetings where they could share their views, confront their interpretations, solve their doubts, assess the feasibility of the specification, and even raise relevant suggestions to be taken into account for next versions.

The major outcomes of the event are of different types.

As implementer party, the face to face interaction with the rest of implementers allowed immediate detection of required changes in the tools.

As editor of the TS and person involved in its review process, the technical discussions maintained with people present in the event on different aspects of the specification, have provided me with a very valuable list of change suggestions that without no doubt will improve the standard and will make it first more flexible (which implies targeting wider communities) and second easier to use. As a perfect example of this, I would like to mention the suggestion for changing the time-stamps management mechanism defined in XAdES.

Finally, the maintenance of a web page, hosted by participant(s) of the event, will allow us to keep a high rate of development and a fast interaction among the current implementers, and will provide an extremely valuable input for new future implementers.

8.3 Vivekanand Sakaram, Baltimore Technologies

In my opinion, the event went well and personally it was a great experience to me. However, certain improvements could be made by ETSI when an interop event is conducted the next time. To start with, XAdES interop could have been held in 3 days may be instead of 5 days should it have happened when there were a good number of vendors having implemented the standard, more or less, completely. Having said that, this was my first interop event and the 4 days I stayed gave me enough time to learn more about XAdES, collaborate with fellow participants and getting them to know more.

When I left Dublin, I wasn't sure what to expect. I was introduced to XAdES just a week ago by my colleague and was going through the standard and implementation. However, the participants of the event made me feel very comfortable and were very welcoming and I got clarified many of the concepts in relation to xml/XAdES signatures. The opportunity to dedicate my full-time completely towards XML signatures, without any distractions and interruptions, was something that I cherish even today.

I must also say the support and help provided by ETSI in relation to accommodation, taxi and other was very helpful. As a recommendation, I would request ETSI to list the hotels in the order of proximity to the venue of the interop event may be as I booked mine without no knowledge of it and with not great transportation facility, it could have been very hard but for the lifts from the participants. My sincere thanks to all! Finally, I thoroughly enjoyed our time at the restaurants in the evenings. It proved to be a great podium for sharing ideas, jokes and also helped me to get a peek at different cultures.

8.4 Eddy Rubens, Microsoft

The Microsoft implementation comes in the form of a library designed to be used by developers. The intention is to facilitate building solutions that adhere to the XAdES standard. The status for the library is to be considered "a work in progress" with a release date early 2004.

From a personal point of view, attending the event was a nice experience and interacting with the other implementers was enriching and enlightening. Some important flaws have been discovered during the event and sometimes corrected on the spot. A real

timesaver as some of these issues would have been hard to uncover without interoperability testing. The remaining issues are on top of my to-do list.

The venue and facilities were very good and the organizers can be complemented. The only drawback was that the main room had no direct view of the beautiful scenery outside, this could also been seen as a bonus because there were no distractions and there was a lot of work to be done.

I wholeheartedly support to keep the spirit of the event alive and keep the channels open to share test cases in the future.

8.5 Peter Lipp, IAIK

I believe the interop event was a success, even if it proved to be too early. This was due mostly because many implementations were not ripe enough and still needed and need some work to go into. Nevertheless, the event proved that interoperability was achieved even with the level of implementations available and, more importantly, gave excellent input to the maintenance process of the standard.

Besides this, meeting other groups of people working on implementations provided the basis of future co operations.

8.6 Martin Centner, IAIK

Following the discussions in the preparation and during the XAdES-PLUGTESTS event it turned out, that there is quite some interest by different software developers and solution providers in implementing the XAdES specification. It turned out as well, that the trust model implied by the XAdES specification—especially the strict sequence of the different XAdES forms—does not apply in all the considered situations and use cases. However, the different qualifying properties defined by the XAdES specification are considered as being quite useful in many different use cases. Future versions of the XAdES specification should take this into account and allow for a more flexible use of the different qualifying properties while still recommend the use of the defined XAdES forms to achieve certain goals and security levels respectively. That is, the different qualifying properties should be defined in the normative part of the XAdES specification, the different XAdES forms should be moved to the informative part of the XAdES specification and should be considered as recommendation or best practice.

9 Conclusions

After having been intensively working during five days in XAdES there is a wide range of conclusions of different types. Below follows a summary.

- Concerning the precise moment when the event took place, on one hand the implementations proved to need further developments to fulfil the whole functionality specified in the standard. This means that the interoperability tests on certain aspects were not as intensive as desired. On the other hand, this, which could be seen as an “early” event, has been a place where the different implementations have been tested with others. And by doing that NOW, misunderstandings of the specification have been clarified; errors in implementations have been corrected and not kept and propagated; doubts on how to deal with future developments, have been shared and solved; etc. In summary, after the event it could be said that there is a team of people that know not only XAdES specification, but the implementation implications very well. It can also be said that the implementations are now much more error-free in terms of alignment with the standard than before. Somehow, it could be said that this event has act as a catalyst for XAdES take-off.
- On the impact that this event has had for the XAdES implementations themselves, and for their implementers:
    a. The XAdES interop email list is still alive and active. During the interop event and afterwards, people have come and sent questions and comments. Somehow, this list is becoming a reference for those that are dealing with XAdES development.
    b. After the event, and in the view of the interest it had, one of the participants has offered to host a portal that could maintain the interoperability matrix, and associated documentation with XAdES activities. This portal could then attract the attention of more existing implementers that could first use the tests in there for assessing their implementations and second issue their own signatures for increasing the test matrix.
    c. Most of the participants in the event are currently participating in standardization forum dealing with standards that are close to XAdES. It would not be difficult at all imagine the organization of a future interop event that would bring together these complementary technologies.
- On the impact that the event will have on the XAdES standard itself.
    a. The event has taken place in the middle of the review of XAdES standard. These precise days, the standard is being updated and reviewed. The timing of the event (refer to the first point in this section) has allowed to feed this process with comments coming from actual implementations and to some extent, deployments. These comments have an added value when compared to those that come from readers.
    b. The long time that the participants had to spend together allowed the celebration of technical meetings where specific issues that could be

improved in a new version of the standard were deeply discussed. This report has listed a number of issues that are currently feeding the review process. Some of them will likely improve it and facilitate implementations development and wide spreading.

c. The presence in the event of implementations with different degrees of alignment with the standard has raised a very interesting point that the ESI group will have to face: the degree of openness of the standard. Some of the issues listed in the present document directly suggest adopting a more flexible strategy concerning to the compliance clauses in the specification. In this way, it could target a much more wide community of users, as it could also give satisfaction to environments where the requirements would not need one of the specific XAdES forms (as they are currently defined), BUT would need a different combination of qualifying properties already defined in XAdES. Proposals in this document are made so that this degree of flexibility can be achieved in future versions of XAdES if they are accepted by ESI.

In a nutshell the event has been a very useful one. We expect it to be the catalyst for the take-off of XAdES by both accelerating the development of the applications and making the standard much more open and flexible. The cooperation that has been started will continue. We plan to continuously add new test cases to the portal which will be available for everybody. This will be helpful for other developers that have not been able to participate yet. We forsee meeting again next year for a new interop, if possible, one which will be based on the changes to XAdES resulting of this activity. .

10 Bibliography

[1] European Telecommunications Standards Institute (ETSI). *XML Advanced Electronic Signatures (XAdES), ETSI TS 101 903 V1.1.1*, February 2002. http://uri.etsi.org/01903/v1.1.1/ts_101903v010101p.pdf.

[2] Internet Engineering Task Force (IETF). *Internet X.509 Public Key Infrastructure: Time-StampProtocol (TSP), Standards Track RFC 3161*, August 2001. http://www.ietf.org/rfc/rfc3161.txt.

[3] Internet Engineering Task Force (IETF). *Internet X.509 Public Key Infrastructure: OnlineCertificate Status Protocol (OCSP), Standards Track RFC 2560*, June 1999. http://www.ietf.org/rfc/rfc2560.txt.

[4] World Wide Web Consortium (W3C). *XML – Signature Syntax and Processing (XMLDSig),W3C Note*, February 12, 2002. http://www.w3.org/TR/xmldsig-core/.

[5] Internet Engineering Task Force (IETF). *Key words for use in RFCs to Indicate RequirementLevels, Best Current Practice RFC 2119*, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

[6] European Telecommunications Standards Institute (ETSI). Requirements for Trust ServiceProvider status information, ETSI TS STF 220-1 V0.1.5, December 2002.