ONVIF^{TM}
Export File Format Specification

Version 1.0
August, 2013

CONTENTS

1 Scope

This document defines the ONVIF file format for exported media. The specification defines the mechanism necessary to support interoperable verification of the authenticity by the receiving party.

2 Normative references

ONVIF™ Core Specification, Version 2.2, May, 2012
<http://www.onvif.org/specs/core/ONVIF-Core-Specification-v220.pdf>

ISO/IEC 23000-10 Information technology – Multimedia application format – Part 10: Surveillance application format

NIST FIPS 180-4 Secure Hash Standard

ISO/IEC 14888-2 Information technology – Security techniques – Digital signatures with appendix – Part 2: Integer factorization based mechanisms

PKCS#1, v2.1 RSA Cryptographic Standard

NIST FIPS 186-3 Digital Signature Standard (DSS)

IETF RFC 344 Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1

ITU-T Recommendation X.690 (2008) | ISO/IEC 8825-1:2008, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

3 Terms and Definitions

3.1 Definitions

Certificate	A certificate as used in this specification binds a public key to a subject entity. The certificate is digitally signed by the certificate issuer to allow for verifying its authenticity
Signature	A digital signature or digital signature scheme is a mathematical scheme for demonstrating the authenticity of a digital message or document.

3.2 Abbreviations

ONVIF	Open Network Video Interface Forum
SHA	Secure Hashing Algorithm

4 Overview

All data a user wishes to carry away separately are put into a metaphorical bag. The bag is than sealed to enable tamper detection. Anyone wanting to use the data from the bag first examines the seal. The data in the bag are identical with the original data as long as the seal is intact. Here, the metaphorical bag is represented by a file and the seal is represented by a signature over all data in the file.

The “bag of evidence” approach builds on procedures for media data and related meta data to be securely extracted from a trusted storage in a separate file. It is defined which meta data have to be preserved in order to provide for accurate replay. Data are provided “as is” without any further assertions, whatsoever, to perpetuate evidence.

Processing power reliefs include usage of hash functions before signature algorithms are applied. Multiple stages of signatures might be applied to collect additional information into a single sealed file.

International state of the art standards are applied for the file structure, hash and signature algorithms. The surveillance application format and the RSA2048 signature defined by ISO/IEC as well as the SHA-256 hash algorithm approved by NIST come into operation for most widespread interoperability.

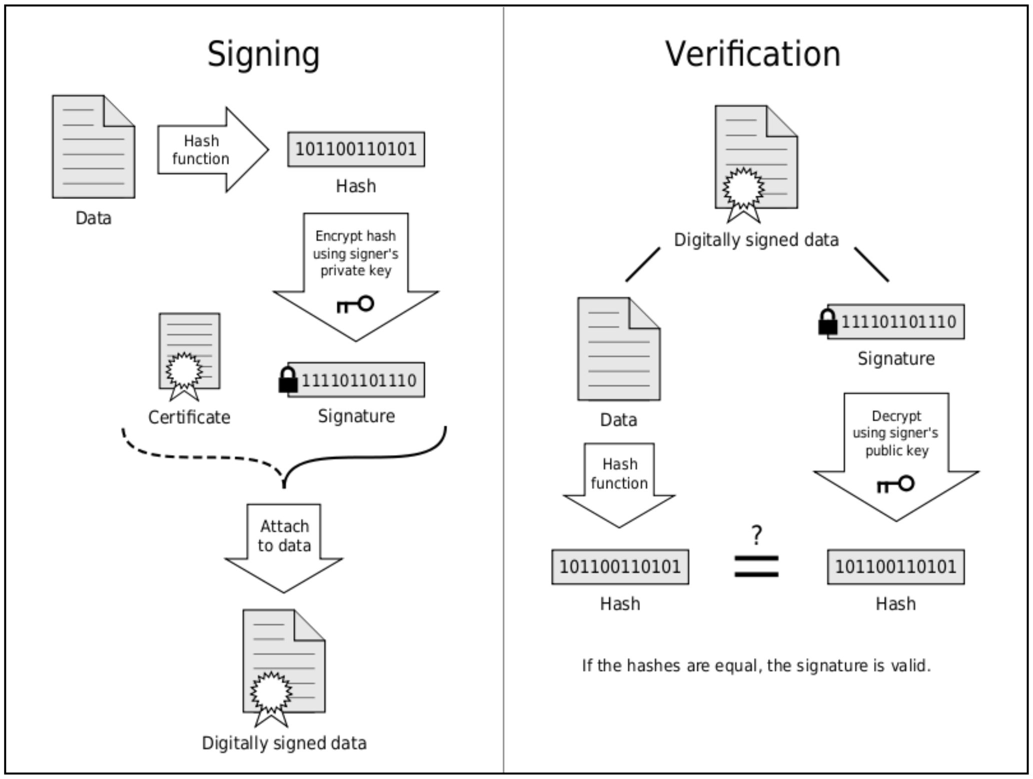


Figure 1 sealing and examination process in a nutshell (Source: Wikipedia)

4.1 Use case 1: Playback of chunked and oversize clips at remote site

An operator exports a Video clip with associated Audio from a DVR of brand A onto two DVDs, because it didn’t fit on one. The selected recording period contains gaps because the recorder did only record when motion is detected. The DVD is then sent to a second site with Software where the content of the DVDs is copied to the local hard disk. The user then plays it back in the

Video Management System of brand B. The operator at the playback station wants to see the gaps in the recording, seek to any time where Video has been exported. On playback he expects the Video to playback smoothly with lip sync Audio.

4.2 Use case 2: Forensic analysis at court

A court receives video clips from a grocery store, a street surveillance system and a metro operator. All three videos are shown in the courts approved video player.

The judges want to see the suspect in all three video clips with exact time information. They also want to have information when the video clips have been exported and whether the video sequence is complete and authentic.

4.3 Use case 3: Playback at players not equipped according to the present specification

An authorized person receives video clips in the format defined in the present specification and wants to play back the media data on players conforming to the underlying standards definitions. Interpretation of the additional information added by the present specification is not required.

5 Export Format

5.1 Required Side Information

The SurveillanceExportBox is required. It is recommended that the SurveillanceExportBox be placed as early as possible in files, for maximum utility.

In order to be able to associate the recording with a camera/microphone and the exporting system the following information shall be placed in the box:

- Source – Description of the video source
    - Name – Name of the camera
    - URL – Address under which the camera can be accessed
    - MAC – Unique physical address of the camera (examples: 08-00-27-00-0C-15, 08:00:27:00:0C:15, 080027000C15)
    - Line – Input line number token for multi channel devices
- Source – Description of the audio source
    - Name – Name of the microphone
    - URL – Address under which the microphone can be accessed
    - MAC – Unique physical address of the microphone
- Export – Unit executing the export
    - Name – Name of the exporting unit
    - URL – Address under which the exporting unit can be accessed
    - MAC – Unique physical address of the exporting unit
    - Time – date and time information on when the export was executed (start time)
    - Operator – Name or identification of the operator performing the export

SurveillanceExportBox

Box Type:	'suep'
Container:	Meta Box ('meta'), file level
Mandatory:	Yes
Quantity:	Exactly one

This box shall provide ordered information about source and export units. Media tracks are identified by their respective UUID. Information has to be available only for the type of media present but separately for each and every track. Information about the exporting unit is leading.

Syntax

```
class SurveillanceExportBox

      extends     FullBox('suep', version = 0, 0){
      string      ExportUnitName;
      string      ExportUnitURL;
      string      ExportUnitMAC;
      UInt(64)    ExportUnitTime;
      string      ExportOperator

      Uint(8)  video_count;
      int i;
      for (i=0; i < video_count; i++) {

            UInt(128)   VideoTrackUUID
            string      VideoSourceName;
            string      VideoSourceURL;
            string      VideoSourceMAC;
            string      VideoSourceLine;
      }

      Uint(8)  audio_count;
      int j;
      for (j=0; j < audio_count; j++) {

            UInt(128)   AudioTrackUUID
            string      AudioSourceName;
            string      AudioSourceURL;
            string      AudioSourceMAC;
      }
}
```

Semantics

String items are null-terminated strings in UTF-8 characters. If not applicable, the string shall contain the null-termination only.

`ExportOperator` is a string that gives Name or identification of the operator performing the export. This string may be empty

`ExportUnitTime` is an integer that gives date and time designation as defined in ISO/IEC 14496-12 of when the export operation has been started

`video_count` is an integer that gives the number of entries in the following list and equals the total number of video tracks

`audio_count` is an integer that gives the number of entries in the following list and equals the total number of audio tracks

5.2 Timing

The `startTime` element of `AFIdentificationBox`[1] shall contain the UTC based time of the first media sample in the fragment.

[1] Box definitions can be found in ISO/IEC 23000-10 Information technology – Multimedia application format – Part 10: Surveillance application format.

5.3 Signature

5.3.1 Preparing the signature input

The input to the signature algorithm are all boxes of the file including boxes for the signature to be created, with the corresponding signature string set to zero. Particularly, the input contains signatures that are already present for repeated signing operations.

5.3.2 Generating the signature

Implementations of this specification shall support RSASSA-PSS signatures as specified in ISO/IEC 14888-2 and PKCS#1 v2.1 with

- SHA-256 as specified in FIPS 180-4 as cryptographic hash function
- an RSA modulus length of 2048 bits
- MGF1 as specified in PKCS#1 v2.1 as mask generation algorithm with SHA-256 as cryptographic hash function
- Salt length 20
- Trailer field number as specified by the trailerFieldBC constant

Implementations may support other digital signature algorithms, if appropriate.

The generated signature string has to be included in the SignatureBox as defined in 5.3.3.

Generating and maintaining parameters of the signature algorithm, particularly signature and verification keys, is outside the scope of this document. Recommendations given, e.g., in FIPS 186-3 should be followed where appropriate.

5.3.3 Include the generated signature in the file

There are no changes to the file itself or the content after the signing operation has been performed. The sole exception is the input of the signature at the appropriate place.

The following box definitions provide for signature identification and inclusion. Encryption is not required; therefore an OriginalFormatBox is not necessary.

Item Protection Box[2]
Box Type: ‘ipro’
Container: Meta box (‘meta’)
Mandatory: Yes
Quantity: Exactly one
The `protection_count` shall be 1.

Protection Scheme Info Box[2]
Box Types: ‘sinf’
Container: Item Protection Box (‘ipro’)
Mandatory: Yes
Quantity: One per signing instance
Contains exactly one SchemeTypeBox and exactly one SchemeInformationBox.

Scheme Type Box[2]

[2] Box definitions can be found in ISO/IEC 14496-12 Information technology -- Coding of audio-visual objects -- Part 12: ISO base media file format.

Box Types:	‘schm’
Container:	Protection Scheme Information Box (‘sinf’)
Mandatory:	Yes
Quantity:	One per signing instance

The `scheme_type` shall be 0x6F656666 („**O**nvif **E**xport **F**ile **F**ormat“).
The `scheme_version` shall be 0x00010000 (version 1).

Scheme Information Box[2]

Box Types:	‘schi’
Container:	Protection Scheme Information Box (‘sinf’)
Mandatory:	Yes
Quantity:	One per signing instance

Contains exactly one SignatureBox and exactly one CertificateBox. May also contain exactly one AdditionalUserInformationBox and exactly one SignatureConfigurationBox.

SignatureBox

Box Types:	‘sibo’
Container:	Scheme Information Box (‘schi’)
Mandatory:	Yes
Quantity:	One per signing instance

Syntax
```
aligned(8) class SignatureBox
      extends Box(‘sibo’) {
      bit(8)  signature[];
}
```

Semantics

`signature` binary byte array. Length depends on used RSA key length.

CertificateBox

Box Type:	‘cert’
Container:	Scheme Information Box (‘schi’)
Mandatory:	Yes
Quantity:	One per signing instance

Syntax
```
aligned(8) class CertificateBox
      extends Box(‘cert’) {
      bit(8) data[];
}
```

Semantics

data is the DER encoded binary byte array representation of the certificate for the key that should be used to verify the signature in the SignatureBox

5.4 Repeated signing

To add e.g. electronic receiving stamps repeated signing of the file might be applied.

Repeat steps defined in 5.3.1 and 5.3.2 and append another ProtectionSchemeInfoBox at the foot of the list of already existing boxes of that type as defined in 5.4.2 while not changing `protection_count` in the ItemProtectionBox. Parsers are required to check for the existence of multiple ProtectionSchemeInfoBox despite `protection_count` is fixed to 1, because any change of content which has already been signed would render the appropriate signature invalid. An optional AdditionalUserInformationBox might be used in order to add information.

Only if the signature algorithm applied deviates from the default the ‘sigC’ box shall be present.

SignatureConfigurationBox

Box Types:	‘sigC’
Container:	Scheme Information Box (‘schi’)
Mandatory:	No
Quantity:	Zero or one per signing instance

Syntax
```
aligned(8) class SignatureConfigurationBox
      extends Box(‘sigC’) {

      bit(8)  AlgorithmIdentifier[];
}
```

Semantics
`AlgorithmIdentifier` is the signature algorithm identifier with optional parameters as defined by RFC 3280 and RFC 4055. It is encoded using the ASN.1 distinguished encoding rules (DER) and has the structure:
```
AlgorithmIdentifier  ::=  SEQUENCE  {
               algorithm               OBJECT IDENTIFIER,
               parameters              ANY DEFINED BY algorithm OPTIONAL  }
```

In order to include optional user information data related to an additional signature the ‘auib’ box is provided.

AdditionalUserInformationBox

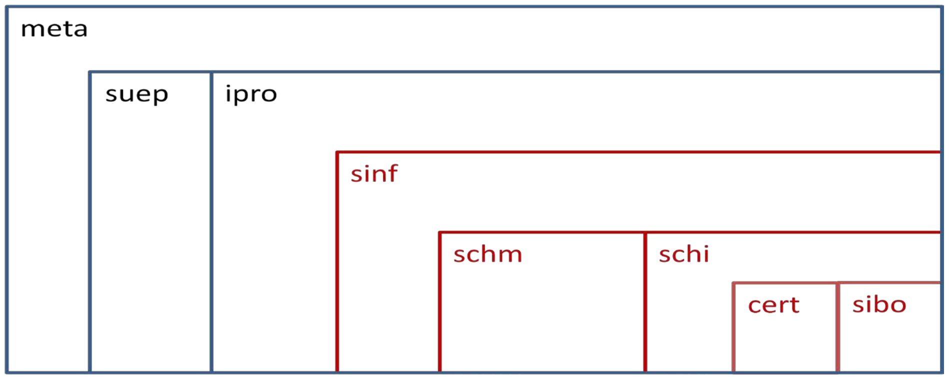
Box Types:	‘auib’
Container:	Scheme Information Box (‘schi’)
Mandatory:	No
Quantity:	Zero or one per signing instance

Syntax
```
aligned(8) class AdditionalUserInformationBox
      extends Box(‘auib’) {
      string  UserInformation;
}
```

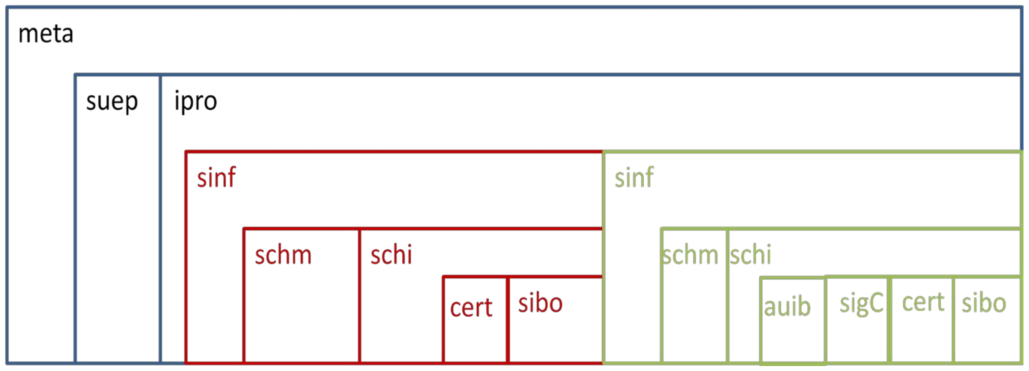
Semantics
`UserInformation` is a null terminated string in UTF-8 characters

Annex A. Repeated Signing (informative)

Characterization of the box arrangement defined in the present specification for data export:



Characterization of the box arrangement after repeated signing:



The red color represents the signature introduced at the export stage. The green color represents another signing operation happening after the export stage. The CertificateBox provides the public key for signature verification. Within the SignatureConfigurationBox information is contained describing a differing signature algorithm and its parameters. Additional information has been added in the AdditionalUserInformationBox..

In order to check validity of a signature the signature itself has to be taken from the SignatureBox and the bit values for the signature string be set to zero. The hashing operation is performed followed by the signature operation on the hash value. Now the two signatures can be compared.

Example: check validity of the first (red) signature from above

- Remove the (green) boxes created for the second signing
- Re-adjust box sizes of ‘ipro’ and ‘meta’ according to the size of removed ‘sinf’ box
- Read the public key from the red CertificateBox (do not change the box content)
- Take out the signature from the red SignatureBox
- Set the bit values of the red SignatureBox to zero
- Perform hash operation on the remaining file data
- Perform signature operation on the obtained hash value
- Compare the just generated signature with the signature taken out before

Annex B. Revision History

Rev.	Date	Editor	Changes
1.0	March 2013	Gero Bäse	First release