NATIONAL SECURITY AUTHORITY

Version 3.0

Qualified Electronic Signature Formats

12 August 2009

NATIONAL SECURITY AUTHORITY

Information Security and Electronic Signature Department

Budatínska 30, 850 07 Bratislava 57

http://www.nbusr.sk/

E-mail: sep@nbusr.sk

Content

1 Introduction

The unambiguous visualization of the electronic document being signed and verified is one of the requirements which is significant in qualified electronic signature (hereinafter referred to as QES) creation and verification. The principal requirement is to create the signature of this document by private key being stored in SSCD whose corresponding public key was stored in the qualified certificate being issued by accredited certification authority. In order to verify QES unambiguously it is necessary to define unambiguous rules for signature format and contents and the type identification of the electronic document being signed with respect to its unambiguous visualization need.

2 Scope

The standard „Qualified Electronic Signature Formats " is issued in accordance with Article 3(6) of the National Security Authority (hereinafter referred to as the NSA) Decree No.135/2009 Coll. on the format and method of creating the qualified electronic signature, the method of publishing the public key of the National Security Authority, the conditions of validity for the qualified electronic signature, procedure during the verification and verification conditions of the qualified electronic signature, format of the time stamp and method of creating it, requirements on the source of time data and requirements for keeping time stamp documentation (on the creation and verification of the electronic signature and time stamp).

The purpose of the present standard is to determine technical requirements for particular types of qualified electronic signatures to ensure the compatibility and unified environment of the electronic signature in the Slovak Republic with respect to electronic signature environment within the European Union where Member States shall ensure that Qualified Electronic Signatures (QES – CWA 14170) on the basis of Article 5 (1) of the European Directive 1999/93/EC [26] meet the following:

Legal effects of electronic signatures
Member States shall ensure that advanced electronic signatures which are based on a qualified certificate and which are created by a secure-signature-creation-device:

a) satisfy the same legal requirements of the signature in relation to data in electronic form in the same manner as a hand-written signature satisfies those requirements in relation to paper-based data;
b) are admissible as evidence in legal proceedings.

3 References

References to documents defining used types and methods.

[1] ETSI	TS 101 733 Electronic Signature Formats (CAdES)	
[2] ETSI	TR 102 272 ASN.1 format for signature policies	
[3] RFC 5280	X.509 PKI Certificate and Certificate Revocation List	5-2008
[4] RFC 3739	Qualified Certificates Profile	3-2004
[5] ETSI	TS 101 862 Qualified Certificate Profile	
[6] RFC 5652	Cryptographic Message Syntax	9-2009
[7] RFC 3161	Time-Stamp Protocol (TSP)	8-2001
[8] RFC 2560	X.509 PKI Online Certificate Status Protocol	8-1999
[9] NSA	Formats of certificates and qualified certificates	
[10] NSA Decree No.135/2009 Coll. on the format and method of QES creation		
[11] ETSI	TS 102 280 X.509 V.3 Cert. Profile for Cert. Issued to Natural Persons	
[12] ETSI	TR 102 437 Guidance on TS 101 456	
[13] ETSI	TS 101 456 Policy Requirements for cert. authorities issuing qualified cert.	
[14] ETSI	TS 102 042 Policy Requirements for cert. authorities issuing public key cert.	
[15] ETSI	TS 102 231 Provision of harmonized Trust-service status information	3-2006
[16] ETSI	TS 101 903 XML Advanced Electronic Signatures (XAdES)	
[17] RFC 5035	Enhanced Security Services (ESS) Update	8-2007
[18] ITU-T	RECOMMENDATION X.509 (08/2005) | ISO/IEC 9594-8:2005	
[19] RFC 3548	The Base16, Base32, and Base64 Data Encodings	7-2003
[20] ISO/IEC	3166 Codes for the representation of countries	
[21] NSA	Specifying the content and formal specifications of document formats for QES	
[22] CEN/CWA	14170 Security Requirements for Signature Creation Application	
[23] NSA	Certification Path Control	
[24] RFC 2046	MIME Part Two: Media Types	11-1996
[25] CEN/CWA	14171 General guidelines for electronic signature verification	

[26] Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures

Notice 1: Note 1:Current version of the Directive 1999/93/EC is available on the web page: http://eur-lex.europa.eu/smartapi/cgi/sga_doc?smartapi!celexapi!prod!CELEXnumdoc&lg=en&model=guicheti&numdoc=31999L0093

4 Abbreviations

ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation 1
CA	Certification Authority
CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
DER	Distinguished Encoding Rules (for ASN.1)
DTD	Document Type Definition
EPES	Explicit Policy-based Electronic Signature
ES-A	Electronic Signature with Archive validation
ES-C	Electronic Signature with Complete validation data
ES-T	Electronic Signature with Time-stamp data
ESS	Enhanced Security Services (enhances CMS)
GMT	Greenwich Mean Time
HTTP	Hyper Text Transfer Protocol
ISO	International Organization for Standardization
MIME	Multipurpose Internet Mail Extensions
OCSP	Online Certificate Status Protocol
OID	Object IDentifier
PKCS	Public Key Cryptographic Standards, Standards published by RSA, Labs.
RIPEMD-160	Race Integrity Primitives Evaluation Message Digest 160
PKIX	internet X.509 Public Key Infrastructure
QC	Qualified Certificate
RSA	Rivest, Shamir and Adleman Algorithm
S/MIME	Secure/Multipurpose Internet Mail Extensions
SHA-1	Secure Hash Algorithm 1
SSCD	Secure-Signature-Creation Device
TSA	Time-Stamping Authorities
TSP	Time Stamp Protocol
TST	Time-Stamp Token
URL	Uniform Resource Locator
XAdES	XML Advanced Electronic Signature
XML	eXtensible Markup Language
QES	Qualified Electronic Signature

5 Rules for QES structures over AdES

5.1 Signature in ASN.1 format

Table 1 CMS envelope of the electronic signature

	Record in ASN.1	Short description
1.	`ContentInfo ::= SEQUENCE {`	CMS envelope, in this case of the signature.
2.	`contentType ContentType,`	OID of data type that is contained in *content.*
3.	`content [0] EXPLICIT ANY DEFINED BY contentType }`	Data identified by OID.
4.	`ContentType ::= OBJECT IDENTIFIER`	OID identifying the type.
5.	`id-signedData OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2 }`	OID identifying the signed data type *SignedData*.

Table 2 CMS type –SignedData

	Record in ASN.1	Short description
1.	`SignedData ::= SEQUENCE {`	Signed data identified by *id-signedData.*
2.	`version CMSVersion,`	CMS syntax version.
3.	`digestAlgorithms DigestAlgorithmIdentifiers,`	A set of algorithm identifiers that are used to calculate the hash from the document being signed.
4.	`encapContentInfo EncapsulatedContentInfo,`	Data from which the hash is calculated • Internal signature: OID of data and data themselves • External signature: it contains only OID of data (RFC 5652, section 5.2).
5.	`certificates [0] IMPLICIT CertificateSet OPTIONAL,`	A set of certificates. It must contain at least a certificate(s) of a signer(s). If the entire paths are not stored in the attribute (Table 14), then in case of long-term verification of the signature, the entire certification paths are contained as well.
6.	`crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,`	A set of CRL. It does not have to be used if CRL is stored in the attribute (Table 15); otherwise it is required to be used.
7.	`signerInfos SignerInfos }`	A set of *SignerInfo*s with signatures of particular signers.
8.	`DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier`	A set of hash algorithms with their parameters defined by means of OID.
9.	`SignerInfos ::= SET OF SignerInfo`	A set of signatures.

Table 3 EncapsulatedContentInfo – inserted data being signed

	Record in ASN.1	Short description
1.	`EncapsulatedContentInfo ::= SEQUENCE`	It contains data that are being signed.
2.	`{ eContentType ContentType,`	OID of data type that are being signed. The *id-data* type (OID 1.2.840.113549.1.7.1) for binary non-interpretable data. It is recommended

		to use MIME envelope for data.
3.	`eContent [0] EXPLICIT OCTET STRING OPTIONAL }`	If *eContent* is missing, then the hash is calculated from external data. If it contains *eContent*, then the hash is calculated from the content of *OCTET STRING* without tag and size of *OCTET STRING*.

Table 4 SignerInfo – a signer's data with his/her signature

	Record in ASN.1	**Short description**
1.	`SignerInfo ::= SEQUENCE {`	The signer's data.
2.	`version   CMSVersion,`	Syntax version is defined in accordance with RFC 5652.
3.	`sid       SignerIdentifier,`	Not protected identification of the signer's certificate – informative. The identifier *IssuerAndSerialNumber* must be used because of compatibility with CAdES and QES signature. A real identifier is found directly in the attributes being signed (signingCertificate/V2) to prevent the substitution attack.
4.	`digestAlgorithm DigestAlgorithmIdentifier,`	The identifier of the signer's hash algorithm by which the hash is calculated and subsequently used for signing. One of the algorithms from the Table 2, line 3.
5.	`signedAttrs [0] IMPLICIT SignedAttributes OPTIONAL,`	A set of attributes being signed. Must be present.
6.	`signatureAlgorithm SignatureAlgorithmIdentifier,`	The algorithm identifier whose private key is owned by a signer. For example *RsaEncryption(1.2.840.113549.1.1.1)*
7.	`signature SignatureValue,`	A digital signature.
8.	`unsignedAttrs [1] IMPLICIT UnsignedAttributes OPTIONAL }`	A set of attributes being unsigned. It contains e.g. a time stamp, ETSI attributes, etc.
9.	`SignerIdentifier ::= CHOICE {`	The identifier of the signer's certificate.
10.	`issuerAndSerialNumber IssuerAndSerialNumber,`	A name of the certificate issuer and serial number of the signer's certificate.
11.	`subjectKeyIdentifier [0] IMPLICIT SubjectKeyIdentifier }`	The identifier of the signer's public key from his/her certificate.
12.	`SignedAttributes ::= SET SIZE (1..MAX) OF Attribute`	A set of attributes being signed. It must be in DER coding.
13.	`UnsignedAttributes ::= SET SIZE (1..MAX) OF Attribute`	A set of attributes being unsigned. DER coding for all fields of the signature is recommended.
14.	`Attribute ::= SEQUENCE {`	The attribute consists of:
15.	`attrType OBJECT IDENTIFIER,`	OID of the identifier,
16.	`attrValues SET OF AttributeValue }`	and values according to OID.
17.	`AttributeValue ::= ANY`	It is defined in the following tables.
18.	`SignatureValue ::= OCTET STRING`	A signature.

Table 5 CMS attributes being signed and unsigned

	Record in ASN.1	**Short description**

1.	`id-contentType OBJECT IDENTIFIER ::= {1 2 840 113549 1 9 3}`
	`contentType ::= OBJECT IDENTIFIER`
		Usage: the attribute being signed which is used in attributes only once.
		OID of data type. It is the same as in *eContentType* and it must be found in attributes only once; e. g. *id-data* (1 2 840 113549 1 7 1)

2.	`id-messageDigest OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 4 }`
	`MessageDigest ::= OCTET STRING`
		Usage: the attribute being signed which is used in attributes only once.
		A hash is calculated from data according to procedure in Table 3, line 3.

3.	`id-signingTime OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 5 }`
	`SigningTime ::= Time`
	`Time ::= CHOICE {`
	` utcTime          UTCTime,`
	` generalizedTime GeneralizedTime`
	`}`
		Usage: the attribute being signed which is used in attributes only once.
		A time declared by a signer.
		From 1st January 1950 up to 31st December 2049 it must be coded in *UTCTime*, then in *GeneralizedTime* in a time zone Greenwich Mean Time (Zulu) and must contain seconds. Applications should know how to process both types:
		UTCTime Format*:* YYMMDDHHMMSSZ,
		GeneralizedTime (ETSI recommends): YYYYMMDDHHMMSSZ.

4.	`id-countersignature OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 6 }`
	`Countersignature ::= SignerInfo`
		Usage: in unsigned attributes only.
		It signs one or more signatures in a chain from DER coded attribute *signature* of *signatureValue* type from *SignerInfo*. It is not intended for QES, because it does not sign data. It is intended just for signing of superior digital signature in whose unsigned attributes it is contained. It is not defined for archive signature type. It is not recommended to be used.

Table 6 Other signing certificate

	Record in ASN.1	**Short description**
1.	`id-aa-ets-otherSigCert OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 2 19 }`	Usage: the attribute being signed which is used in attributes only once. It is not recommended to be used because it has been superseded by *ESS signing certificate v2*. OID identifying the attribute with the sequence of certificate identifiers from the signer's certificate up to the root certificate. It must contain the identifier of the signer's certificate as a minimum.

2.	`OtherSigningCertificate ::= SEQUENCE`
	`{ certs SEQUENCE OF OtherCertID,`
	`  policies SEQUENCE OF`
	`      PolicyInformation OPTIONAL`
	`-- NOT USED IN THE PRESENT DOCUMENT }`
		Certs – a sequence of certificate identifiers from the signer's certificate.
		Policies are not used.

3.	`OtherCertID ::= SEQUENCE {`
	`otherCertHash OtherHash,`
	`issuerSerial IssuerSerial OPTIONAL }`
		OtherCertHash – a hash of the certificate. If *issuerSerial* is indicated, then *directoryName* must be used in searching.

4.	`OtherHash ::= CHOICE {`
	`  sha1Hash OtherHashValue,`
	`      -- This contains a SHA-1 hash`
	`  otherHash OtherHashAlgAndValue}`
		Sha1 hash is recommended to be used because of compatibility unless another hash algorithm is required to be used.

	`OtherHashValue ::= OCTET STRING` `OtherHashAlgAndValue ::= SEQUENCE {` `hashAlgorithm AlgorithmIdentifier,` `hashValue OtherHashValue }`	

Table 7 CertificateRefs

	A.1.1.1 Record in ASN.1	Short description
1.	`id-aa-ets-certificateRefs OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 2 21}`	Usage: the attribute being unsigned which is used in attributes only once. OID identifying a complete list of certificate identifiers of the entire certification path without the signer’s certificate. - this field is only informative
2.	`CompleteCertificateRefs ::= SEQUENCE OF OtherCertID`	A complete list of certificate identifiers *OtherCertID* of the entire certification path that is used for signature verification of the signer’s certificate. The certification path is without the signer’s certificate. *OtherCertID* is defined in Table 6, line 3.

Table 8 RevocationRefs

	Record in ASN.1	Short description
1.	`id-aa-ets-revocationRefs OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 2 22}`	Usage: the attribute being unsigned which is used in attributes only once. OID identifying a complete list of CRL or OCSP identifiers which are used for verification of the signer’s certificate validity and the entire certification path, Table 7. - this field is only informative
2.	`CompleteRevocationRefs ::= SEQUENCE OF CrlOcspRef` `CrlOcspRef ::= SEQUENCE {` `crlids [0] CRLListID OPTIONAL,` `ocspids [1] OcspListID OPTIONAL,` `otherRev [2] OtherRevRefs OPTIONAL` `}`	A complete list of references to CRL or OCSP which starts by information about the verification of the signer’s certificate and finishes by trusted root CA exactly according to the path in Table 7. It is recommended to be used for the certificate identified by means of *OtherCertID.*
3.	`CRLListID ::= SEQUENCE {` `crls SEQUENCE OF CrlValidatedID` `}` `CrlValidatedID ::= SEQUENCE {` `crlHash OtherHash,` `crlIdentifier CrlIdentifier OPTIONAL` `}` `CrlIdentifier ::= SEQUENCE {` `crlissuer Name,` `crlIssuedTime UTCTime,` `crlNumber INTEGER OPTIONAL` `}` `OcspListID ::= SEQUENCE {` `ocspResponses SEQUENCE OF OcspResponsesID` `}` `OcspResponsesID ::= SEQUENCE {`	At least one *CRLListID* or *OcspListID* must be defined for each certificate from the certification path in Table 7. *crlIdentifier* must be defined in CRL identifiers. *crlNumber* does not have to be defined in *crlIdentifier*. *ocspRepHash* must be defined in *OcspResponsesID* identifiers.

	`ocspIdentifier OcspIdentifier,`
	`ocspRepHash OtherHash OPTIONAL`
	`}`
	`OcspIdentifier ::= SEQUENCE {`
	`ocspResponderID ResponderID,`
	`-- As in OCSP response data`
	`producedAt GeneralizedTime`
	`-- As in OCSP response data }`	

4.	`OtherRevRefs ::= SEQUENCE {`
	`otherRevRefType OtherRevRefType,`
	`otherRevRefs ANY DEFINED BY`
	`otherRevRefType}`
	`OtherRevRefType ::= OBJECT IDENTIFIER`
		It is not used.

Table 9 SignatureTimeStampToken

	Record in ASN.1	**Short description**
1.	`id-aa-signatureTimeStampToken OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 2 14 }`	Usage: the attribute being unsigned that can be used in attributes several times with various TSA. OID identifying a time stamp that stamps the signer's signature value, Table 4, line 7.

2.	`TimeStampToken ::= ContentInfo`
	`-- contentType is id-signedData`
	`-- content is SignedData`
	`-- eContentType within SignedData`
	`-- is id-ct-TSTInfo`
	`-- eContent within SignedData is`
	`-- TSTInfo`
	
	`TSTInfo ::= SEQUENCE {`
	`version INTEGER { v1(1) },`
	`policy TSAPolicyId,`
	`messageImprint MessageImprint,`
	`-- MUST have the same value as the`
	`-- similar field in TimeStampReq`
	`serialNumber INTEGER,`
	`-- Time-Stamping users MUST be`
	`-- ready to accommodate integers`
	`-- up to 160 bits.`
	`genTime GeneralizedTime,`
	`accuracy Accuracy OPTIONAL,`
	`ordering BOOLEAN DEFAULT FALSE,`
	`nonce INTEGER OPTIONAL,`
	`-- MUST be present if the similar`
	`-- field was present in`
	`-- TimeStampReq. In that case it`
	`-- MUST have the same value.`
	`tsa [0] GeneralName OPTIONAL,`
	`extensions [1] IMPLICIT Extensions`
	`OPTIONAL`
	`}`
	`Accuracy ::= SEQUENCE {`
	`seconds INTEGER OPTIONAL,`
	`millis [0] INTEGER (1..999)`
	`OPTIONAL,`
	`micros [1] INTEGER (1..999)`
	`OPTIONAL }`
		A value in *messageImprint* field within *TimeStampToken* is a hash calculated from the signer's *signature*, Table 4, line 7 which is inserted in SignerInfo in CMS type signedData that is time stamped.
		
		The time stamp is the only piece of information about the time inserted directly in the signature which can be trusted during the signing. It is not used only for obtaining the information about the time but also for securing the signature in case of compromising the signer's key. If the signature is time stamped before the key has been compromised, then it is considered to be valid because the time stamp proves the key validity before its compromising.
		
		CRL or OCSP for time stamp validity verification is appended to the time stamp and not to the signature whose components the time stamp has stamped.
		
		It is possible to time stamp any electronic document as well and thus to confirm its integrity before its signing time. Then in the signing process the time stamp is inserted into the attribute being signed *d-aa-ets-contentTimestamp*.
		More detailed description is provided by Annex D.

Table 10 EscTimeStamp

	Record in ASN.1	Short description
1.	`id-aa-ets-escTimeStamp OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 2 25}`	Usage: the attribute being unsigned that can be used in attributes several times with various TSA. OID identifying a time stamp whose purpose is to protect the electronic signature validity in case of compromising CA key and CA certificate revocation.
2.	`ESCTimeStampToken ::= TimeStampToken`	The time stamp stamps the hash that will be inserted in *messageImprint* field within *TimeStampToken.* The hash is calculated from concatenated values of fields (in DER without type and size of encoded value) from the following attributes: • *OCTETSTRING* from *SignatureValue* field within the SignerInfo; • signature-time-stamp, Table 9; • complete-certificate-references, Table 7; • complete-revocation-references, Table 8.

Table 11 CertCRLTimestamp

	Record in ASN.1	Short description
1.	`id-aa-ets-certCRLTimestamp OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 2 26}`	Usage: the attribute being unsigned that can be used in attributes several times with various TSA. OID identifying a time stamp whose purpose is to protect the electronic signature validity in case of compromising CA key and CA certificate revocation by stamping references to CA certificates and certificate revocation lists only.
2.	`TimestampedCertsCRLs ::= TimeStampToken`	The time stamp stamps the hash that will be inserted in *messageImprint* field within *TimeStampToken.* The hash is calculated from concatenated values of fields (in DER without type and size of encoded value) from the following attributes: • complete-certificate-references, Table 7; • complete-revocation-references, Table 8. It allows making the operation of TSA that issues time stamps for big organizations more effective. When CA issues a new CRL, it stamps a complete certificate list and CRL only once. Then CA will publish a stamped list which will be only appended to the signature by particular signers without the necessity of another stamping.

Table 12 ArchiveTimestamp

	Record in ASN.1	**Short description**
1.	`id-aa-ets-archiveTimestamp OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 2 48}`	Usage: the attribute being unsigned that can be used in attributes several times with various TSA. OID identifying a time stamp whose purpose is the long-term archiving of the electronic signature with complete certificates and data for validity verification in order to verify the electronic signature even if CA key is compromised or CA certificate is revoked or if CA services are unavailable or from other reasons, e. g. if the signature algorithm becomes less secure. Before expiration or revocation of the archive TSA certificate, in case of the long-term verification need, repeated appending of the new *archiveTimestamp* is required.
2.	`ArchiveTimeStampToken ::= TimeStampToken`	TSA stamps (signs) the hash with the algorithm of longer key length to secure the signature for a longer period. The signature will be inserted in *messageImprint* field within *TimeStampToken.* The hash is calculated from concatenated values of fields in such sequence as indicated in the signature, while *Attribute* fields will be used in such sequence as occur in unsigned attributes of the signature. The Annex D. Approved signature policy which is valid to the time of issuance of time stamp being verified is used in verification.

Table 13 SigningCertificate

	Record in ASN.1	**Short description**
1.	`id-aa-signingCertificate OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 2 12 }`	Usage: the attribute being signed that can be used in attributes only once. It is recommended to use *ESS signing certificate v2* instead of it, Table 20[17]. OID identifying the attribute with the sequence of certificate identifiers from the signer's certificate up to the root certificate. It must contain the identifier of the signer's certificate as a minimum.

2.	`SigningCertificate ::=  SEQUENCE {`
	`  certs     SEQUENCE OF ESSCertID,`
	`  policies  SEQUENCE OF`
	`      PolicyInformation OPTIONAL  }`
		Certs – a sequence of certificate identifiers from the signer's certificate.
		Policies are not used.

3.	`ESSCertID ::=  SEQUENCE {`
	`  certHash     Hash,`
	`  issuerSerial IssuerSerial OPTIONAL`
	`}`
		CertHash - a hash SHA-1 of the certificate. If *issuerSerial* is indicated, then *directoryName* must be used in searching.

4.	`Hash ::= OCTET STRING`
	
	`  -- SHA1 hash of entire certificate`
	`IssuerSerial ::= SEQUENCE {`
	` issuer     GeneralNames,`
		Only *sha1*hash from the entire certificate is used.

	`serialNumber CertificateSerialNumber}`	

Table 14 CertValues

	Record in ASN.1	Short description
1.	`id-aa-ets-certValues OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 2 23}`	Usage: the attribute being unsigned that can be used in attributes only once. OID identifying the certificate list of the entire certification path as a minimum without the signer's certificate, according to references defined in Table 7. It can also contain other certificates, e.g. cross certificates or certificates for verification of indirect CRL or OCSP responses.
2.	`CertificateValues ::= SEQUENCE OF Certificate`	A list of X.509 certificates.

Table 15 RevocationValues

	Record in ASN.1	Short description
1.	`id-aa-ets-revocationValues OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 2 24}`	Usage: the attribute being unsigned that can be used in attributes only once. OID identifying the CRL and OCSP lists of the entire certification path as a minimum according to references defined in Table 8. It can also contain other CRL and OCSP, e. g. which can be used for cross certificate verification or CRL or OCSP.

2.	`RevocationValues ::= SEQUENCE {`
	`  crlVals [0] EXPLICIT SEQUENCE OF`
	`      CertificateList OPTIONAL,`
	`  ocspVals [1] EXPLICIT SEQUENCE OF`
	`      BasicOCSPResponse OPTIONAL,`
	`  otherRevVals [2] EXPLICIT`
	`      OtherRevVals  OPTIONAL }`
	`OtherRevVals ::= SEQUENCE {`
	`  otherRevValType OtherRevValType,`
	`  otherRevVals ANY DEFINED BY`
	`      OtherRevValType`
	`}`
	`OtherRevValType ::= OBJECT IDENTIFIER`
		The CRL and OCSP information lists about the revocation of certificate validity according to references defined in Table 8.

Table 16 SignerLocation

	Record in ASN.1	Short description
1.	`id-aa-ets-signerLocation OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 2 17}`	Usage: the attribute being signed that can be used in attributes only once. OID identifying the additional information about the address of the place (postal ITU-T) where the signer has performed the electronic signing.

2.	`SignerLocation ::= SEQUENCE {`
	`  -- at least one of the following`
	`  -- shall be present`
	` countryName [0] DirectoryString`
	`      OPTIONAL,`
		The address of the place, e. g. of the city where the signer has performed the electronic signing. The text in *DirectoryString* fields is non-empty and *UTF8String* coding must be used.

	`-- As used to name a Country in X.500` `localityName [1] DirectoryString OPTIONAL,` `-- As used to name a locality in X.500` `postalAdddress [2] PostalAddress OPTIONAL` `}` `PostalAddress ::= SEQUENCE SIZE(1..6) OF DirectoryString`	*PostalAddress* format has: 6 lines x 30 letters 1. Street Number 2. Zip Code Locality 3. State …

Table 17 SigPolicyId

	Record in ASN.1	Short description
1.	`id-aa-ets-sigPolicyId OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 2 15 }`	Usage: the attribute being signed that can be used in attributes only once. OID identifying the used signature policy. The signature policy stipulates minimal technical security rules that must be met by a signer and verifier so that the certificate and electronic signature could be considered as valid and in compliance with the content of the field *FieldOfApplication* from the signature policy. It defines e.g. a set of secure algorithms to calculate the hash and minimal key lengths for asymmetric algorithms used in certificates and electronic signatures.
2.	`SignaturePolicyIdentifier ::=CHOICE{` `SignaturePolicyId SignaturePolicyId,` `SignaturePolicyImplied SignaturePolicyImplied` `-- not used in this version }` `SignaturePolicyId ::= SEQUENCE {` `sigPolicyId SigPolicyId,` `sigPolicyHash SigPolicyHash,` `sigPolicyQualifiers SEQUENCE SIZE (1..MAX) OF SigPolicyQualifierInfo OPTIONAL }` `SignaturePolicyImplied ::= NULL` `SigPolicyId ::= OBJECT IDENTIFIER`	Only *SigPolicyId* is used – the OID identifier which unambiguously identifies the signature policy that is used in signing and verification process.
3.	`SigPolicyHash ::=OtherHashAlgAndValue`	It contains the identifier of the hash algorithm and the hash from the signature policy. If the signature policy is defined in ASN.1 (in DER coding), then the hash is calculated from *`signPolicyHashAlg`* and *`signPolicyInfo`*.
4.	`SigPolicyQualifierInfo ::= SEQUENCE {` `sigPolicyQualifierId SigPolicyQualifierId,` `sigQualifier ANY DEFINED BY sigPolicyQualifierId }`	There is defined the address, from which the signature policy can be obtained.
5.	`-- sigpolicyQualifierIds defined` `-- in the present document` `SigPolicyQualifierId ::= OBJECT IDENTIFIER` `id-spq-ets-uri OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 5 1 }` `SPuri ::= IA5String` `id-spq-ets-unotice OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 5 2}` `SPUserNotice ::= SEQUENCE {`	Informative internet address for obtaining the signature policy, e.g. in ASN.1 DER coding is found in *SPuri*. Information about the signature policy, its type and potential usage is displayed in *SPUserNotice*.

	`noticeRef NoticeReference OPTIONAL,`
	`explicitText DisplayText OPTIONAL}`
	`NoticeReference ::= SEQUENCE {`
	`organization DisplayText,`
	`noticeNumbers SEQUENCE OF INTEGER}`
	`DisplayText ::= CHOICE {`
	`visibleString VisibleString (SIZE (1..200)),`
	`bmpString BMPString (SIZE (1..200)),`
	`utf8String UTF8String(SIZE(1..200))}`
		At least OID of the policy, the hash of the policy and the content of *SPUserNotice* are required to be displayed to a verifier in readable form while verifying the signature.

Table 18 ContentTimestamp

	Record in ASN.1	Short description
1.	`id-aa-ets-contentTimestamp OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 2 20}`	Usage: the attribute being signed that can be used in attributes several times with various TSA. OID identifying a time stamp whose purpose is to confirm that a form of data being signed has already existed in time of the time stamp issuance, i.e. before the signature itself. C*ontentTimestamp*, calculated from the data being signed, can be inserted or created in the signing process.
2.	`ContentTimestamp ::= TimeStampToken`	The time stamp stamps the hash that will be inserted in *messageImprint* field within *TimeStampToken.* The hash is calculated from data according to procedure defined in Table 3, line 3 (or according to procedure defined in Annex D).

Table 19 Signer attribute (role)

	Record in ASN.1	Short description
1.	`id-aa-ets-signerAttr OBJECT IDENTIFIER ::= { 1 2 840 113549 1 9 16 2 18}`	Usage: the attribute being signed that can be used in attributes only once. OID identifying the signer's attributes whose purpose is to confirm that the data being signed were signed by a signer in a role being declared or demonstrated through the attribute certificate.

2.	`SignerAttribute ::= SEQUENCE OF CHOICE {`
	`claimedAttributes`
	`[0] ClaimedAttributes,`
	`certifiedAttributes`
	`[1] CertifiedAttributes`
	`}`
	`ClaimedAttributes ::= SEQUENCE OF Attribute`
	`CertifiedAttribute ::=`
	`AttributeCertificate –- as defined`
	`-- in RFC 3281: see clause 4.1.`
		Roles that the signer only declares are listed in the attribute *claimedAttributes*. Certified roles by means of attribute certificates are listed in *CertifiedAttributes*.

Table 20 ESS signing certificate v2

	Record in ASN.1	Short description
1.	`id-aa-signingCertificateV2 OBJECT`	Usage: the attribute being signed that can be

	`IDENTIFIER ::= { 1 2 840 113549 1 9 16 2 47}`	used in attributes only once. OID identifying the attribute with the sequence of certificate identifiers from the signer’s certificate up to the root certificate. It must contain the identifier of the signer’s certificate as a minimum. [17]

2.	`SigningCertificateV2 ::= SEQUENCE {`
	`  certs SEQUENCE OF ESSCertIDv2,`
	`  policies SEQUENCE OF`
	`    PolicyInformation OPTIONAL`
	`}`
		Certs – a sequence of certificate identifiers from the signer’s certificate.
		Policies are not used.

3.	`ESSCertIDv2 ::= SEQUENCE {`
	`  hashAlgorithm AlgorithmIdentifier`
	`    DEFAULT {algorithm id-sha256 },`
	`  certHash     Hash,`
	`  issuerSerial IssuerSerial OPTIONAL`
	`}`
		certHash – a hash of the certificate. If *issuerSerial* is indicated, then *directoryName* must be used in searching.

4.	`Hash ::= OCTET STRING`
	`IssuerSerial ::= SEQUENCE {`
	` issuer      GeneralNames,`
	` serialNumber CertificateSerialNumber`
	`}`
		Sha1 hash is recommended to be used because of compatibility unless another more secure hash algorithm is required to be used.

5.2 Signature in XML format

Rules that are valid for the content of ASN.1 attributes in CMS signature defined in section 5.1 are also valid for the content of elements in XML signature that is specified in the document XAdES ETSI TS 101 903. The following table assigns the XML element from XML signature to the attribute from ASN.1 signature because of unambiguity in defining of QES mandatory attributes.

Table 21 ASN.1 attributes and XML elements of the QES with the same significance of the content

	ASN.1 attribute	XML element
1.	`(id-contentType)`	*(DataObjectFormat)*+ in MIME envelope there is a type MimeType = “message/rfc822” and Encoding = “base64” and if the MIME envelope is not used, then the type defined in MimeType must be in compliance with the document [21].
2.	`(id-messageDigest)`	It does not have the equivalent in XML signature. In XML signature the hash is calculated on the basis of rules defined in the document XAdES and is contained in *DigestValue* element.
3.	`(id-signingTime)`	*(SigningTime)*
4.	`(Id-aa-ets-otherSigCert) or (id-aa-signingCertificate)`	*(SigningCertificate)*
5.	`(id-aa-ets-sigPolicyId)`	*(SignaturePolicyIdentifier)*
6.	`(id-aa-ets-contentTimestamp)*`	*(AllDataObjectsTimeStamp)** or *(IndividualDataObjectsTimeStamp)**
7.	`(id-aa-ets-signerLocation)?`	*(SignatureProductionPlace)?*
8.	`(id-aa-ets-certificateRefs)`	*(CompleteCertificateRefs)*
9.	`(id-aa-ets-revocationRefs)`	*(CompleteRevocationRefs)*
10.	`(id-aa-signatureTimeStampToken)+`	*(SignatureTimeStamp)+*
11.	`((id-aa-ets-escTimeStamp)* (id-aa-ets-certCRLTimestam)* )+`	*((SigAndRefsTimeStamp)* (RefsOnlyTimeStamp)*)+*
12.	`(id-aa-ets-archiveTimestamp)+`	*(ArchiveTimeStamp)+*
13.	`(id-aa-ets-certValues)`	*(CertificatesValues)*
14.	`(id-aa-ets-revocationValues)`	*(RevocationValues)*
15.	`(id-aa-ets-signerAttr)`	*(SignerRole)*

Occurrence of the ASN.1 attribute in CMS signature and XML element in XML signature:

( )	- it must be found only once
( )*	- it does not have to be found but it can be found several times
( )?	- it does not have to be found but it can be found only once
( )+	- it must be found at least once

The following declarations of XML namespace must be used in XML scheme:

```
<?xml version="1.0"?>
<schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://uri.etsi.org/01903/v1.3.2#"
    targetNamespace="http://uri.etsi.org/01903/v1.3.2#"
    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"/>

More detailed information about profiles of XAdES signatures used in administrative communication to exchange the data signed by XML and specification of usage of particular XAdES elements are defined in profiles of the document: XAdES_QES Profile – the Qualified Electronic Signature Format based on XAdES[1].

Note 2: In case of ambiguity between the requirements defined in the referenced document *XAdES_QES Profile* and the requirements defined in the present NSA standard, the requirements of the present standard are obligatory.

5.3 Other signature formats and PAdES

Directive 1999/93/EC of the European Parliament and of the Council [26] and also the NSA Decree No.135/2009 Coll. do not restrict the qualified electronic signature format to CMS or XML signature format but enable the signature implementation in other formats that ensure the requirements which were laid down by Directive and Decree. Examples are PDF AdES or PGP AdES or other signature formats which are used in different types of documents. Presently these formats do not allow storing all necessary data for verification of the qualified certificate validity which must be issued for the public key corresponding to private key being stored in SSCD of the signer. In order to ensure a short-term as well as a long-term verification of the signature formats it is required to define a unified data repository for the short-term and long-term validity verification of the signatures.

Data for verification of the short-term and long-term signature validity consist of at least:

- a time stamp whose hash value when computed includes the signer’s digital signature as a minimum,
- data for validity verification in the form of CRL or OCSP for certificates of the signer, time stamp, CA and certificates for verification of data on certificate validity revocation (indirect CRL or OCSP),
- signing certificates for signature verification of a time stamp, CRL, OCSP and the signature of the document,
- cross certificates and CA certificates,
- used signature policies.

In order to ensure the compatibility and possibility of the long-term verification a unified format for storing such data is defined in Annex E.2. In order to ensure a time stamping of formats which do not allow appending the time stamp for the long-term verification, an integrity signature type is defined in Annex B.2 which in this case must be provided with the time stamp of the signature. All types of signatures can be also archived for the long term by following the procedure described in Annex B.

[1] XAdES_ZEP v1.0 - http://www.ditec.sk/ep/signature_formats/xades_zep/v1.0
XAdES_ZEP v1.0 - http://www.ditec.sk/ep/signature_formats/xades_zep/v1.1
Formát zloženého podpisu v1.0 - http://www.ditec.sk/ep/signature_formats/xades_zep_data_signatures/v1.0
Formát zloženého podpisu v1.1 - http://www.ditec.sk/ep/signature_formats/xades_zep_data_signatures/v1.1

Annex A (normative) Qualified Electronic Signature Formats

Qualified electronic signatures which are used in communication with public bodies of the Slovak Republic and were created since 1st January 2008 must be of CAdES [1] or XAdES [16] type; other types than CAdES and XAdES must not be used. CAdES or XAdES signatures must contain all mandatory attributes required for particular QES formats but may also contain other optional attributes which are not specified by this annex.

With respect to interoperability needs it is recommended to sign a MIME envelope containing electronic documents according to rules from the NSA document [21] based on Article 3(5) of the NSA Decree No.136/2009 Coll. in communication with public bodies of the Slovak Republic.

Note 3: The MIME envelope enables to assign unambiguously a MIME type and coding to the electronic document. It also provides a uniform binary form of the document being signed which enables the compatibility of XAdES and CAdES signature types signing the same hash due to which the unambiguity of multiple signatures is ensured.

Note 4: In direct signing of the electronic document by using XAdES signature without the use of the MIME envelope, it is possible to use different canonical functions and thus to sign different hash values from the same document which can cause the ambiguity in multiple signatures. It can be solved by signing the MIME envelope with XAdES.

Note 5: If the attribute *id-aa-ets-sigPolicyId* is not a part of attributes being signed or OID and hash published in trusted list of the Authority are not used, then the application for QES must ensure that in QES signing and verification were met all requirements of at least one signature policy which was valid and approved by the NSA in time of signature creation (the requirement to ensure the interoperability in EU when the national minimal requirements are met, e.g. for key sizes and algorithm types).

A.1 Qualified Electronic Signature without Time Stamp

Mandatory attributes:

```
id-contentType
id-messageDigest
id-aa-ets-signingCertificateV2 or id-aa-signingCertificate
id-aa-ets-sigPolicyId
```

Optional attributes:

```
id-signingTime
id-aa-ets-contentTimestamp
id-aa-ets-signerLocation
id-aa-ets-certificateRefs
id-aa-ets-revocationRefs
id-aa-signatureTimeStampToken
id-aa-ets-escTimeStamp or id-aa-ets-certCRLTimestamp
id-aa-ets-archiveTimestamp
id-aa-ets-certValues
id-aa-ets-revocationValues
id-aa-ets-signerAttr
```

A.2 Qualified Electronic Signature with Time Stamp

Mandatory attributes:

```
      id-contentType

      id-messageDigest
      id-aa-ets-signingCertificateV2 or id-aa-signingCertificate
      id-aa-ets-sigPolicyId
      id-aa-signatureTimeStampToken
```

Optional attributes:

```
      id-signingTime
      id-aa-ets-contentTimestamp
      id-aa-ets-signerLocation
      id-aa-ets-certificateRefs
      id-aa-ets-revocationRefs
      id-aa-ets-escTimeStamp or id-aa-ets-certCRLTimestamp
      id-aa-ets-archiveTimestamp
      id-aa-ets-certValues
      id-aa-ets-revocationValues
      id-aa-ets-signerAttr
```

A.3 Qualified Electronic Signature with complete validation data

Mandatory attributes:

```
      id-contentType
      id-messageDigest
      id-aa-ets-signingCertificateV2 or id-aa-signingCertificate
      id-aa-ets-sigPolicyId
      id-aa-signatureTimeStampToken
      id-aa-ets-certificateRefs
      id-aa-ets-revocationRefs
      id-aa-ets-certValues   or   in   SignedData.certificates   must   be   all
certificates for verification of a signing certificate up to Trust Anchor (a
root certificate)
```

Optional attributes:

```
      id-signingTime
      id-aa-ets-escTimeStamp or id-aa-ets-certCRLTimestamp
      id-aa-ets-contentTimestamp
      id-aa-ets-signerLocation
      id-aa-ets-archiveTimestamp
      id-aa-ets-revocationValues
      id-aa-ets-signerAttr
```

Note 6: This type of signature is not recommended due to found imperfections when using indirect CRL and OCSP.

A.4 Archive Qualified Electronic Signature

Mandatory attributes:

```
      id-contentType
      id-messageDigest
      id-aa-ets-signingCertificateV2 or id-aa-signingCertificate
      id-aa-ets-sigPolicyId
      id-aa-signatureTimeStampToken
      id-aa-ets-certValues
      id-aa-ets-revocationValues
      id-aa-ets-archiveTimestamp
```

Optional attributes:

```
      id-signingTime
      id-aa-ets-certificateRefs
      id-aa-ets-revocationRefs
      id-aa-ets-escTimeStamp or id-aa-ets-certCRLTimestamp
      id-aa-ets-contentTimestamp
      id-aa-ets-signerLocation
      id-aa-ets-signerAttr
```

Note 7: The archive signature is especially used to protect the signer's signature from the long-term perspective so that the signature is possible to be verified and considered as valid even in cases of CA compromising or inaccessibility to CRL or OCSP data and also if algorithms used for signature creation are becoming less secure. In these cases, it is possible to verify the signature and consider as valid because the archive signature will ensure a longer trusted protection of the content of all information necessary for signature verification by using a time stamp signature.

Note 8: Archiving of the electronic signature by using the archive qualified electronic signature requires individual time stamping of each signature with stronger algorithms (or longer key lengths) in sufficiently long time intervals what is together with all certificates and data necessary for certificate validity verification not effective with respect to greater number of signatures and therefore systems archiving a great number of electronic signatures are recommended to use the integrity signature pursuant to procedure in Annex B.

A.5 Multiple signature

If a multiple signature is used, then the signature validity of each signer is verified separately in the electronic signature verification. The signature invalidity of one signer does not cause the signature invalidity of another one which was valid earlier and signed the same document. Embedded signatures (countersignature) are not recommended to be used for QES.

The multiple signature XAdES or XAdES with CAdES simultaneously is not possible to be realized by means of one ETSI format, and therefore there is defined a storage of ETSI formats into "ZEPf (ZIP) format" which enables to realize the multiple signature of the binary same document being stored in the MIME envelope according to [21].

Annex B (normative) The storage procedure and the signature format for the long-term storage of electronic documents signed by QES

The integrity signature type defined in section B.2 is used for the long-term storage of electronic documents and electronic documents signed by QES or documents signed by other types of signature. The integrity signature type is used in the way defined in section B.1 unless there is used another procedure ensuring the integrity of archived data which enables the verification to trusted time to which used key, algorithms and archived data were verified as valid, not damaged and any change made after that time is detectable.

B.1 Procedure in creation and control of integrity signature of files

Document files and signature files which are necessary to be secured by the integrity signature, e.g. when archiving, are used to create the sequence of references to documents secured by the integrity signature, pursuant to B.2, thus by signing the sequence of references the first integrity signature is created.
Prior to certificate validity expiration of the first integrity signature or the time stamp certificate of the digital signature, if the time stamp was appended to the integrity signature, a new sequence of all files from the previous integrity signature is calculated by using the currently secure hash algorithm. The previous integrity signature and also its time stamp are supplemented by all certificates required for their verification together with data for validity verification (CRL, OCSP) of these certificates up to the trusted root certificate. At the end of the new sequence the record of the file with the previous integrity signature is appended. The new sequence is signed by a new key to which the certificate with the longer validity than the validity of the previous integrity signature is issued.

In verification [25], the procedure is in reverse order. Firstly the most recent integrity signature is verified, then the hash values from textual document are checked and thus it is proceeding up to the first integrity signature. Verification of each signature is performed to the time of the signature time stamp. If the signature does not contain the time stamp, then the verification is performed to the time of the following integrity signature which may contain CRL or OCSP for verification of the previous integrity signature. In verification it is also possible to verify hashes of particular files defined in the signed textual document where the incorrect hash does not cause the error in integrity signature verification but it will result in information about the change of the integrity archived file since the last integrity archiving.

Verification output of the long-term storage of electronic documents will be:

- a verification status,
- an electronic file being verified,
- a sequence of integrity signatures from the first up to current integrity signature enabling the reverse integrity and validity verification of the electronic document by any verifier while the content of other integrity archived electronic documents except for their name, hash and note cannot be obtained or revealed.

B.2 Integrity electronic signature

Integrity electronic signature is the internal CMS signature pursuant to A.1 with the optional time stamp which signs a textual document coded in UTF8 whose internal structure consists of line sequence the format of which is defined in B.3 while the line with NOTICE in not mandatory and the line FILE with the following line HASH with the hash value are mandatory. The integrity electronic signature is stored in the file with the extension ".P7M". The textual document is stored in the field *encapContentInfo* within *SignedData* (Table 2, line 4).

B.3 Format of textual document

CRLF	character (13) + character (10)

FILE=<[URL]|[file name]>CRLF
HASH (<algorithm>**:**<algorithm OID>**)**=<file hash – capital letters>CRLF
NOTICE=<note, for example name of archive CD/DVD disc >CRLF

B.4 Example of sequence of file attributes

```
FILE=rewards.eml
HASH(SHA1:1 3 14 3 2 26) = F15BF971C276AC7173065A741DF998C00DF44F31
NOTICE=File being signed.
FILE=http://www.example.sk/other/rules.doc
HASH(SHA1:1 3 14 3 2 26) = 8A01657D752FEFF5A7F3E7910C2C5D45D708D7EB
NOTICE=Original file.
```

The integrity signature can be also used for archiving of files, for example files which are published on the website or in archive CD/DVD or file servers. The field FILE can contain only the file name or URL for the file or the complete path with the name of the file in the file system on which the integrity archiving has been realized. The note appended to the file name can contain for example the name of CD/DVD media or the physical placement in the storage – archive or other necessary notes appended to integrity archived file. The integrity signature is not intended for the long-term archiving but for ensuring the integrity of documents during the usage period of hash and signing algorithms which are used in TXT document being signed and in the signature. Verification of the integrity signature is assumed within the maximal time when hash algorithms used in TXT document are considered to be secure and within the maximal time when the trusted validity status of used certificates in the integrity signature can be obtained.

Annex C (normative) Certification path storage in all QES formats

Following attributes identify the certification path unambiguously in qualified electronic signatures. They are inserted by a signer and the verifier must use them:

CAdES: id-aa-ets-signingCertificateV2 or id-aa-signingCertificate
XAdES: SigningCertificate

These attributes will contain references to certificates according to built certification path. The certification path is built and verified in compliance with rules defined in the NSA document [23] mainly specifying the RFC 5280 profile from ITU-T X.509 where rules for qualified certificates are not defined.

Annex D (normative) Time Stamp calculation

The following table describes attributes being included in the calculation of the hash value which is stored in the field *messageImprint* within *TimeStampToken.*

Table 22 Identification of signature attributes from which the hash is calculated

Type of Timestamp	Identification
id-aa-ets-contentTimestamp	D
id-aa-signatureTimeStampToken	S
id-aa-ets-archiveTimestamp	A

ASN.1 data elements which are included as the part of the superior object in the hash value calculation are indicated by means of '"'.

Table 23 CAdES signature in ASN.1

ASN.1
`ContentInfo ::= SEQUENCE {`
`contentType ContentType, -- id-signedData`
`content [0] EXPLICIT ANY DEFINED BY contentType }`

Table 24 Complete SignedData in ASN.1

	ASN.1	**Tag**	**Len**	**Value**
1.	`SignedData ::= SEQUENCE {`			
2.	`version CMSVersion,`			
3.	`digestAlgorithms DigestAlgorithmIdentifiers,`			
4.	`encapContentInfo SEQUENCE {`	A	A	A
5.	`eContentType ContentType,`	"	"	"
6.	`eContent [0] EXPLICIT`	"	"	"
7.	`OCTET STRING OPTIONAL` `-- not present if signature is detached` `},`	"	"	", D
8.	`-- External Data (if signature detached)*`			A, D
9.	`certificates [0] IMPLICIT CertificateSet OPTIONAL,`	A	A	A
10.	`crls [1] IMPLICIT CertificateRevocationLists`	A	A	A

	`OPTIONAL,`			
11.	`signerInfos SET OF`			
12.	`SEQUENCE {  -- SignerInfo`			
13.	`version      CMSVersion,`	A	A	A
14.	`sid    SignerIdentifier,`	A	A	A
15.	`digestAlgorithm    DigestAlgorithmIdentifier,`	A	A	A
16.	`signedAttrs [0] IMPLICIT SET SIZE (1..MAX) OF`	A	A	A
17.	`SEQUENCE { -- Attribute`	"	"	"
18.	`attrType OBJECT IDENTIFIER,`	"	"	"

19.	`attrValues SET OF AttributeValue`
	`} OPTIONAL,`
		"	"	"

20.	`signatureAlgorithm`
	`SignatureAlgorithmIdentifier,`
		A	A	A

21.	`signature OCTET STRING, -- SignatureValue`	A	A	A, S
22.	`unsignedAttrs [1] IMPLICIT SET SIZE (1..MAX) OF`			
23.	`SEQUENCE {`	A	A	A
24.	`attrType OBJECT IDENTIFIER,`	"	"	"

25.	`attrValues SET OF AttributeValue`
	`} OPTIONAL`
	`}`
	`}`
		"	"	"

* *External data* means data protected by external signature where these data are not included in *eContent* field of CAdES signature. A hash value calculation for *contentTimestamp* and *archiveTimestamp* include the hash calculation through external data. The algorithm used in the external signature may become weak in the future (due to its compromising) and therefore the integrity of external signed data is required to be protected by appending a new *archiveTimestamp* or the integrity signature.

Annex E (informative) Formats of binding the electronic documents with electronic signature or QES

E.1 S/MIME file format

It is necessary to limit particular MIME types and codings to the basic set for documents being signed according to [21] and for the signature S/MIME itself because of unambiguous processing in the verification and displaying of qualified electronic signatures stored in S/MIME containing OID *id-data* (1 2 840 113549 1 7 1) in *ContentType* which does not specify the type of signed data exactly, as follows:

1. Text in a character set US ASCII and 7 bit coding

   If the type and coding are not defined in S/MIME, then this type and coding are supposed to be used. They are subsets of UTF-8 and this is the advantage.

   Content-Type: text/plain; charset=us-ascii
   Content-Transfer-Encoding: 7bit

2. Text in a character set UTF-8 and 8 bit coding

   Content-Type: text/plain; charset=UTF-8
   Content-Transfer-Encoding: 8bit

3. Document in permitted format [21] and in Base64 coding

If it is necessary to sign documents that do not meet requirements for Content-Type: text/plain, then these documents must only be in formats permitted by NSA decrees issued to the Act No. 215/2002 Coll. on Electronic Signature and on amendment and supplementing of certain acts as amended and according to [21]. Documents must be coded in Base64 (`Content-Transfer-Encoding`). The application that sings or verifies the qualified electronic signature must know how to display the document on the basis of `Content-Type;` otherwise it declares that it is not possible to verify the signature because the document cannot be displayed to a signer or verifier.

4. Types dividing MIME to particular parts

   Content-Type: multipart/signed

   Content-Type: multipart/mixed

If QES is stored in internal signature CMS (*.p7m) in S/MIME, then the textual document is supposed to be in UTF-8 or with MIME head according to [21].

Encrypted message format (*.p7m) in S/MIME is not described in details in the present document because the qualified certificate must not be used for encryption but only for verification of qualified electronic signature validity and authenticity and for verification of the signer’s identity.

E.1.1 Example of signed data CMS (*.p7s) encoded in S/MIME format

A signed message in S/MIME format is supplemented with an optional head (first 6 lines in the example) that enables the signature verification in commonly accessible users’ e-mails too.
The message is stored in the file whose extension is ***.eml** (e-mail). Signed messages are recommended to be sent as attachments of another e-mail because if the message was modified in the process of e-mail sending, transmitting or processing, then the signature would be damaged irrecoverably and its verification would not be possible. As examples could be mentioned S/MIME incompatible programmes which change the message format, different antiviral reports about message control or advertisement information being supplemented during the message transmission which may completely damage the signed data.

```
Message-ID: <27x5x2004at09x30x57x3B90>
From: <peter.rybar@infolooks.sk>
To: <info@nbusr.sk>
Subject: signature in S/MIME file (*.EML)
Date: Thu, 27 May 2004 09:30:57 +0200
MIME-Version: 1.0
Content-Type: multipart/signed;
      protocol="application/x-pkcs7-signature";
      micalg=SHA1;
      boundary="--=_NextPart27x5x2004at09x30x57x3B90"

This is a multi-part message in MIME format.

----=_NextPart27x5x2004at09x30x57x3B90
Content-Type: multipart/mixed;
      boundary="--=_NextPart2X8X2005at11X20X05-E97E"

----=_NextPart2X8X2005at11X20X05-E97E
Content-Type: text/plain
Content-Transfer-Encoding: 7bit

This is signed message.
```

```
In SMIME format.

----=_NextPart2X8X2005at11X20X05-E97E
Content-Type: text/plain; charset=UTF-8;
Content-Transfer-Encoding: base64

77u/dGVzdCBzIENQDQrEvsWhxI3FpcWlxb7FpcO9w73DocOtw63DqcOpw6nDqT3DusOkxYjDtMKn
w7TCp8Knw7TDtC4tLQ0KLS4tLQ0KwqfDusKnw7rFiMO6cMWIw7pww6TDusOkPcK0w6nDrW8NCnl1
...
xL4rxL7EjSvEvg0KxL7EvsWhxaHEjcSNxaXFpcW+xb7DvcO9w6HDocOtw63DqcOpPT3CtMK0w7rD
usOkw6TDtMO0wqfCp8WIxYgNCktPTklFQw0K

----=_NextPart2X8X2005at11X20X05-E97E--

----=_NextPart27x5x2004at09x30x57x3B90
Content-Type: application/x-pkcs7-signature;
      name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
      filename="smime.p7s"

MIIGQQYJKoZIhvcNAQcCoIIGMjCCBi4CAQExCzAJBgUrDgMCGgUAMAsGCSqGSIb3DQEHAaCCA9gw
...
F9Q5jHSMaByoXE2wkZ365srlimkKXO97p+NiEpZwzI9tIi37Z4IrRbppjz3BjbMqHYdpkP06jWzm
KYbhjscDhpoL

----=_NextPart27x5x2004at09x30x57x3B90—
```

E.2 ZIP file format (ZEPf)

It is necessary to bind a document being signed, its signatures and information necessary for document signature verification and other documents relating to the document being signed into a format whose processing is simple and commonly accessible. ZIP belongs to such commonly accessible formats. It is necessary to determine rules about naming of particular fields in ZIP file for simpler processing and ensuring the compatibility of applications which use or want to use the format mentioned above. One of the important information in the signature use is the time when the activity was performed. This fact can be used in naming of particular fields. Names of particular fields in directory structure will contain the insertion time to ZIP file, in GeneralizedTime “YYYYMMDDhhmmssZ“ format. First letters of fields in directory structure indicate the field type. The serial number which distinguishes files with the same type and time within the ZIP directory can follow the letter “Z”. Files in ZIP can be stored in one main directory “DYYYYMMDDhhmmssZ” which always contains the signed document and its signatures while additional data may be stored into sub-directories: *Certificate* (a list of certificates), *Revocation* (a list of CRL and OCSP not only for verification of used signatures in ZIP file but also for verification of other signatures which are archived with the integrity signature placed in ZIP file), *Policy* (a list of used policies), *Other* (non-specified list of attached files).

Table 25 Names of basic file set in ZIP format

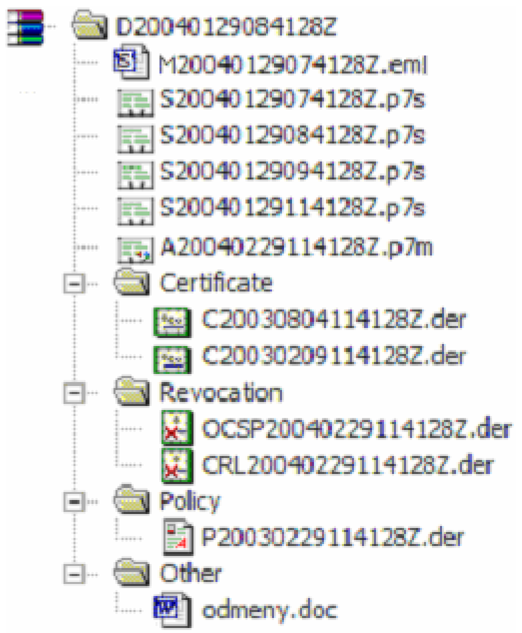
	Field type	Description	Example
1.	D	Directory – the main signature directory. It is not mandatory but it simplifies the usage when the ZIP content e.g. the signature and signed document is copied outside the ZEPf (ZIP) file.	D20040129084128Z
2.	S	Sign – external signature: simple, with a time stamp, with	S20040129084128Z.p7s

		complete verification data, archive or a multiple signature.	S20040129084128Z.xml
3.	M	MIME envelope of signed files which are stored in the file “.EML” according to [21].	M20040129084128Z.eml
4.	A	Integrity – integrity signature of files which signs files with *.p7s, *.p7m and *.XML extensions, all certificates, CRL or OCSP, signature policy and also other files. Signatures that are integrity archived for the first time must be extended to X-long form or QES archive form. The integrity signature is not intended for the long-term archiving but only for ensuring the integrity of documents during the usage period of hash algorithms that are used in archival TXT document. It is defined in section B.2.	A20040229114128Z.p7m A20040229114128Z.xml
5.	P	Policy – Signature Policy	P20030229114128Z.der
6.	C	Certificate	C20030209114128Z.der
7.	CRL	Certificate Revocation List	CRL20040229114128Z.der
8.	OCSP	Online Certificate Status	OCSP20040229114128Z.der

The table mentioned above specifies a minimal set of file types which must be processed by each QES application for administrative communication. Electronic documents being signed for administrative communication are recommended to be stored in MIME coding in the file with (*.EML) extension.

E.2.1 Example of fields in ZIP format (ZEPf) and names of standard directories

The file M20040129074128Z.eml may contain only documents which are in one MIME envelope bound by MIME coding which is profiled in the document [21] being issued by the NSA on the basis of Article 3 (5) of the NSA Decree No.136/2009 Coll. The file M20040129074128Z.eml may also represent SMIME format for binding of the signed document and the signature itself which can be whenever divided into the signed file itself with the extension “.EML” and the file with the signature which contains CMS signature with the file extension “.P7S”. The file with the extension “.P7M” signifies the internal CMS signature which may contain the signed textual document in the format defined in Annex B.2.



```
D20040129084128Z
    M20040129074128Z.eml
    S20040129074128Z.p7s
    S20040129084128Z.p7s
    S20040129094128Z.p7s
    S20040129114128Z.p7s
    A20040229114128Z.p7m
    Certificate
        C20030804114128Z.der
        C20030209114128Z.der
    Revocation
        OCSP20040229114128Z.der
        CRL20040229114128Z.der
    Policy
        P20030229114128Z.der
    Other
        odmeny.doc
```

E.3 DITEC XML file format

DITEC XML format for binding of mainly XML documents and XAdES signatures used in administrative communication for exchange of XML signed data among systems of state administration defines the profiles specified in the document:

XAdES_QES Profile – the qualified electronic signature profile based on XAdES.[2]. – annex The format of compound electronic signature

Note 8: In case of ambiguity between the requirements defined in the referenced document and the requirements defined in the present NSA standard, the requirements of the present standard are obligatory.

[2] XAdES_ZEP v1.0 - http://www.ditec.sk/ep/signature_formats/xades_zep/v1.0
XAdES_ZEP v1.0 - http://www.ditec.sk/ep/signature_formats/xades_zep/v1.1
Formát zloženého podpisu v1.0 - http://www.ditec.sk/ep/signature_formats/xades_zep_data_signatures/v1.0
Formát zloženého podpisu v1.1 - http://www.ditec.sk/ep/signature_formats/xades_zep_data_signatures/v1.1

Annex F (informative) Revisions made since previous version

F.1 Additional requirements

The following items have been added which significantly affect the requirements:

Sections 5.2 and E.3 specify QES profile of XML AdES signature.
Section 5.3 specifies procedures for the long-term verification of electronic signatures by using the integrity signature and ZEPf format, if the signature format does not allow the storage of all necessary data for the long-term verification.
Annex A was modified according to new decrees of the Act No.215/2002 Coll. which were issued in 2009.
Annex A.4 does not contain *id-aa-ets-certificateRefs or id-aa-ets-revocationRefs* among mandatory fields any more in order to archive directly also QES with the signature time stamp.
Annex B was attached. It contains definitions of procedures and signature formats to ensure requirements of the amendatory act on ES which defines a new accreditation service “the long-term storage of electronic documents signed by qualified electronic signature”.
Section E.2 defines the minimal file types for ZEPf format.

F.2 Updated requirements

The following items have been updated to extend choices or otherwise modify requirements:

The content of the document has been changed due to the NSA Decree No.135/2009 Coll. and other decrees to the amendatory act on ES.
Annex C only contains fields which are protected by the signer’s signature. Not protected fields were removed because their content cannot be mandatory if it is not protected from the modification.
In section E.2 a closing signature “F” was removed. This type of signature was also removed from the decree defining two methods of creating the multiply signature.
Section E.2 contains the definition of directory names and mandatory file names as well as a new name for MIME envelope containing files being signed.

F.3 Clarifications

The following items have been updated to clarify existing requirements:

The amendatory act and decrees to the act on ES have specified and defined new rules whose fulfillment was not defined in the previous versions of QES formats and therefore the previous documents defining QES formats were revoked as well as the previous decrees.

F.4 Editorial

A number of other editorial changes were made which do not affect the technical content of the present document:

The document was reformatted and divided into annexes. The archival signature was renamed the integrity signature in order not to be substituted for QES archive format.

Annex G (informative) Bibliography

Basic documents of the Slovak Republic legislation for electronic signature
http://www.nbusr.sk/en/electronic-signature/legislation/index.html
Qualified electronic signature formats
http://www.nbusr.sk/en/electronic-signature/approved-formats/index.html
Certification path creation and certificate validity verification
http://www.nbusr.sk/en/electronic-signature/verification/index.html

- IETF RFC 4158 "Internet X.509 Public Key Infrastructure: Certification Path Building"

NOTE: Available at http://www.rfc-archive.org/getrfc.php?rfc=4158

- IETF RFC 5217 "Multi-Domain PKI Interoperability" July 2008

NOTE: Available at http://www.rfc-archive.org/getrfc.php?rfc=5217

- IETF RFC 4853 (2007): "Cryptographic Message Syntax (CMS) Multiple Signer Clarification"
- IETF RFC 3447 (2003): "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1"
- ISO/IEC 8825-1:1998, Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
- ISO/IEC 19794-2:2005, Biometrics — Biometric Data Interchange Formats — Part 2: Finger Minutiae
- IETF RFC 3279 (2002): "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile"
- IETF RFC 4055 (2005): "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure - Certificate and Certificate Revocation List (CRL) Profile"
- IETF RFC 3281 (2002): "An Internet Attribute Certificate profile for Authorization"
- IETF RFC 3370 (2002): "Cryptographic Message Syntax (CMS) Algorithms"
- ETSI TR 102 038: "TC Security - Electronic Signatures and Infrastructures (ESI); XML format for signature policies"
- ETSI TS 101 861: "Time stamping profile"
- EN 14890-2:2008: "Application Interface for smart cards used as Secure Signature Creation Devices - Part 2: Additional Services"
- ETSI TS 101 456: "Electronic Signatures and Infrastructures (ESI); Policy requirements for certification authorities issuing qualified certificates"
- ETSI TS 102 042: "Electronic Signatures and Infrastructures (ESI); Policy requirements for certification authorities issuing public key certificates"
- CWA 14167-1: "Security Requirements for Trustworthy Systems Managing Certificates for Electronic Signatures - Part 1: System Security Requirements"
- CWA 14167-2: "Security Requirements for Trustworthy Systems Managing Certificates for Electronic Signatures - Part 2: Cryptographic module for CSP Signing Operations with Backup - Protection Profile"
- CWA 14167-3: "Security Requirements for Trustworthy Systems Managing Certificates for Electronic Signatures - Part 3: Cryptographic module for CSP key generation services - Protection profile (CMCKG-PP)"
- CWA 14167-4: "Security Requirements for Trustworthy Systems Managing Certificates for Electronic Signatures - Part 4: Cryptographic module for CSP signing operations - Protection profile - CMCSO PP"

- W3C Recommendation (10 June 2008): "XML Signature Syntax and Processing (Second Edition)"

NOTE: Available at http://www.w3.org/TR/xmldsig-core/

- W3C Recommendation (10 December 2002): "XML Encryption Syntax and Processing"

NOTE: Available at http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/

- CWA 14169: "Secure Signature-Creation Devices "EAL 4+"

- IETF RFC 4949 "Internet Security Glossary, Version 2" August 2007

NOTE: Available at http://www.rfc-archive.org/getrfc.php?rfc=4949

- NIST X.509 path validation test suite

NOTE: Available at http://csrc.nist.gov/pki/testing/x509paths.html http://csrc.nist.gov/pki/testing/pathdiscovery.html

- Object Identifier (OID) Repository: ITU-T X.660 & X.670 Recommendation series (or ISO/IEC 9834 series of International Standards)

NOTE: Available at http://www.oid-info.com/

- FESA – Forum of European Supervisory Authorities,

NOTE: Available at http://www.fesa.rtr.at

- OID tree structure,

NOTE: Available at http://www.darmstadt.gmd.de/secude/Doc/htm/oidgraph.htm

- Common ISIS-MTT Specification for interoperable PKI applications. Version 1.1. 16 March 2004

- Internet Draft "X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA"

NOTE: Available at http://tools.ietf.org/html/draft-ietf-pkix-sha2-dsa-ecdsa-05

- Internet Draft "X.509 Public Key Infrastructure Time-Stamp Protocol (TSP) "

NOTE: Available at http://tools.ietf.org/html/draft-ietf-pkix-rfc3161bis-01

- TeleTrusT Deutschland e. V., "OID-Liste",

NOTE: Available at http://www.teletrust.de/index.php?id=171

- European Commission http://ec.europa.eu/

- IDABC stands for Interoperable Delivery of European eGovernment Services to public Administrations, Businesses and Citizens. - eSignature Agenda & Presentations

NOTE: Available at http://ec.europa.eu/idabc/en/document/7312

- European Network and Information Security Agency (ENISA)

NOTE: Available at http://www.enisa.europa.eu/

- PKIX Status Pages http://tools.ietf.org/wg/pkix/

Annex H History

Version	Date of issuing	Note	Editor
Version 1.0.	30 September 2004	First issuing (revoked)	Peter Rybár, NSA
Version 1.1	14 August 2005	Second issuing	Peter Rybár, NSA
Version 1.2	6 November 2005	Adding of multipart MIME	Peter Rybár, NSA
Version 2.0 No.3198/2007/IBEP-009	22 July 2007	Specifying the identification of XAdES signature in ZEPf(ZIP)	Peter Rybár, NSA
Version 3.0 No. 2425/2010/IBEP/OEP-001	12 August 2009	Standards of QES formats preceding the version 3.0 are revoked. A change pursuant to amendatory decree from 2009.	Peter Rybár, NSA