BSI Technical Guideline 03125  
Preservation of Evidence of Cryptographically Signed Documents  

Annex TR-ESOR-F:  
Formats

<table>
<thead>
<tr>
<th>Designation</th>
<th>Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbreviation</td>
<td>BSI TR-ESOR-F</td>
</tr>
<tr>
<td>Version</td>
<td>1.2</td>
</tr>
<tr>
<td>Date</td>
<td>31.01.2015</td>
</tr>
</tbody>
</table>
# Table of contents

1. Introduction............................................................................................................................ 4

2. Overview................................................................................................................................ 6

3. Definition of the archive data object (XAIP)........................................................................... 8
   3.1 Overview of the XAIP data structure – the `<XAIP>` element.............................................. 8
   3.2 The xaip:packageHeaderType............................................................................................ 9
   3.3 The xaip:metaDataSectionType....................................................................................... 14
   3.4 The xaip:objectsSectionType.......................................................................................... 17
   3.5 The xaip:credentialSectionType..................................................................................... 19
   3.6 The Delta-XAIP-element `<DXAIP>`............................................................................. 22

4. Payload data formats............................................................................................................. 24
   4.1 Meta data............................................................................................................................ 24
      4.1.1 Extensible Markup Language (XML). .............................................................................. 24
      4.1.2 XML Schema (XSD)........................................................................................................ 25
   4.2 Content data (object data)................................................................................................... 25
      4.2.1 Documents (records)........................................................................................................ 25
      4.2.2 Multimedia formats......................................................................................................... 30
   4.3 Base64 encoding.................................................................................................................. 33

5. Cryptographic data formats.................................................................................................... 34
   5.1 Signature formats................................................................................................................ 34
      5.1.1 PKCS#7 / CMS / CAdES.................................................................................................. 34
      5.1.2 XML signatures / XAdES............................................................................................... 34
      5.1.3 Other signature formats.................................................................................................... 35
   5.2 Certificate formats................................................................................................................ 35
   5.3 Certificate validation formats............................................................................................... 36
      5.3.1 Online Certificate Status Protocol (OCSP / RFC 2560 / RFC 6960)............................... 36
      5.3.2 Server-Based Certificate Validation Protocol (SCVP / RFC 5055)................................. 36
   5.4 Time stamp.......................................................................................................................... 37
   5.5 Evidence Record (pursuant to RFC 4998 /RFC 6283)......................................................... 37
      5.5.1 EvidenceRecord pursuant to RFC 4998........................................................................... 38
      5.5.2 `<EvidenceRecord>` pursuant to [RFC6283]................................................................. 39
   5.6 Recommendations for the implementation........................................................................... 41

6. Annex – XML schema definition............................................................................................ 42
1. Introduction

The goal of the Technical Guideline "Preservation of Evidence of Cryptographically Signed Documents" is to specify security-related requirements for the long-term preservation of evidence of cryptographically signed electronic documents and data along with associated electronic administrative data (meta data).\(^1\) A Middleware defined for this purpose (TR-ESOR-Middleware) in the sense of this Technical Guideline includes all the modules (M) and interfaces (S) ["S" for the German word "Schnittstellen"] used for securing and preserving the authenticity and proving the integrity of the stored documents and data.

The Reference Architecture introduced in the Main Document of this Technical Guideline consists of the functions and logical units described below:

- The S.4 input interface of the TR-ESOR-Middleware which serves to embed the TR-ESOR-Middleware in the existing IT and infrastructure landscape;
- The "ArchiSafe-Module" ([TR-ESOR-M.1]) which regulates the flow of information in the Middleware, implements the security requirements for the interfaces with the IT applications and ensures that the application systems are decoupled from the ECM/Long-Term Storage;
- The "Cryptographic-Module" [TR-ESOR-M.2] and the associated S.1 and S.3 interfaces that provide all the functions needed for creating (optional) and verifying electronic signatures, post-verifying electronic certificates and for obtaining qualified time stamps for the Middleware. Furthermore, it may provide functions for the encryption and decryption of data and documents;
- The "ArchiSig-Module" ([TR-ESOR-M.3]) with the S.6 interface that provides the functions needed for the preservation of evidence of the digitally signed documents;
- An ECM/Long-Term Storage with the S.2 and S.5 interfaces that assumes the physical archiving/storage and also the storage of the meta data that preserve evidence.

This ECM/Long-Term Storage is no longer directly a part of the Technical Guideline, but requirements will be set for it through the two interfaces that are still part of the TR-ESOR-Middleware.

The application layer that may include an XML adapter is not a direct part of the Technical Guideline either, even though this XML adapter may be implemented as part of a Middleware.

The IT Reference Architecture depicted in Figure 1 is based on the ArchiSafe\(^2\) Reference Architecture and is supposed to make possible and support the logical (functional) interoperability of future products with the goals and requirements of the Technical Guideline.

---

1. Note on the terminology used in this Annex:
   In this Annex F to the Technical Guideline 03125, the more specific technical term “data objects” is also used in some cases to refer to “information packages”.
2. For more information, see [http://www.archisafe.de](http://www.archisafe.de).
This Technical Guideline has a modular design and the individual annexes to the Main Document specify the functional and security-related requirements for the needed IT components and interfaces of the TR-ESOR-Middleware. The specifications are strictly platform-, product-, and manufacturer-independent.

This document bears the designation "Annex TR-ESOR-F" and specifies data formats that are suitable for the preservation of evidence of cryptographically signed documents from the perspective of the functional and legal requirements.

Figure 1: Schematic depiction of the IT Reference Architecture

This Technical Guideline has a modular design and the individual annexes to the Main Document specify the functional and security-related requirements for the needed IT components and interfaces of the TR-ESOR-Middleware. The specifications are strictly platform-, product-, and manufacturer-independent.

This document bears the designation "Annex TR-ESOR-F" and specifies data formats that are suitable for the preservation of evidence of cryptographically signed documents from the perspective of the functional and legal requirements.
2. Overview

The goal of the storage of electronic data with preservation of evidence over long periods of time is the provable, authentic, i.e. attributable and intact saving, conservation and availability of electronic data and meta data at least for the duration of the legally prescribed retention periods. In this respect, guaranteeing the negotiability is also part of ensuring the availability of electronic documents, i.e. the readability and context of the data and meta data in relation to the business processes at their basis over long periods of time using the IT systems typical at the time they are made available.

As a rule, the contents of electronic data and documents are coded by IT systems at the application level as a stream (sequence) of characters of a finite set of characters $Z_1$. The meta data is also coded as a sequence of the character set $Z_2$. The character sets $Z_1$ and $Z_2$ may, but do not have to be the same. Furthermore, the meta data satisfies certain syntactic and semantic rules that are founded in the specification of the meta data sets.

The associated meta data may be divided into two categories:

- A set of meta data $M_1$ contains information about the set of characters $Z_1$ used to code the documents and thus includes information regarding the presentation and formatting of the actual content data.
- A set of meta data $M_2$ contains additional descriptive meta information regarding the digital documents (e.g. creator, date, file reference number etc.) and thus ensures, for example, that the documents or data may be found again and assigned to a business process.

Therefore, the goal of the storage of electronic documents with preservation of evidence is to guarantee
- The authenticity (and thus the non-repudiation),
- The integrity (intactness) and the
- Availability

of the digital contents and its meta data and character sets for long periods of time, but at least as long as the duration of the statutory retention periods. The prerequisite for this are standardised, reliable and verifiable trusted data infrastructures and transactions for the electronic documents to be stored.

In order to ensure the long-term availability of the electronic documents stored, solely open, standardised and stable data formats that support a long-term and largely system- and platform-independent interpretability of the data should therefore be used both for the content data and for the meta data if possible. The primary intention of this requirement is to avoid a format transformation of the stored electronic documents at least for the duration of the legally prescribed retention periods, because this is connected to a significant amount of effort – in particular in the case of electronically signed documents.

In order to prove the integrity and authenticity of the data to be stored, this Technical Guideline stipulates the use of trusted cryptographic security measures that include both the actual content data and the "descriptive" meta data, reliably link them and are able to maintain and renew the evidential value of the cryptographic security measures for the duration of the statutory retention periods in conformity with the law.

Securing the authenticity of electronic data and documents also includes the reliable and permanent linking of the electronic documents stored with all meta information that is needed to find the documents and for the legally viable and auditable traceability (reconstruction) of the business processes on which the data is based.

The purpose of this specification is to define and describe electronic data formats and

---

3 If the original data has to first be converted into a standardised and long-term stable format and if the IT systems required to process the documents are expected to be available during the retention period planned, it may make sense to additionally store the original data in the original format.
transaction protocols that are suitable for adequately mapping and implementing the legal and functional requirements of long-term storage with preservation of evidence in the sense of this Technical Guideline.

Using the <XAIP> element, Section 3 of this document specifies an XML-based container format for archive data objects (XAIP) that is generated by Middleware components that conform to this Technical Guideline and, using the <DXAIP> element, a Delta-XAIP structure transmitted during the ArchiveUpdateRequest (see [TR-ESOR-E]).

Section 4 of this document then describes electronic data formats for the content data (primary information) and the meta data that are recommended for the long-term storage of payload data primarily with regard to long-term availability and machine readability and interpretability at the time this Technical Guideline was published.

Section 5 of this document describes structures, formats and algorithms for the generation and interpretation of cryptographic data that are suitable for the long-term securing of the integrity, authenticity, and probative value of electronic documents at the time this Technical Guideline was published.

Finally, the annex to this document (see Section 6) contains the complete specified XML schema.
3. Definition of the archive data object (XAIP)

An archive data object, i.e. an electronic document in the sense of this Technical Guideline intended for the long-term storage in an electronic archive system, is a self-explanatory and well-formed XML document that may be verified against a valid and authorised XML schema (also called **XML-formatted archive data object** or **XAIP** for short in the following). Such an archive data object contains all content data (primary information) and meta information that is needed for a reliable and complete reconstruction of business processes or administrative procedures up until the expiry of the legally prescribed retention periods.

The description of the archive data objects in a valid XML schema ensures that:

- The archive data objects may be evaluated for syntactic correctness prior to the submission to the electronic long-term storage,
- Necessary additions to and extensions of the meta data may be made with little effort by adding and extending existing meta data structures or including additional XML schemata, and
- The cryptographic security measures needed for proving the authenticity and integrity of data subject to the duty to retain certain documents on account of legal requirements, such as electronic signatures or electronic time stamps may be permanently and reliably linked to the data to be secured.

The following sections of this document first describe basic syntactic and semantic structures that an archive data object that conforms to the goals and requirements of this Technical Guideline should implement. They are based in large part on agreements and experiences of the ArchiSafe project of the German National Meteorology Institute (Physikalisch-Technische Bundesanstalt) [PTB 05] and concepts of the OAIS reference model [OAIS], the Metadata Encoding and Transmission Standards [METS], the Victorian Electronic Records Strategy [VERS] and the XML Formatted Data Unit [XFDU] Standards of the Consultative Committee for Space Data Systems (CCSDS) of the American Aeronautics and Space Administration (NASA).

In the definition and description of the XML data structures, this specification differentiates between binding (obligatory) and optional data elements. A binding data element shall be available in an archive data object that conforms to this specification and be interpretable by an electronic archive system that conforms to this Technical Guideline. An optional element may occur. It does not necessarily have to be interpretable, but it may not hinder the automatic processing (interpretation) of other elements. An optional element shall occur if its presence is linked to certain conditions, such as the presence of electronic signatures, and these conditions have been fulfilled.

The following requirement applies:

(A3-1) For the preservation of evidence of cryptographically signed documents, the XAIP format described in this Section and specified by the corresponding XML schema should be used. Deviations in the XML format used are permissible, but it shall be explained that equivalent functionality is supported. It shall be explained in particular how a transformation into the XAIP format as specified here may be performed.

3.1 Overview of the XAIP data structure – the **<XAIP>** element

An archive data object pursuant to this Technical Guideline consists of an **<XAIP>** element that is defined as follows:

```xml
<xsd:element name="XAIP" type="xaip:XAIPType"/>
```

The **xaip:XAIPType** is defined as follows:
The `xaip:XAIPType` includes the following elements:

- `<packageHeader>` [required]
  The `<packageHeader>` element contains general information about the archive data object and describes, for example, the logical structure of the archive data object. Further information on the `xaip:packageHeaderType` can be found in Section 3.2.

- `<metaDataSection>` [optional]
  The `<metaDataSection>` element includes meta information for the description of the business and archiving context, if any. The metaDataSection should contain all information that is needed for the transparent and long-term interpretation of the business and archiving context. Further information on the `xaip:metaDataSectionType` can be found in Section 3.3.

- `<dataObjectsSection>` [optional]
  The `<dataObjectsSection>` element includes the payload data of the archive data object, if any. This element may, for example, be used to store content data in different platform- or application-specific data formats in an XAIP container or to archive entire files or records together with many different documents. Further information on the `xaip:dataObjectsSectionType` can be found in Section 3.4.

- `<credentialsSection>` [optional]
  If necessary, the `<credentialsSection>` element includes credential objects in the form of Evidence Records or supplemental evidence data, such as signatures, time stamps, certificates or signature verification information. Further information on the `xaip:credentialsSectionType` can be found in Section 3.5.

### 3.2 The xaip:packageHeaderType

The `xaip:packageHeaderType` is used to define the XAIP element and is defined as follows:

```xml
<xs:complexType name="xaip:packageHeaderType">
  <xs:sequence>
    <xs:element name="AOID" type="xs:string" maxOccurs="1" minOccurs="0"/>
    <xs:element name="packageInfo" type="xs:string" minOccurs="0"/>
    <xs:element name="versionManifest" type="xaip:versionManifestType" maxOccurs="unbounded" minOccurs="1"/>
    <xs:element ref="ds:CanonicalizationMethod" maxOccurs="1" minOccurs="0"/>
    <xs:element name="extension" type="xaip:extensionType"/>
  </xs:sequence>
</xs:complexType>
```

```xml
The xaip:XAIPType includes the following elements:

- `<packageHeader>` [required]
  The `<packageHeader>` element contains general information about the archive data object and describes, for example, the logical structure of the archive data object. Further information on the `xaip:packageHeaderType` can be found in Section 3.2.

- `<metaDataSection>` [optional]
  The `<metaDataSection>` element includes meta information for the description of the business and archiving context, if any. The `metaDataSection` should contain all information that is needed for the transparent and long-term interpretation of the business and archiving context. Further information on the `xaip:metaDataSectionType` can be found in Section 3.3.

- `<dataObjectsSection>` [optional]
  The `<dataObjectsSection>` element includes the payload data of the archive data object, if any. This element may, for example, be used to store content data in different platform- or application-specific data formats in an XAIP container or to archive entire files or records together with many different documents. Further information on the `xaip:dataObjectsSectionType` can be found in Section 3.4.

- `<credentialsSection>` [optional]
  If necessary, the `<credentialsSection>` element includes credential objects in the form of Evidence Records or supplemental evidence data, such as signatures, time stamps, certificates or signature verification information. Further information on the `xaip:credentialsSectionType` can be found in Section 3.5.

### 3.2 The xaip:packageHeaderType

The `xaip:packageHeaderType` is used to define the XAIP element and is defined as follows:

```xml
<xs:complexType name="xaip:packageHeaderType">
  <xs:sequence>
    <xs:element name="AOID" type="xs:string" maxOccurs="1" minOccurs="0"/>
    <xs:element name="packageInfo" type="xs:string" minOccurs="0"/>
    <xs:element name="versionManifest" type="xaip:versionManifestType" maxOccurs="unbounded" minOccurs="1"/>
    <xs:element ref="ds:CanonicalizationMethod" maxOccurs="1" minOccurs="0"/>
    <xs:element name="extension" type="xaip:extensionType"/>
  </xs:sequence>
</xs:complexType>
```
As a rule, the <AOID> element is used as an unique identifier of the archive data object generated by the TR-ESOR Middleware or the ECM/Long-Term Storage. The internal structure of this identifier is not defined by this specification; the definition is left to the manufacturer or user of the TR-ESOR-Middleware.

The <packageInfo> element may include basic information about the archive data object in the text format, so that future users are also able to understand the format of the XAIP document and to interpret the contents.

The <versionManifest> element is used as a "version-specific table of contents" and may occur several times. In each case, it contains information about a version of the archive data object. The structure of the xaip:VersionManifestType is explained in more detail below.

The <CanonicalizationMethod> element specifies the canonicalisation method to be applied. If this element is missing, the canonicalisation is carried out by the TR-ESOR-Middleware using the standard algorithm (C14N - Canonical XML [C14N]).

With the optional <extension> element, user-specific extensions may be made. It is recommended to coordinate these extensions and XAIP profiles resulting from this with the Federal Office for Information Security (BSI).

With the packageID attribute, a unique identifier of the <packageHeader> element is available, which may serve as a reference point within the archive data object if necessary. If the <packageHeader> element is to be cryptographically protected, a <protectedObjectPointer> element is used to refer to this packageID attribute. The packageID attribute may be generated by the business application and serves in it as a further identifier.

The structure of the xaip:VersionManifestType is defined as follows:

The structure of the xaip:extensionType is defined as follows:

---

4 The <AOID> element is optional to make the generation of the AOID by the requested component as part of the ArchiveSubmissionRequest possible (see Annex E). If this element is not already available, it shall be generated by the TR-ESOR-Middleware or the ECM/Long-Term Storage.
<versionInfo> [optional]

The <versionInfo> element contains information on the corresponding version of
the archive data object in the text format.

<preservationInfo> [required]

The <preservationInfo> element includes information on the storage (e.g. retention period, archiving appraisal) of the archive data object. The structure of the xaip:preservationInfoType is explained in more detail below.

<submissionInfo> [optional]

The <submissionInfo> element contains information on the sender of the archive data object. The structure of the xaip:submissionInfoType is explained in more detail below.

<packageInfoUnit> [required, unbounded]

The <packageInfoUnit> element includes information on a certain content data unit. This element may occur several times. The structure of the xaip:packageInfoUnitType is explained in more detail below.

<extension> [optional]

With the optional <extension> element, user-specific extensions may be made. The structure of the xaip:extensionType is depicted on page 10.

It is recommended to coordinate these extensions and XAIP profiles resulting from this with the Federal Office for Information Security (BSI).

@VersionID [required]

With the VersionID attribute, an unique identifier of the version of the archive data object is available. The VersionID attribute should be created in the form "v1", "v2", "v3" etc.

The structure of the xaip:preservationInfoType is defined as follows:

<retentionPeriod> [required]

The <retentionPeriod> element specifies the date until which the documents have to be stored. This element is evaluated during the ArchiveDeletionRequest (see [TR-ESOR-E]).

$status> [optional]
The `<status>` element may contain information on the status of the archive data object, which may be evaluated before the archive data object is deleted.\(^5\)

The concrete assignment and evaluation of this element is not the subject of this specification. These determinations should rather be the subject of the XAIP profiles. It is recommended to coordinate such XAIP profiles with the Federal Office for Information Security (BSI).

`<extension>` [optional]

With the optional `<extension>` element, user-specific extensions may be made. The structure of the `xaip:extensionType` is depicted on page 10.

It is recommended to coordinate these extensions and XAIP profiles resulting from this with the Federal Office for Information Security (BSI).

The structure of the `xaip:submissionInfoType` is defined as follows:

```xml
<xs:complexType name="submissionInfoType">
  <xs:sequence>
    <xs:element name="clientID" type="saml:NameIDType"/>
    <xs:element name="submissionUnit" type="saml:NameIDType" maxOccurs="1" minOccurs="0"/>
    <xs:element name="submissionAuthor" type="saml:NameIDType" maxOccurs="1" minOccurs="0"/>
    <xs:element name="submissionTime" type="xs:dateTime" maxOccurs="1" minOccurs="0"/>
    <xs:element name="extension" type="xaip:extensionType" maxOccurs="1" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

`<clientID>` [required]

The `<clientID>` element includes the identifier of the business application making the request. Further details on this element are specified in [SAMLv2], Section 2.2.2. This element may be derived by the TR-ESOR-Middleware from the available authentication information.

`<submissionUnit>` [optional]

If necessary, the `<submissionUnit>` element designates the organisational unit of the business application making the request. The structure of the element is specified in [SAMLv2], Section 2.2.2.

`<submissionAuthor>` [optional]

If necessary, the `<submissionAuthor>` element designates the author or sender of the archive data object. The structure of the element is specified in [SAMLv2], Section 2.2.2.

`<submissionTime>` [optional]

The `<submissionTime>` element should be inserted by the business application making the request and indicates the time the archive data object was transmitted.

`<extension>` [optional]

With the optional `<extension>` element, user-specific extensions may be made. The structure of the `xaip:extensionType` is depicted on page 10.

It is recommended to coordinate these extensions and XAIP profiles resulting from this with the Federal Office for Information Security (BSI).

The structure of the `xaip:packageInfoUnitType` is defined as follows:

\(^5\) With this element, the "appraisal note" required in the government agency environment may be realised in particular.
<unitType> [optional]

The <unitType> element indicates the type of the respective content data unit. The possible forms of the <unitType> element should be defined as part of the XAIP profiles. It is recommended to coordinate the XAIP profiles with the Federal Office for Information Security (BSI).

<textInfo> [optional]

The <textInfo> element contains information about the stored content data unit.

<protectedObjectPointer> [required, unbounded]

With the number of the <protectedObjectPointer> elements indicated here, it is defined which parts of the archive data object are included in the generation of hash values and are therefore protected by the corresponding Evidence Record. With respect to the details of the hash value generation, a distinction is made between payload data and supplemental evidence data on the one hand and meta data and other XAIP-specific XML structures, such as VersionManifest and packageInfoUnit, on the other hand:

For payload data and supplemental evidence data, the actual data – without enclosing XML tags and in its original format (i.e. without Base64 coding) – is included in the generation of hash values so that such data that has already been protected by Evidence Records may be imported seamlessly and embedded in an XAIP container. For meta data and other XAIP-specific XML structures however, the complete XML element – including the enclosing XML tags and the XML attributes that may be contained therein – is canonicalized with the specified canonicalization procedures and then included in the generation of hash values.\(^6\)

The hash values that were generated in this manner form a data object group in the sense of [RFC 4998] and [RFC 6283].

<unprotectedObjectPointer> [optional, unbounded]

With an unprotectedObjectPointer indicated here, it is clarified that the object which is referred to here logically belongs to the indicated XAIP version. This

\(^6\) For example, the packageID attribute (see page 10) could be referred to here in order to include the packageHeader element in the generation of the hash tree. In this case, however, it must be taken into account that the packageHeader structure would be changed by further updates and that the cryptographic protection of the packageHeader element therefore often only makes sense if no further updates are to be expected.
object, however, is *not* included in the generation of hash values and it is therefore not covered and cryptographically protected by an Evidence Record. For example, such an element may be used to refer to an Evidence Record that belongs to this version and may be stored in the CredentialSection.

```
<packageInfoUnit> [optional, unbounded]
  An packageInfoUnit element may, in turn, include a sequence of
  packageInfoUnit elements so that hierarchically nested structures may thus be
  mapped.
</extension> [optional]
  With the optional <extension> element, user-specific extensions may be made.
  The structure of the xaip:extensionType is depicted on page 10.
  It is recommended to coordinate these extensions and XAIP profiles resulting from
  this with the Federal Office for Information Security (BSI).
```

@packageUnitID [required]
  The packageUnitID attribute is used as the unique identifier of the content data unit.

### 3.3 The xaip:metaDataSectionType

The structure of the *xaip:metaDataSectionType* is defined as follows:

```
<xs:complexType name="metaDataSectionType">
  <xs:sequence>
    <xs:element ref="xaip:metaDataObject" maxOccurs="unbounded" minOccurs="1"/>
  </xs:sequence>
</xs:complexType>
```

```
The xaip:metaDataSectionType consists of a sequence of
xaip:metaDataObject elements of the xaip:metaDataObjectType in which the
corresponding meta data is stored.
The structure of the xaip:metaDataObjectType is defined as follows:
```

```
<xs:complexType name="metaDataObjectType">
  <xs:complexContent>
    <xs:restriction base="xs:anyType">
      <xs:attribute name="metaDataID" type="xs:ID" use="required"/>
      <xs:attribute name="dataObjectID" type="xs:IDREF" use="required"/>
      <xs:attribute name="category" type="xs:string"/>
      <xs:attribute name="classification" type="xs:string"/>
      <xs:attribute name="type" type="xs:string"/>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

Pursuant to this, an xaip:metaDataObject may have any structure and include the attributes listed below:

@metaDataID [required]
  The metaDataID attribute is used as the unique identifier of the meta data object so that it can be referred to.

@dataObjectID [required]
With the `dataObjectID` attribute, it is clarified which data object or which XAIP structure element this meta data object refers to. However, it must be taken into account here that the `dataObjectID` of a binary data object (see Section 3.4) is not protected by cryptographic mechanisms and that a meta data object could therefore later be assigned by the `protectedObjectPointer` mechanism explained above to another data object without being noticed. In order to avoid this ambiguity, a meta data object that refers to a binary data object should contain a `dataObjectCheckSum` element explained below by means of which the data object is identified in a unique manner.

@category [optional]

The `category` attribute determines the general category to which the meta data object is assigned.

@classification [optional]

The `classification` attribute provides further information on the more detailed classification of the meta data object within the category defined by the `category` attribute.

@type [optional]

The `type` attribute indicates the concrete type of the meta data object. This attribute determines the internal syntactic structure and semantics of the `metaDataObject` element.

The `type` attributes used and the corresponding subelements of the `xaip:metaDataObject` element should be laid down in XAIP profiles. It is recommended to coordinate these XAIP profiles with the Federal Office for Information Security (BSI).

Unless otherwise specified by XAIP profiling, the following values that are defined in [XFDU] and correspond to the OAIS model should be used for the `category` and `classification` attributes:

<table>
<thead>
<tr>
<th>Category</th>
<th>Classification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMD(^8)</td>
<td>DESCRIPTION</td>
<td>Descriptive content meta data that describes the referenced data object in more detail to enable or support, for example, finding and accessing the data object.</td>
</tr>
<tr>
<td>OTHER</td>
<td>Other descriptive meta data.</td>
<td></td>
</tr>
<tr>
<td>REP(^9)</td>
<td>SYNTAX</td>
<td>Meta data that describes the syntax of another data object in more detail.(^10)</td>
</tr>
<tr>
<td>DED(^11)</td>
<td>Meta data that describes the</td>
<td></td>
</tr>
</tbody>
</table>

---

\(^7\) A meta data object may be assigned to a payload information package (`dataObject`), an credential object (`credential`), another meta data object (`metaDataObject`) or an XAIP structure element.

\(^8\) DMD = Descriptive Meta Data

\(^9\) REP = Representation Meta Data

\(^10\) Here, the data format used should be described in a manner that is precise enough for processing at a later point in time. For example, XML schema files that may be used for a schema validation of application-specific data objects may be stored here.
### Category Classification Description

<table>
<thead>
<tr>
<th>Category</th>
<th>Classification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTHER</td>
<td>semantics of another data object in more detail.</td>
<td>Other meta data that supports the representation of another data object.</td>
</tr>
<tr>
<td>PDI</td>
<td>REFERENCE</td>
<td>Meta data that describes the generation of identifiers and makes it possible for external systems to access an archive data object.</td>
</tr>
<tr>
<td>CONTEXT</td>
<td>Meta data that describes the environment in which the archive data object was developed.</td>
<td></td>
</tr>
<tr>
<td>PROVENANCE</td>
<td>Meta data that documents the history of the archive data object.</td>
<td></td>
</tr>
<tr>
<td>OTHER</td>
<td>Other storage-specific meta data.</td>
<td></td>
</tr>
<tr>
<td>OTHER</td>
<td>Using the category &quot;OTHER&quot;, meta data that cannot be assigned clearly to one of the categories mentioned above (DMD, REP, PDI) can be stored.</td>
<td></td>
</tr>
</tbody>
</table>

**<xs:element name="dataObjectCheckSum" type="xaip:checkSumType"/>**

<dataObjectCheckSum> [optional] If necessary, the dataObjectCheckSum element may be used to insert the cryptographic hash value of the data object to which this meta data object relates and which is referred to with the dataObjectID attribute mentioned below into the meta data object so that the assignment between the data object on which it is based and this meta data object is ensured with cryptographic security measures.

### 3.4 The xaiP:dataObjectsSectionType

The structure of the xaiP:dataObjectsSectionType is defined as follows:

---

11 DED = Data Entity Dictionary
12 For example, ontology files in which the semantics of application-specific data objects are described may be stored here.
13 PDI = Preservation Description Information
14 It should be noted that no meta data of the FIXITY class occur in the XAIP, since the elements to protect the integrity and authenticity are stored in the XAIP in the CredentialSection.
15 The meta data of the category PDI may be both, content meta data (e.g. record context, processing and protocol information) as well as technical meta data (e.g. information on on the conversions carried out and on the software and hardware environment).
16 For the specification of meta data of the PROVENANCE class, the events defined in [PREMIS], for example, may be used.
The `xaip:dataObjectsSectionType` consists of a sequence of `xaip:dataObject` elements of the `xaip:dataObjectType`.

The structure of the `xaip:dataObjectType` is defined as follows:

```xml
<xs:complexType name="dataObjectType">
  <xs:sequence>
    <xs:choice>
      <xs:element name="binaryData" type="xaip:binaryDataType" minOccurs="0"/>
      <xs:element name="xmlData" type="dss:AnyType" minOccurs="0"/>
    </xs:choice>
    <xs:element name="checkSum" type="xaip:checkSumType" minOccurs="0"/>
    <xs:element name="transformInfo" type="xaip:transformInfoType" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="dataObjectID" type="xs:ID" use="required"/>
</xs:complexType>
```

`<binaryData>` [choice]

This element is used to store binary data. In this respect, the data shall be encoded pursuant to [RFC4648] Base64 and the type of the data should be indicated in the `MimeType` attribute pursuant to the list of registered media types maintained by IANA. The TR-ESOR-Middleware should perform a corresponding validation of the data transmitted with respect to the data format specified by the `MimeType` attribute.

`<xmlData>` [choice]

This element is used to store XML data, using the `dss:AnyType` defined in [OASIS-DSS]. The TR-ESOR-Middleware should perform a corresponding validation of the data transmitted with respect to XML schema files that were defined administratively.

`<checksum>` [optional]

If necessary, this element contains the cryptographic checksum of the data object. The subject of the checksum calculation is the Base64-decoded payload data object for `<binaryData>` elements and the XML data canonicalized pursuant to the `<packageHeader>/<CanonicalizationMethod>` for `<xmlData>` elements. Further details on the `xaip:checkSumType` are explained in more detail below. In this respect, the checksum always relates to the payload data as it is actually

---

17 See [http://www.iana.org/assignments/media-types/media-types.xhtml](http://www.iana.org/assignments/media-types/media-types.xhtml).
available in the XAIP, i.e. especially taking possible transformations that have already been carried out into account.

<transformInfo> [optional]
This element is used to document one or several operations (transformations) that had been carried out on the original data before it was stored in the archive system\(^{18}\) in order to repeat these steps if necessary or – if it is a reversible transformation – to be able to reverse them in an automated manner. The syntax of the \texttt{xaip:transformInfoType} is defined below.

Further details shall be defined as part of a corresponding XAIP profile. It is recommended to coordinate XAIP profiles with the Federal Office for Information Security (BSI).

If a transformation also includes manual procedures or process steps that were specified incompletely, which could not be reversed completely in an automated manner, the \texttt{transformInfo} element should not be used. In such a case, using a suitable meta data object with the \texttt{category=PDI} and \texttt{classification=PROVENANCE} is instead recommended.

@dataObjectID [required]
The \texttt{dataObjectID} attribute is used as the unique identifier of the data object.

The structure of the \texttt{xaip:checkSumType} is defined as follows:

\[
\begin{align*}
\text{<xs:complexType name="checkSumType">} \\
\text{<xs:sequence>} \\
\text{\hspace{1em}<xs:element name="checkSumAlgorithm" type="xs:anyURI" />} \\
\text{\hspace{1em}<xs:element name="checkSum" type="xs:hexBinary" />} \\
\text{</xs:sequence>} \\
\text{</xs:complexType>}
\end{align*}
\]

\[<\text{checkSumAlgorithm}> [required]\]
This element specifies the algorithm used to create the checksum. Pursuant to this specification, the following algorithms may occur:

- http://www.w3.org/2000/09/xmldsig#sha1
- http://www.w3.org/2001/04/xmldsig-more#sha224
- http://www.w3.org/2001/04/xmlenc#sha256
- http://www.w3.org/2001/04/xmldsig-more#sha384
- http://www.w3.org/2001/04/xmlenc#sha512
- http://www.w3.org/2001/04/xmlenc#ripemd160

A restriction of the algorithms to be used or the specification of additional algorithms may be made as part of an XAIP profile.

\[<\text{checkSum}> [required]\]
This element includes the cryptographic checksum calculated using the algorithm mentioned above.

The structure of the \texttt{xaip:transformInfoType} includes a sequence of \texttt{transformObject} elements and is defined as follows:

\[
\begin{align*}
\text{<xs:complexType name="transformInfoType">} \\
\text{<xs:sequence>} \\
\text{\hspace{1em}<xs:element name="transformObject" type="xaip:transformObjectType" maxOccurs="unbounded">} \\
\text{</xs:element>} \\
\text{</xs:sequence>} \\
\text{</xs:complexType>}
\end{align*}
\]

\[\text{\textsuperscript{18} For example, data stored in encrypted form may be decrypted for processing. Moreover, the transformInfo element may be used to describe necessary decoding and conversion steps. When processing binary data, however, the description of the obligatory Base64 encoding does not have to be indicated. Here, the checksum rather refers to initial original data anyway.}\]
The structure of the **xaip:transformObjectType** is defined as follows:

```xml
<xs:complexType name="transformObjectType">
  <xs:sequence>
    <xs:element name="transformAlgorithm" type="xs:anyURI" />
    <xs:element name="Parameters" type="xs:anyType" maxOccurs="1" minOccurs="0" />
  </xs:element>
  <xs:attribute name="transformObjectID" type="xs:ID" use="required" />
  <xs:attribute name="order" type="xs:string" />
</xs:complexType>
```

**<transformAlgorithm> [required]**

This element specifies the algorithm used for the transformation; for example, the algorithm may be a compression-, canonicalisation- or encryption-algorithm.

**<parameters> [optional]**

If necessary, this element includes further parameters which are required to carry out the transformation. The detailed structure of this element depends on the transformation algorithm used.

**@transformObjectID [required]**

The transformObjectID attribute is used as the unique identifier of the transformation object.

**@order [optional]**

If several reversible transformations are available, the order attribute specifies in which order the sequence of transformations may be reversed. Based on [XFDU], the order attribute is a positive integer and the beginning of the processing starts with the number "1" and is incremented in each step.

The permissible algorithms and parameters shall be defined as part of an XAIP profile. It is recommended to coordinate such an XAIP profile with the Federal Office for Information Security (BSI).

### 3.5 The xaip:credentialSectionType

The structure of the **xaip:credentialSectionType** is defined as follows:

```xml
<xs:complexType name="credentialsSectionType">
  <xs:sequence>
    <xs:element ref="xaip:credential" maxOccurs="unbounded" minOccurs="1" />
  </xs:element>
</xs:complexType>
```

**<credential> [required, unbounded]**

The xaip:credentialSectionType consists of a sequence of xaip:credential elements of the xaip:credentialType that contain "evidence records" or "supplemental evidence data".

19 "Evidence records" serve to prove the intactness of the integrity and authenticity of the archived data objects. In accordance with the specifications of the IETF's ERS standard, a set of evidence records includes a set of archive time stamps of sufficient quality about the stored (signed) archive data objects that prove the intactness of the data and additional information that prove the correctness and validity of electronic signatures at the time of signing as well as the timely and legally compliant signature renewal. "Supplemental evidence data" are signatures or time stamps for exactly one data object or document and it also includes the verification data necessary for the verification of the signature or time stamp signature, such as certificates as well as CRL lists and OCSP responses to these certificates. The credential objects prove that the document was not changed any more after being archived. Supplemental evidence data proves that the signatures and...
The structure of the **xaip:credentialType** is defined as follows:

```xml
<xs:complexType name="credentialType">
    <xs:choice>
        <xs:element ref="dss:SignatureObject" />
        <xs:element name="certificateValues" type="xades:CertificateValuesType" />
        <xs:element name="revocationValues" type="xades:RevocationValuesType" />
        <xs:element ref="xaip:evidenceRecord" />
        <xs:element ref="vr:VerificationReport" />
        <xs:element name="other" type="xaip:extensionType"/>
    </xs:choice>
    <xs:attribute name="relatedObjects" type="xs:IDREFS"/>
    <xs:attribute name="credentialID" type="xs:ID" use="required"/>
</xs:complexType>
```

**<dss:SignatureObject> [choice]**

In this element, signature objects and time stamps for payload or meta data objects which are referred to with the relatedObjects attribute may be stored if necessary. The structure of the dss:SignatureObject element is defined in [OASIS-DSS].

**<certificateValues> [choice]**

In this element, certificates that are needed for the verification of signatures and time stamps may be stored if necessary. The structure of this element is defined in [XAdES]. In this case, the relatedObjects attribute should refer to the corresponding signatures or time stamps.

**<revocationValues> [choice]**

In this element, certificate status information that is needed for the verification of signatures and time stamps may be stored if necessary. The structure of this element is defined in [XAdES]. In this case, the relatedObjects attribute should refer to the corresponding signatures or time stamps.

**<evidenceRecord> [choice]**

In this element, credential objects in the form of Evidence Records pursuant to [RFC4998] and [RFC6283] may be stored if necessary. The structure of the xaip:evidenceRecord element is explained in more detail below. Pursuant to this specification, an Evidence Record usually relates to a certain XAIP version and the relatedObjects attribute should refer to the corresponding versionID attribute of the <versionManifest> element. With this reference, the corresponding <protectedObjectPointer> elements may be determined if necessary, clarifying which data objects the Evidence Record exactly refers to. Moreover, the description of the <protectedObjectPointer> element (see page 13) clarifies how exactly the hash values are generated for the different objects.

**<vr:VerificationReport> [choice]**

In this element, the results of the signature verification that was carried out should be stored. As explained in more detail in [OASIS-VR], a vr:VerificationReport

---

20 Federal Office for Information Security
element contains a sequence of IndividualReport elements, in which the verification results for individual credential objects are included. If a vr:VerificationReport includes several IndividualReport elements, the verified credential object shall be referred to in the WhichDocument attribute of the IndividualReport/SignedObjectIdentifier element. Details on the vr:VerificationReport element are defined in [OASIS-VR].

In the other element, other supplemental evidence data and evidence records may be stored.

If necessary, the relatedObjects attribute refers to the object(s) which the credential object (credential) refers to. If the credential object is an evidenceRecord element, the relatedObjects attribute refers to the associated VersionID attribute. In case of a vr:VerificationReport element, this attribute refers to the verified object that may, for example, be a dss:SignatureObject or an evidenceRecord.

If necessary, the credentialID attribute is used as the unique identifier of the credential object (credential).

The xaip:evidenceRecord element is based on the xaip:evidenceRecordType which is defined as follows on the basis of the ec:EvidenceRecordType described in more detail in [eCard-2]:

```xml
<xs:element name="evidenceRecord" type="xaip:EvidenceRecordType" />
<xs:complexType name="EvidenceRecordType">
  <xs:complexContent>
    <xs:extension base="ec:EvidenceRecordType">
      <xs:attribute name="AOID" type="xs:string" />
      <xs:attribute name="VersionID" type="xs:string" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The ec:evidenceRecordType as specified in [eCard-2] is defined as follows:

```xml
<complexType name="EvidenceRecordType">
  <choice>
    <element name="xmlEvidenceRecord" type="ers:EvidenceRecordType" />
    <element name="asn1EvidenceRecord" type="base64Binary" />
  </choice>
</complexType>
```

XML-based Evidence Records pursuant to [RFC6283] are stored in the xmlEvidenceRecord element. ASN.1-based Evidence Records pursuant to [RFC4998] are stored Base64-encoded in the asn1EvidenceRecord element.

If the xaip:evidenceRecord element is embedded in a corresponding credential element within an XAIP element as in this case, the additional attributes (AOID and VersionID) may be omitted.

3.6 The Delta-XAIP-element <DXAIP>

As part of the ArchiveUpdate function (see Annex [TR-ESOR-E]), a <DXAIP> element is transmitted which contains an additional updateSection of the xaip:updateSectionType in addition to the usual <XAIP> structure (see Section 3) and is specified as follows:

```xml
<x:s:element name="DXAIP" type="xaip:DXAIPType" />
```
Thus, the <DXAIP> element includes the following subelements:

<packageHeader>[required]

The <packageHeader> element in the <DXAIP> element consists of the following subelements:

- <AOID>[required] – specifies the archive data object to be supplemented by the Delta-XAIP-element.
- <packageInfo>[optional] – if this element does not exist yet in the XAIP, it is created accordingly. If the element is already available, the element transmitted is ignored and a corresponding warning is returned.
- <versionManifest>[required] – it shall be available exactly once and specifies the new version of the archive data object. The versionID attribute contained in this element shall not be among the identifiers already assigned in the existing <XAIP> element and should be determined by incrementing the version indicated in the <updateSection> element. The <protectedObjectPointer> and <unprotectedObjectPointer> elements contained therein shall either refer to the newly transmitted meta, payload or credential data objects or to corresponding place holders in the <updateSection> element.

Data may be deleted logically in a new version by removing the corresponding <protectedObjectPointer> or <unprotectedObjectPointer> elements.

<metaDataSection>[optional]

The <metaDataSection> element may contain additional meta information on the description of the business and archiving context. Further information on the xaip:metaDataSectionType can be found in Section 3.3.

<dataObjectsSection>[optional]

The <dataObjectsSection> element may include additional payload data of the archive data object. Further information on the xaip:dataObjectsSectionType can be found in Section 3.4.

<credentialsSection>[optional]

The <credentialsSection> element may include additional credential objects in the form of Evidence Records or supplemental evidence data, such as signatures, time stamps, certificates or signature verification information. Further information on the xaip:credentialsSectionType can be found in Section 3.5.

<updateSection>[required]

The <updateSection> element shall include information on which version of the archive data object the update refers to and, if necessary, further place holder elements which bear the ID attribute of a meta, payload or credential data object that was taken over without any changes from a previous version.

The structure of the xaip:updateSectionType is defined as follows:
The `<prevVersion>` element shall contain the unique identifier of the archive data object version which the update process refers to. Further information on the `VersionID` attribute of the `versionManifest` element can be found on page 11. To ensure that the update process can be carried out successfully, the `<prevVersion>` element transmitted shall correspond to the `VersionID` attribute of the current archive data object version.

If necessary, the `<placeHolder>` elements make the missing ID attributes from meta, payload or credential data objects of a previous version without any changes available.

The structure of the `xaip:placeHolderType` is defined as follows:

```
<x:complexType name="placeHolderType">
  <x:attribute name="objectID" type="xs:ID" use="required"/>
</x:complexType>
```

Thus, the `<placeHolder>` element mentioned above contains exactly the `objectID` attribute:

@objectID [required]

The `objectID` attribute of the `<placeHolder>` element shall correspond to the ID attribute of a meta, payload or credential data object that is already included in a previous version of the archive data object.
4. Payload data formats

The following section describes electronic data formats that are recommended for the long-term storage of payload data primarily with regard to long-term availability and machine readability and interpretability at the time this Technical Guideline was published.\(^{24}\) Payload data includes both the actual content data (primary information or also object data) and the meta data that describes the business and archiving context.

4.1 Meta data

In the broadest sense, meta data is data that describes other data. Meta data is markup data that describes the structures and context of data during the processing of the data by IT systems that create, process, administer and save the data. The meta data of an archive data object serves to identify and reconstruct the administrative or business context of the content data stored.

XML-based solutions have established themselves as the global standard for a cross-platform exchange of data for the structural markup (description) of electronic documents i.e. the mapping and machine-readable description of document structures.\(^{25}\)

An archive data object, i.e. an electronic document in the sense of this Technical Guideline intended for the long-term storage in an electronic archive system, is therefore a self-explanatory and well-formed XML document that may be verified against a valid and authorised XML schema.

4.1.1 Extensible Markup Language (XML)

The Extensible Markup Language [XML] is a format description language developed primarily for the Internet for the exchange of structured data and was standardised in 1997 by World Wide Web Consortium (W3C).\(^{26}\)

On the syntax level, XML as a text-based meta markup language does not only support the description, but above all it also the automatic display, manipulation and processing of logically structured data and, furthermore, it is characterised by good expandability and a high degree of flexibility.

On the semantic level, rules and structure definitions in XML syntax (XML schema) support the mapping of structured content models. XML schemata do not only allow a formal and machine-readable description of an XML vocabulary allowed for the exchange of data, but they allow also the development of complex data structures and the formulation of processing instructions. An XML schema uses a formal grammar to determine which XML elements are defined, which processing rules should be implemented in what manner and which meaning the individual elements have.

The confidentiality of XML documents may be ensured by "XML Encryption" [XMLENC], the integrity and authenticity by "XML Signature" [XMLDSIG]. In this respect, different forms of XML signatures are differentiated depending on whether the signatures are within or outside the XML document.

Application

For all data/documents and as character set and descriptive language for all of the archive data

---

\(^{24}\) Further definitions and recommendations regarding suitable document formats may be found at http://www.kbst.bund.de/saga.

\(^{25}\) For that reason, the Standards and Architectures for E-Government Applications [SAGA] in the version 3.0 of October 2006 therefore promote XML as the universal and primary standard for the exchange of data in and with the administration. Systems to be newly created should generally be able to exchange data using XML.

\(^{26}\) For more information, see http://www.w3c.org/XML/
objects that conform to this Technical Guideline: A valid XML-formatted archive data object shall fulfil the XML specification of the World Wide Web Consortium (W3C) [XML].

4.1.2 XML Schema (XSD)

XML schema is an XML markup language to define the structures of XML documents (XML instances). With the help of an XML schema, it is possible to formulate structures and limitations expressed in the form of rules or structure models that apply to a class of XML documents. XML schemata are usually used in order to document an applicable and valid XML vocabulary for the exchange of data within a business area or industry.

In this respect, the primary purpose of an XML schema is to validate XML documents. By validating an XML document against a schema, it is possible to ensure that the contents of an XML document correspond to the agreed rules.

An archive data object in the sense of this Technical Guideline shall be verified against a valid and authorised XML schema before being filed in electronic long-term storage. The authorisation shall ensure that the creator (owner) of the schema is uniquely identified and the schema cannot be manipulated without this being noticed.

Application

For all archive data objects that conform to this Technical Guideline: A schema that conforms to this Technical Guideline shall implement the normative recommendations of the XML Schema Working Group of the W3C [XSD].

4.2 Content data (object data)

For permanent and legally compliant storage of electronic primary information (content data), it shall be ensured that the negotiability and (machine) readability of the stored electronic information may be guaranteed by suitable measures at least for the duration of the statutory retention periods.

In order to ensure long-term negotiability, only standardised and data formats stable in the long term with an open description should therefore be used. The use of standard formats not only reduces the dependence on hardware and software environments, but also the necessity for future format transformations which require a significant amount of effort, in particular when electronic signatures are used.

4.2.1 Documents (records)

The Organisation Concept of the public administration and the Standards and Architectures for E-Government Applications (SAGA), like the Model Requirements for the

27 A basic set of requirements that is formally described in the XML Schema Requirements Note of the W3C leads to a number of use cases for schemata and the design guidelines on which they are based (see http://www.w3.org/TR/NOTE-xml-schema-req).

28 For more information, see also Drucksache 16/5927 des Deutschen Bundestages (Printed paper 16/5927 of the German Bundestag) of 4 July 2007. According to this, standards are to be considered open "if they make the exchange between different platforms and applications possible and are adequately documented. The interfaces must be open and the technical specifications executable. In doing so, the design of the conditions of use should correspond to the specifications of the international standardisation organisation."


30 See http://www.kbst.bund.de/saga.

Management of Electronic Records - Moreq\textsuperscript{10} promoted by the European Commission recommend that only a few standardised data formats be used for the long-term storage of electronic records. These include (at the time this Technical Guideline was published):

4.2.1.1 Text (ASCII)

ASCII (American Standard Code for Information Interchange) stands for a character set and for a text format. An ASCII text describes a document that only consists of the ASCII character set, and thus does not include any layout information and is thus particularly well suited for simple text information and meta data. The ASCII code was specified in 1972 by the International Organization for Standardization as ISO 646\textsuperscript{33} and, from today's perspective, offers the best prerequisites for ongoing negotiability. Because ASCII by its definition does not incorporate any character sets, correct display is not always guaranteed.

The format originally consists of 7 bits with which 128 (character) combinations may be displayed, i.e. a character set based on the Latin alphabet as used in the modern English language. 8-bit ASCII character sets were defined in order to map some special characters, such as the umlauts in the German language.

A further development of the ASCII encoding is the so-called Unicode Standard\textsuperscript{34}. Depending on the coding table, it is based on 8 (UTF-8), 16 (UTF-16) or 32 (UTF-32) bits per character code.

**Application**

For simple, unformatted texts (text files): Documents (text data) that are written in Latin letters only should use the character set defined in ISO standard ISO 646:1991 (ASCII) \[\text{ANSIX3.4}\].

Documents (text data) that are not written in Latin letters only should use the current version of the Unicode Standard [UNICODE].

Unicode is the functional equivalent to the ISO 10646-1:2000 standard. Texts that conform to Unicode are generally coded in UTF-8 or UTF-16.

4.2.1.2 PDF/A

PDF/A-1\textsuperscript{35} is a standard established as an ISO standard on the basis of [PDF 1.4] for the long-term archiving of electronic documents.

Published as ISO 19005-1:2005\textsuperscript{36}, PDF/A-1 defines the requirements for a PDF that conforms to the standard and regulates the use of PDF with regard to the long-term stability and reproducibility.

The standard specifies two levels of conformance:

- **PDF/A-1a - Level A conformance**: both unique visual reproducibility and the consistent use of Unicode and content structuring of the document.
- **PDF/A-1b - Level B conformance**: unique visual reproducibility.

PDF/A is set up as a series of standards. Additional parts were developed in the competent ISO committee (ISO TC171 SC2 WG5). In 2011, a second part of the standard, PDF/A-2, that is based on the ISO version of the PDF format (ISO 32000\textsuperscript{37}) and takes into account technological innovations that have arisen in the meantime, such as JPEG 2000, was published. The third part of the standard, PDF/A-3\textsuperscript{38}, based on [PDF 1.7], was published in

---


\textsuperscript{33} See http://www.iso.org/.

\textsuperscript{34} This specification is available at http://www.unicode.org/.

\textsuperscript{35} See http://www.adobe.com/enterprise/pdfs/pdfarchiving.pdf.

\textsuperscript{36} See http://www.iso.org/.

\textsuperscript{37} See http://www.iso.org/.

\textsuperscript{38} See http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm%3Fcsnumber%3D57229.
2012 and extends the standard by the embedding of any file type (XML, CAD etc.) for archiving purposes.

Because there has already been another series of standards based on PDF since 2001, PDF/X\textsuperscript{39}, for the exchange of masters in the graphic industry it was ensured during the development of PDF/A that a valid PDF/A file may also be valid PDF/X file.

PDF/A, like PDF version 1.4, supports different security mechanisms, in particular also the embedding of electronic signatures in the internationally recognised PKCS#7 format [PKCS#7].

PDF/A is also recommended by the Guidelines for Electronic Records Management [Moreq2] promoted by the European Commission as the format for archiving purposes.

**Application**

For all (primarily character-oriented) static documents.

**Recommendation**

PDF files should fulfil the ISO 19005-1 (PDF/A-1) standard. When using PDF/A-1, the following limitations apply depending on the level of conformance:

- **Converting**: If the file is converted from another file format, e.g. Microsoft Word, into PDF/A, then a program shall be used that explicitly enables the creation of PDF/A-compliant files. As an alternative, it may be output in version 1.4 of PDF, for example with the help of the program developed by Adobe, Acrobat Distiller in version 5.0 or higher, or the freely available AFPL GhostScript in version 7.0 or higher. In this case, too, however, final conversion into PDF/A should be carried out with a corresponding converter, because PDF 1.4 allows structures and contents that are not allowed in PDF/A.

- **Tagged PDF**: Tagged PDF stipulates a defined internal structure for the text content of a PDF file in order to retrieve its contents with suitable tools so that they may be processed further for other purposes. Possible applications are the transmission of contents in file formats such as XML, HTML or RTF. Files shall be created in such a manner that they correspond to Tagged PDF as described in [PDF 1.4]. The level of conformance A (PDF/A-1a) may be reached in this manner.

- **Linearised PDF**: Linearised PDF was developed in order to be able to show the pages of a PDF file as quickly as possible within network environments, such as the World Wide Web. This profile does not impair the ability to archive a file in the long term; using it when generating PDF files is allowed.

- **Meta data**: A file that may be archived should be as self-explanatory as possible, which may be guaranteed, among other means, by storing meta data on the basis of the eXtensible Metadata Platform (XMP)\textsuperscript{40} developed by Adobe in the file itself. The storage of meta data shall be carried out in the XMP format. This also applies to the original PDF meta data such as author, title, creation tool etc. if they are used in the document.

- **Encryption and other security settings**: Read-access to a file may not be subject to any limitations whatsoever; in particular, no passwords may be used in order to prevent the application of certain functions to the file. It is thereby guaranteed, for example, that the file may be transmitted into other formats for the purposes of archiving if this is necessary to maintain its contents. Limitations on printing shall be avoided. Security options that go further, such as limitations on the extraction of passages of text, or copying options shall not be used.

- **Authenticity and integrity**: The use of cryptographic procedures (electronic

\textsuperscript{39} PDF/X is standardised in the ISO standards 15929 and 15930; ISO 15929 defines the PDF/X approach in general; ISO 15930 defines concrete parts of the standard; for more information, see http://www.iso.org/.

\textsuperscript{40} See XMP Adobe eXtensible Metadata Platform; for more information, see http://www.adobe.com/products/xmp.
signatures and time stamps) in order to prove the authenticity and integrity of a PDF file is recommended when not expressly required by legal regulations. In this respect, though, only those signature application components that have been evaluated and certified by the BSI as secure signature application components in the sense of the German Signature Act (SigG) should be used. If signatures are embedded in the document, the container structure introduced with PDF 1.3 based on PKCS#7 applies.

- **Text:** Each font used in the text of the file shall be embedded in the file; the file shall contain the graphic descriptions of all of the fonts used therein in addition to the actual text. For optimisation, only the characters currently used in the text for description must be embedded (subsetting). This measure ensures that PDF display programs can always display the file in the form intended by the author without having to make use of replacement fonts that could change the visual appearance of the file. The standard fonts provided by Adobe shall also always be embedded if they are used in the document. In general, only publicly available fonts that are in no way subject to limitations by the owners of the rights should be used. If possible, all of the characters contained in the file should be available in machine-readable encoded form and not as digitalised pictures of characters so that they are available for search functions. For text sections which include characters that go beyond the ISO-8859-1 character set for US-American English and Western European languages [ISO-Latin-1]41, the [UNICODE] character set should be used. Thus, an author should code texts either in ISO-Latin-1 or in UNICODE, which is realised concretely by choosing a corresponding fonts that have consistent coding, such as Arial Unicode MS.

- **Graphics:** If the file contains graphic depictions (figures), then they shall either be included with their source colour profiles (e.g. CalRGB) or all graphics are assigned a uniform target colour profile or OutputIntent (e.g. standard red–green–blue [sRGB]42). This ensures a device-independent colour space specification. The so-called transparency key introduced as a visual effect in version 1.4 of PDF that makes it possible to generate graphics which appear to overlap or be translucent shall not be used in general. Corresponding pictures with transparency are to be converted (flattening). Alternative views (e.g. so-called downsampling for quick previews in the web) are not allowed for either colour or black and white (greyscale) pictures. The same applies to levels. The picture formats allowed in a PDF/A-1 file are, among others, TIFF/G4, JBIG, JBIG2 and JPEG. The JPEG2000 format is not allowed in PDF/A-1.

- **Integration:** All contents needed for the presentation of the document shall be included in the file itself so that it is not necessary to load data streams from external sources. The file shall not include any object that would require an external application program for its display. A view that requires a rendering for specific output devices is not allowed.

- **Audio/video:** For the reasons listed above, a file shall not contain audio or video data streams.

- **Links:** Internal links that refer to jump labels within the file, such as headlines, are allowed. External links that refer to jump labels outside the file, such as hyperlinks to resources in the Internet, should be designed if possible in such a manner that all symbolic addresses of these links, such as file paths, uniform resource locators (URL) or persistent identifiers, are contained in the text of the file and can be displayed on

---


42 IEC standard: IEC 61966-2-1 – Ed. 1.0 – Bilingual Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management - Default RGB colour space - sRGB, see [http://webstore.iec.ch/webstore/webstore.nsf/artnum/025408](http://webstore.iec.ch/webstore/webstore.nsf/artnum/025408).
the screen or printed on paper without any additional measures. For example: Instead of "Website of the ArchiSafe Project" with a link behind the text, it should be written: "Website of the ArchiSafe Project (http://www.archisafe.de)".

- **Comments**: Using comments is allowed as long as they do not contain audio or video data streams or any other file attachments.
- **Executable actions**: The file *shall not* contain executable command strings, such as scripts; in particular, JavaScript *shall not* be included in the fields of forms nor in any other place. Form fields themselves are allowed.

### 4.2.1.3 ODF

The Open Document Format (ODF) was standardised by OASIS\(^{43}\) as an XML-based document format for texts, spreadsheet calculations, presentations and other office documents. The contents of the documents and information about their layout are separated from each other and they can thus be processed independently. It may be used to exchange complex documents that are intended for further processing.

In November 2006, OpenDocument v1.0 was published as a standard under the name ISO/IEC 26300:2006\(^{44}\). The OpenDocument format is supported by platform-independent, licence-cost-free and Open Office package from OpenOffice.org\(^{45}\).

**Application**

For all (primarily) character-oriented documents.

### 4.2.1.4 TIFF\(^{46}\)

The "Tagged Image File Format" (TIFF) makes it possible to save graphic information without any loss of information and has been standardised pursuant to ISO 12639 for media-independent image processing. The coding of the format makes it possible to store multiple views (e.g. thumbnails) or versions of a graphic as well as text information as metadata in a file.

The use of TIFF is especially recommended if the graphic information of a document is of great significance for its expressiveness. TIFF is supported by all common graphic and presentation applications.

In order to achieve maximum interoperability, only characteristics from the "Baseline TIFF"\(^{47}\) should be used. TIFF may be used if the ability of the format to depict multi-page documents is needed. TIFF is especially well-suited for scanned text documents (grey-scale or B/W graphics).

**Application**

For figures (non-coded information or raster images).

TIFF files *should* fulfil the Basis-TIFF specification in the version 6.0 [TIFF6]. Furthermore, the following extensions *may* be used:

- CCITT bi-level encoding and
- LZW compression.

---


\(^{44}\) See [http://www.iso.org/](http://www.iso.org/).


\(^{47}\) Those characteristics of TIFF files that all TIFF-capable programs should support are summarised under "Baseline TIFF". For example, only the two compression procedures "Huffman" and "Packbits" belong to "Baseline TIFF", whereas "LZW", "JPEG", "ZIP" and "CCITT" are optional extensions that are not implemented in every TIFF-capable program.
4.2.1.5 JPEG

The JPEG format\(^{48}\) (Joint Photographic Experts Group) stands for a compression procedure and a graphic format and is one of the most common graphic formats used in the Internet. The JPEG standard was published as ISO/IEC 10918-1 in 1992. Because the definition of the structure of JPEG files allows many freedoms, a standard that organises the exchange of JPEG-compressed image data was defined with the "JPEG File Interchange Format" (JFIF). JFIF is based on the JPEG standard and is platform-independent. The use of the JPEG format as an alternative to TIFF may be recommended if a trade-off between picture quality and file size has to be found.

**Application**

For figures (non-coded information or raster images).

4.2.1.6 PNG

PNG\(^{49}\) (Portable Network Graphics Format) was developed by the later "PNG Development Group" as an alternative to the GIF format and is especially suitable for applications in the Internet because of the possibility of loss-free compression and incremental display of the graphics. The PNG specification is open and was elevated to an international standard in 2003 as ISO/IEC 15948\(^{50}\). Most image editing programs support the PNG format as standard. So-called calibration data blocks allow the calibration of the display so that, for example, the picture may be printed exactly as the author sees it on the screen.

**Application**

For figures (non-coded information or raster images).

4.2.2 Multimedia formats

"Multimedia" is characterised by the possibility to use diverse digital means for displaying information (video, audio, picture and text) at the same time.

In this context, a multimedia format is a self-explanatory file format that contains the content data and (in the case of container formats, see below) their structure and interrelations. Three multimedia formats have to be distinguished:

- An audio format describes the structure of an audio file,
- A video format describes the structure of a video file,
- A container format may contain multiple data elements (called "streams") with different formats. Initially, this is not limited to multimedia formats. The container also regulates the interplay and order of the individual data streams. Thus, for example, a container may contain multiple audio streams on a video stream and it is possible to embed subtitles (e.g. in the form of graphics). The audio streams may also be encoded with various codecs\(^{51}\) that are used to encode or decode analogue signals.

---


\(^{49}\) See \url{http://www.w3.org/TR/PNG/}.

\(^{50}\) See \url{http://www.iso.org/}.

\(^{51}\) Codec, a word coined for encoder / decoder; a program or procedure (format) with which multimedia data (audio or video signals) is digitally encoded for the transmission by digital services / networks and then decoded again when it is played. In most cases, the analogue signal is not digitalised without losses during the encoding, but rather there is a dynamic reduction of the analogue signal and a data compression of the digital signal which, according to the extent and the procedure, can make a significant difference in the quality of the reconversion of the digital data stream into analogue signals. The main goal of the dynamic reduction and compression is a reduction in the bandwidth needed for the transmission of the digital signal or a reduction in the amount of storage capacity needed for the saving. The compressed files are deposited in a special code format.
or digital audio or video data, and thereby map different sound qualities, for example. In the following sections, the most common representatives of the file format classes "audio" and "video" and "container" are described briefly. The common container formats are usually not perceived as container formats, but rather as audio or video formats. Thus, the container formats are also described in the following two sections. The recommended file formats are marked accordingly.

**NOTICE:** The multimedia formats that are common today and recommended in this document have not been analysed yet adequately for their suitability for long-term storage in a manner comparable to PDF. For the long-term availability of the archived data, it may thus be necessary to archive additional data, such as sound models or video drivers, along with the raw data.

**NOTICE:** Of course, only a small selection of recommended multimedia formats may be provided here. There are numerous other formats used in the scope of digitally saving audio and video information. In the context of long-term storage, however, these formats may currently only be given a qualified recommendation or none at all, either because they are not common enough, there are no open specifications or the legal situation (in particular the licence conditions) makes using them more difficult or does not recommend it. Therefore, this Technical Guideline is oriented towards the general Standards and Architectures for E-Government Applications of the Federal Government (SAGA V5.0) as far as it is possible and reasonable.

### 4.2.2.1 Audio formats

#### 4.2.2.1.1 Ogg Encapsulation Format

Ogg [RFC3533] is a container format for multimedia files. The development of the container format is directed by the Xiph.Org Foundation. Codecs are also made available. The best-known and most popular codec is the Vorbis audio codec. As an alternative, the loss-free FLAC audio codec may be used to ensure a better quality.

Ogg is an alternative to proprietary formats that is free from software patents and unlimited and, therefore, it is suitable for an electronic long-term storage due to the available detailed format specification in particular.

Ogg-Vorbis is recommended as an audio format in SAGA V 5.0 (Standards and Architectures for E-Government Applications).

Furthermore, the format has the advantage that it may also be used as a recommended video format (by using another codec), which contributes to the reduction in the overall number of data formats that are suitable for long-term storage.

#### 4.2.2.1.2 MP4 / MPEG-4 Part 14

MP4 is also a container format for multimedia files. The development of the container format was directed by the Moving Picture Experts Group and is standardised in ISO/IEC 14496-12 and -14 (MPEG-4 Part 12 and 14).[55](#)

As an audio format, MPEG-4 is an open, manufacturer-independent standard and is suitable for electronic long-term storage. The audio format is recommended in SAGA V 5.0.

Furthermore, MP4 has the advantage that it may also be used as a recommended video format (by using another codec), which contributes to the reduction in the overall number of data formats that are suitable for long-term storage.

---

52 See [https://xiph.org/flac/](https://xiph.org/flac/).


formats that are suitable for long-term storage.

4.2.2.1.3 Advanced Audio Coding (AAC)

Advanced Audio Coding (AAC) is a standardised lossy audio format. AAC is specified in ISO/IEC 13818-7\textsuperscript{56} and is treated as an improved successor to MP3, because the quality is usually higher with the same bit rate.

AAC is part of the MPEG-2 and MPEG-4 specifications and was developed by the Moving Pictures Experts Group. AAC itself is free of patents and licences, but codec manufacturers for AAC must pay licence fees.

SAGA 5.0 does not mention AAC. Regardless of this, its openness and expected future popularity mean that it may certainly be recommended for long-term storage.

4.2.2.1.4 EBU Broadcast Wave Format (BWF)

In certain situations, it can also make sense to use the Broadcast Wave Format (BWF) pursuant to [EBU-BWF] specified by the European Broadcasting Union (EBU) for the long-term storage of audio data.

4.2.2.2 Video formats

4.2.2.2.1 Ogg Encapsulation Format

As mentioned above, Ogg [RFC3533] is a container format for multimedia files. The Theora\textsuperscript{57} video codec, which was also developed by the Xiph.Org Foundation, is available as a suitable codec.

Ogg is an alternative to proprietary formats that is free from software patents and unlimited and is suitable for electronic long-term storage due to the available detailed format specification in particular.

Ogg-Theora is recommended as a video format in SAGA V 5.0 (Standards and Architectures for E-Government Applications).

4.2.2.2.2 MP4 / MPEG-4 Part 14

MP4 is also a container format for multimedia files. The development of the container format was directed by the Moving Picture Experts Group and is standardised in ISO/IEC 14496-12 and -14 (MPEG-4 Part 12 and 14)\textsuperscript{58}.

As an video format, MPEG-4 is an open, manufacturer-independent standard and is suitable for electronic long-term storage. The video format is recommended in SAGA V 5.0.

Furthermore, the MP4 has the advantage that it may also be used as a recommended audio format (by using another codec), which contributes to the reduction in the overall number of data formats that are suitable for long-term storage.

4.3 Base64 encoding

All (binary) primary information (content data) that was introduced in Chapter 3.4 should be saved within the dataObjectsSection of the XAIP structure of the archive data object (see Chapter 3.4). At the same time, of course, there is also the requirement here that the format and data be readable for a long time in the future, including by other systems and platforms.

\textsuperscript{56} See http://www.iso.org/.
\textsuperscript{57} See http://www.theora.org/.
\textsuperscript{58} See http://www.iso.org/.
Entering binary data into an XML structure without changes is technically possible, but this usually causes a number of problems and surely leads to a certain dependency on a software or a platform. For that reason, binary data must first be brought into a platform-independent form. For legal reasons, the encoding necessary on the contents of the binary data shall not make any changes; thus, this shall be a bijective mapping\textsuperscript{59}. The most common method is BASE64 encoding pursuant to [RFC4648]. It is primarily used in the MIME\textsuperscript{60} (Multipurpose Internet Mail Extensions) Internet standard and is thus mainly used for sending e-mail attachments. This is necessary to ensure smooth transport of any binary data, since the Simple Mail Transfer Protocol (SMTP)\textsuperscript{61} in its original version was only designed for sending 7-bit ASCII characters.

For BASE64 encoding, 3 bytes (24 bits) each of the origin file are mapped to 4 blocks each with 6 bits of the target file. In doing so, the order of the bits is maintained. 6 bits per block allow $2^6 = 64$ possible characters. That is where the name of the procedure comes from. For these 64 possible characters, the characters A–Z, a–z, 0–9, + and / were chosen. These characters are contained both in ASCII and in EBCDIC and are independent of a code page. Thus, data may also be exchanged between non-ASCII systems. Due to the mapping of 3 bytes to 4 characters, the space requirements rise by 33\%. However, this disadvantage is usually accepted.\textsuperscript{62}

The following procedure should be applied to the encapsulation of content data in XAIP document that conforms to this Technical Guideline:

- If the content data only consists of text (ASCII) (see Chapter 4.2.1.1) or XML, these data may be inserted into the XAIP data structure without re-coding.
- All other data formats shall be encoded in BASE64 before they are inserted into the XAIP data structure.
- For BASE64-encoded data, an entry for the corresponding MIME format of the content data should be in the meta data section that contains information that could be needed for an adequate representation of the content data.

\textsuperscript{59} A mapping $f: \mathcal{A} \rightarrow \mathcal{B}$ is called bijective (or clearly reversible) if different elements from an origin set $\mathcal{A}$ are mapped to different elements in an target set $\mathcal{B}$, and if, conversely, each element from $\mathcal{B}$ occurs as a depiction of exactly one element in the origin set $\mathcal{A}$.

\textsuperscript{60} See http://www.ietf.org/rfc/rfc2045.txt.

\textsuperscript{61} See http://www.ietf.org/rfc/rfc821.txt.

\textsuperscript{62} There are also other coding procedures. However, they are usually only used for special applications (e.g. in the post-script file format from Adobe or for the address encoding of IPv6 addresses) or are not used at all.
5. Cryptographic data formats

The following section describes a few cryptographic data formats that are recommended for trustworthy long-term storage at the time this Technical Guideline was published.

5.1 Signature formats

The signature formats common in practice are in particular the ASN.1-based signatures of the CMS family pursuant to [PKCS#7, RFC3852 or RFC5652, ETSI 101733, ETSI 103 173] and XML-based signatures pursuant to [XMLDSIG, ETSI 101903]. If the signature formats used in the environment of the trustworthy long-term storage can be influenced, these formats should be used, whereby the recommendations in Section 5.1.1 and 5.1.2 should be taken into consideration.

In general, though, the use of other signature formats (see [HK 06b]) may be necessary and reasonable depending on the application-specific requirements. In this case, the recommendations in Section 5.1.3 should be taken into consideration.

5.1.1 PKCS#7 / CMS / CAdES

The Cryptographic Message Syntax (CMS) signature format pursuant to [RFC3852] or [RFC5652] going back to [PKCS#7] is the most commonly used ASN.1-based signature format in practice. Because the data to be signed in this signature format are treated merely as binary objects – without consideration for an internal structure – any data can be signed, but the signatures cannot be embedded into the payload without further ado.

Building on the basic CMS structure, specific extensions are defined in [ETSI 101733] or [RFC5126] that take the special requirements for advanced electronic signatures pursuant to § 2 No. 2 German Signature Act [SigG] into account. For example, there are attributes for the countersigning of signatures (CounterSignature), the insertion of attribute certificates (SignatureAttributes), time stamps (ContentTimeStamp, SignatureTimeStampToken, ESCTimeStampToken, TimestampedCertsCRLs and ArchiveTimeStampToken), certificates (CertificateValues) and revocation information (RevocationValues). Furthermore, there is also an international standard profile [ISO14533-1] for the long-term storage of CAdES-based signatures, and definitions which aim at increasing interoperability may be found in the CAdES Baseline Profile [ETSI 103173].

The following is laid down in this Technical Guideline for the treatment of CMS- and CAdES-based signatures:

(A5.1-1) CMS-based signatures shall be verified prior to the generation of the initial archive time stamp. In doing so, all certificates used during the verification shall be stored in the SignedData.certificates element in accordance with [ISO14533-1] and [ETSI 103 173]. Accordingly, the revocation information used for the verification shall also be stored in the SignedData.crls element, whereby the certificate revocation lists are to be stored in the SignedData.crls.crl option and OCSP responses in the SignedData.crls.other option.

5.1.2 XML signatures / XAdES

In addition to the CMS-based signatures explained above, XML-based signatures pursuant to [XMLDSIG] are also increasingly being used in practice. The advantage of this signature format is that the specific characteristics of XML-based data are taken into account and thus,
for example, only explicitly defined parts of a document may be signed and the signatures themselves may be embedded into the payload data. Here, too, there are specific extensions for advanced electronic signatures pursuant to § 2 No. 2 German Signature Act [SigG] that are defined in [ETSI 101903] or [XAdES].

For example, there are signature properties for the countersigning of signatures (CounterSignature), the insertion of attribute certificates (SignerRole), time stamps (AllDataObjectsTimeStamp, IndividualDataObjectsTimeStamp, SignatureTimeStamps, SigAndRefsTimeStamps, RefsOnlyTimeStamps and ArchiveTimeStamps), certificates (CertificateValues) and revocation information (RevocationValues). Furthermore, there is also an international standard profile [ISO14533-2] for the long-term storage of XAdES-based signatures, and definitions which aim at increasing interoperability may be found in the XAdES Baseline Profile [ETSI 103171].

The following is laid down in this Technical Guideline for the treatment of XML- and XAdES-based signatures:

(A5.1-2) XML-based signatures shall be verified prior to the generation of the initial archive time stamp. In doing so, the certificates used for the signature verification, which are not already included in the ds:KeyInfo element, shall be inserted in the xades:CertificateValues element of the XAdES signature in accordance with [ISO14533-2] and [ETSI 103171]. Similarly, the revocation information used during the signature verification, which is not already included in the ds:KeyInfo element, shall be inserted into the xades:RevocationValues element of the XAdES signature.

5.1.3 Other signature formats

For the treatment of other signatures which are not standard-compliant PKCS#7/CMS/CAdES or XML/XAdES signatures, this Technical Guideline specifies the following recommendations:

(A5.1-3) These signatures should also be verified prior to the generation of the initial archive time stamp.

Unlike with the PKCS#7/CMS/CAdES or XML/XAdES signatures, the certificates and revocation information used as part of the signature verification should be stored in corresponding <credential> elements so that this information may be accessed if necessary. In doing so, certificates should be stored in the <certificateValues> element and revocation information in the <revocationValues> element within a <credential> element.

5.2 Certificate formats

Among the various standardised certificate formats, the public key and attributes certificates pursuant to [X.509] are of particular importance for trustworthy long-term storage. In the environment of trustworthy long-term storage, certificates are used in particular for the verification of (qualified) electronic signatures. In doing so, the recommendations in [Common-PKI] (Part 5 and 9) should be taken into consideration and corresponding verifications of the certificate status (see Section 5.3) should be carried out.

(A5.2-1) The certificate chains resulting from this up to a trustworthy root entity shall be inserted into the signature or the XAIP container as explained in Section 5.1.
5.3 Certificate validation formats

(A5.3-1) If the validity of the certificates is not otherwise ensured in the certificate paths generated during the signature verification, an explicit verification of the validity status of the certificates shall be carried out.

For this purpose, the Online Certificate Status Protocol (OCSP) pursuant to [RFC2560] or [RFC6960] should be used. As an alternative, the Server-Based Certificate Validation Protocol (SCVP) pursuant to [RFC5055] may be used if it is based on OCSP information from the issuers of the certificate.

If the mechanisms available for the verification of the certificate status for the certificates used may be influenced in the environment of trustworthy long-term storage, no certificate revocations lists pursuant to [RFC5280] should be used, because this cannot generally determine the validity of a (qualified) electronic signature without any doubt.

5.3.1 Online Certificate Status Protocol (OCSP / RFC 2560 / RFC 6960)

In the case of the Online Certificate Status Protocol (OCSP) pursuant to [RFC2560] or [RFC6960], current information about the certificate status is usually requested directly from the issuer of the certificate through a challenge-response protocol. Because the disclosure from the issuer occurs practically at the time of the signature verification, it may be ensured that the revocation information is up to date – unlike the case when, for example, certificate revocation lists are used. In environments with a high number of OCSP transactions, the recommendations pursuant to [RFC5019] may be observed.

(A5.3.1-1) As described in Section 5.1, OCSP-based status information shall be inserted into the corresponding signature or into the XAIP container.

5.3.2 Server-Based Certificate Validation Protocol (SCVP / RFC 5055)

In the case of the Server-Based Certificate Validation Protocol (SCVP) pursuant to [RFC5055], the tasks necessary in the scope of the validation of a certificate, which may be very complex under certain circumstances, may be outsourced to a specialised service. This makes sense, for example, when a complex signature verification must be carried out by a component with low computing power (e.g. mobile terminal). For these purposes, the SCVP client sends a request to the SCVP server in which, for example, the certificate to be verified is submitted or otherwise specified. After that, the SCVP server creates and verifies the complete certificate path; in doing so, it uses the primary revocation information in the form of OCSP responses or certificate revocation lists as a basis, which are typically provided by the issuer of the certificate.

When the SCVP server is requested, it should be made clear in the wantBack element (in the query element of the CVRequest) by means of the OID id-swb-pkc-revocation-info that the corresponding revocation information should be returned.

(A5.3.2-1) The revocation information returned in the CVResponse (in the replyObjects/replyWantBacks/revInfoWantBack/RevocationInfos) should be extracted and inserted into the corresponding signature or into the XAIP container as described in Section 5.1.
5.4 Time stamp

To request (qualified) time stamps, the Time Stamp Protocol (TSP) pursuant to [RFC3161] should be used in particular\(^{63}\).

(A5.4-1) In order to make it easy to verify the time stamps at a later point in time, only the signature of the time stamp service should be included in the SignedData container and after the generation of a time stamp and the expiry of the so-called "grace period" (see [ETSI 101733], Section 4.4.2), the verification of the time stamp shall be carried out automatically. In doing so, all certificates used during the verification shall be stored in the SignedData.certificates element in accordance with [ETSI 103 173]. Accordingly, the revocation information used for the verification shall also be stored in the SignedData.crls element, whereby the certificate revocation lists are to be stored in the SignedData.crls.crl option and OCSP responses in the SignedData.crls.other option.

5.5 Evidence Record (pursuant to RFC 4998 /RFC 6283)

Pursuant to the Evidence Record Syntax (ERS) Standard of the IETF [RFC4998] or [RFC6283]\(^{64}\), an Evidence Record is a unit of data with which the existence of stored data and documents at a defined point in time may be technically proven. It includes cryptographic Evidence Records with which the integrity and authenticity of electronically saved data and documents may be verified at all times. Technically, the ERS standard is based on the approach that cryptographic checksums (hash values) of the archive data objects (XAIP documents) are organised in a hash tree (pursuant to Merkle [MER 1980]) when stored in the archive system as cryptographically unique representatives of the data to be stored and that the roots of the hash tree are secured ("sealed") with a qualified time stamp containing a qualified electronic signature for proving the integrity (see also Annex [TR-ESOR-M.3]). This first initial time stamp is also referred to as initial archive time stamp pursuant to the ERS standard [RFC4998] or [RFC6283].

The source of trust for the archive time stamp and thus for legally compliant re-signing pursuant to § 17 German Signature Ordinance (SigV) is the qualified time stamp that contains a qualified electronic signature. Its data structure should fulfil the requirements of the "Time Stamp Protocol (TSP)" [RFC3161] and the "Cryptographic Message Syntax (CMS)" pursuant to [RFC3852] or [RFC5652] and [ISO14533-1].

In the case that re-signing or a time stamp renewal is necessary that is sufficient if only the digital signature procedure threatens to lose its suitability as a security measure, but the hash algorithm remains suitable, a new archive time stamp includes the hash value of the original time stamp in the hash tree that is to be generated with a new final qualified time stamp so that a secure and verifiable, chronological chain of evidence made of cryptographically linked archive time stamps arises. The Evidence Record resulting from this contains an additional ArchiveTimeStamp element in the ArchiveTimeStampChain element pursuant to [RFC4998] or [RFC6283] that has already existed beforehand.

If the suitability of the used hash algorithm as a security measure is (also) threatened, the hash tree shall be renewed. In doing so, the archive data object is hashed with a suitable algorithm and a new ArchiveTimeStampChain element with a corresponding ArchiveTimeStamp element is inserted into the ArchiveTimeStampSequence

\(^{63}\) Furthermore, in environments with a high number of necessary individual time stamps, the use of archive time stamps (ArchiveTimeStamp) that are based on hash trees and standardised in [RFC4998] or [RFC6283] or the design of the so-called "interval-qualified time stamps" explained in [HK 09] is recommended.

\(^{64}\) RFC 4998 shall, RFC 6283 may be supported.
element. Further information in this respect may also be found in [RFC4998] or [RFC6283].

The technical proof of the maintenance of the integrity and therefore, if necessary, the authenticity of the data stored in the electronic long-term storage, then occurs, along with the presentation of the actual archival data and the associated, valid certificates of existing electronic signatures, in particular with the proof of the integrity of the cryptographic representatives of the archive data objects, i.e. the hash values and archive time stamps.

For these purposes, the ERS standard specifies a so-called Evidence Record. This Evidence Record contains in particular a sequence of archive time stamps with which the integrity and authenticity of the archive data objects may be proven. An archive time stamp, in turn, contains all necessary data from the hash tree (reducedHashTree) needed for verifying that the archive data object belongs to the hash tree. The root of the hash tree is sealed with a qualified time stamp (see also Annex [TR-ESOR-M.3]).

(A5.5-1) The generation of an ERS pursuant to [RFC4998] shall be supported. The generation of an ERS pursuant to [RFC6283] may be supported. The structure and assignment of the Evidence Records may be found in Annex [TR-ESOR-ERS]. A rough overview is provided in the text below.

5.5.1 EvidenceRecord pursuant to RFC 4998

An EvidenceRecord pursuant to [RFC4998] is an ASN.1 structure defined as follows:

EvidenceRecord ::= SEQUENCE {
    version             INTEGER { v1(1) },
    digestAlgorithms    SEQUENCE OF AlgorithmIdentifier,
    cryptoInfos         [0] CryptoInfos OPTIONAL,
    encryptionInfo      [1] EncryptionInfo OPTIONAL,
    archiveTimeStampSequence ArchiveTimeStampSequence
}

CryptoInfos ::= SEQUENCE SIZE (1..MAX) OF Attribute

The version field (e.g. v1) describes the current version of the ERS standard.

The digestAlgorithms field contains a list of all hash algorithms with which hash values were created for the data saved during the retention period.

The optional cryptoInfos field makes it possible to transport data that is needed for the validation of the information contained in the archiveTimeStampSequence field.

The encryptionInfo field which is also optional includes necessary information on the correct handling of encrypted contents.

The archiveTimeStampSequence field contains a sequence of linked archive time stamp chains (ArchiveTimestampChain elements) which were generated for the data stored over the course of the retention period. An ArchiveTimestampChain element contains a sequence of at least one or several elements of the ArchiveTime stamp type that are sorted in ascending order according to the time of the time stamp contained therein.

ArchiveTimeStampSequence ::= SEQUENCE OF ArchiveTimestampChain
ArchiveTimestampChain ::= SEQUENCE OF ArchiveTime stamp

ArchiveTimestampChain and ArchiveTimeStampSequence are chronologically ordered based on the time of the final time stamp. Within an archive time stamp chain, all reduced hash trees are based on the same hash algorithm.

Pursuant to the IETF’s ERS standard, an archive time stamp is defined as follows:
ArchiveTimeStamp ::= SEQUENCE {
    digestAlgorithm             [0] AlgorithmIdentifier OPTIONAL,
    attributes                  [1] Attributes OPTIONAL,
    reducedHashtree             [2] SEQUENCE OF PartialHashtree OPTIONAL,
    timeStamp                   ContentInfo -- TimeStampToken pursuant to [RFC3161]
}
TimeStampToken ::= ContentInfo
-- [RFC3161]
-- contentType is id-signedData ([RFC3852])
-- content is SignedData ([RFC3852])
The digestAlgorithm field identifies the hash algorithm used. If the field is not
available, the ERS standard assumes that the hash algorithm of the time stamp was used for
the generation of the hash value.
  • The optional attributes field may contain additional information on the rules
    applied to re-signing or the application of the archive time stamp.
The reducedHashtree field contains all hash values that are needed for the mathematical
verification of the hash value nodes into which the original hash value of the archive data
object including the final qualified time stamp has been incorporated.
The timeStamp field includes a qualified time stamp that was generated using a qualified
electronic signature as cryptographic confirmation of the integrity of the data returned with
the Evidence Record.

5.5.2  <EvidenceRecord> pursuant to [RFC6283]
An <EvidenceRecord> element pursuant to [RFC6283] is of the EvidenceRecordType
which has the following structure:

<x:s:element name="EvidenceRecord" type="EvidenceRecordType" />
<x:s:complexType name="EvidenceRecordType">
  <x:s:sequence>
    <x:s:element name="EncryptionInformation" type="EncryptionInfo" minOccurs="0" />
    <x:s:element name="SupportingInformationList" type="SupportingInformationType" minOccurs="0" />
    <x:s:element name="ArchiveTimeStampSequence" type="ArchiveTimeStampSequenceType" />
  </x:s:sequence>
  <x:s:attribute name="Version" type="xs:decimal" use="required" fixed="1.0" />
</x:s:complexType>

<EncryptionInformation> [optional]
If necessary, the <EncryptionInfo> element may include information on how to
deal with encrypted data. Further details on the EncryptionInfoType may be found in
[RFC6283].

<SupportingInfoList> [optional]
The <SupportingInfoList> element may include further supporting
information, such as information on the suitability of cryptographic algorithms as
security measures pursuant to [RFC5698]. Further details on the
SupportingInformationType can be found in [RFC6283].

<ArchiveTimeStampSequence> [required]
Similarly to the ASN.1-based Evidence Record pursuant to [RFC4998], the
<ArchiveTimeStampSequence> element contains a sequence of
<ArchiveTimeStampChain> elements, that, in turn, includes a sequence of
<ArchiveTimeStamp> elements. Further details on the
ArchiveTimeStampSequenceType can be found in [RFC6283].
5.6 Recommendations for the implementation

Based on the requirements listed above, this section makes recommendations in which manner these requirements should be implemented concretely for the S.4 XML adapter – ArchiSafe-Module interface. The same procedure may be used for all other interfaces.

(A5.6-1) In order to be able to process any data format and to be able to link the cryptographic data and meta data with the payload data, the XAIP container defined in Chapter 6 or a derivative thereof should be used as the central data element in the protocol. For the protocol, this means in particular that all data is kept in a single data element and are logically connected to each other in this manner. The protocol is not responsible for the logical correctness of this data element after receipt.

(A5.6-2) To protect the integrity and the confidentiality during the transmission and to authenticate the requests and answers (responses), a "trusted channel", such as a TLS tunnel, should be established with certificate-based authentication on both sides prior to any communication between the client module and the ArchiSafe-Module. Neither requests nor answers (responses) shall be sent through insecure channels. Both the client and the server should ensure this.

(A5.6-3) The "trusted channel" shall ensure the integrity and confidentiality of the data transmitted in it with sufficiently strong cryptographic procedures pursuant to [TR 02102]. The archive system shall enforce this and shall not accept any weak procedure during the establishment of the tunnel.

(A5.6-4) The "trusted channel" shall be maintained at least for the duration of a transaction65. Request and answers (responses) to a transaction shall be transmitted through the same "trusted channel".

(A5.6-5) If a "trusted channel" is interrupted during a transaction for any reason, the client shall not expect any answer (response) of any kind from the archive system. In this case, the client shall establish a new "trusted channel" and determine the receipt of the request, the current status or the end of the transaction to the server by means of STATUS requests.

(A5.6-6) The "trusted channel" should be maintained as long as desired and used for any number of transactions (also parallel).

(A5.6-7) A protocol by means of which the technical confirmation of the receipt of a client request is realised among other things shall be chosen as the transmission protocol within the "trusted channel".

(A5.6-8) Recommendation for the protocol on the application layer is SOAP document/literal encoding66. The external interfaces of all archive system components will be published with WSDL; they may be based on an external XML schema.

(A5.6-9) Furthermore, it shall be taken into account that the archive modules may process several (many) transactions – also based on several client applications – at the same time.

---

65 Here, the term "transaction" includes the client request to a server and the resulting server response to the client.

66 Literal encoding uses an XML Schema to validate the SOAP data and offers a significantly better performance than RPC encoding especially for large payload data (in this respect, see also [FC 07], p. 76 et seq. or http://www-128.ibm.com/developerworks/webservices/library/wsoapenc).
6. Annex - XML schema definition

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xaip="http://www.bsi.bund.de/tr-esor/xaip/1.2"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xades="http://uri.etsi.org/01903/v1.3.2#"
  xmlns:ers="urn:ietf:params:xml:ns:ers"
  xmlns:vr="urn:oasis:names:tc:dss-x:1.0:profiles:verificationreport:schema#"
  xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
  xmlns:ec="http://www.bsi.bund.de/ecard/api/1.1"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  targetNamespace="http://www.bsi.bund.de/tr-esor/xaip/1.2"
  elementFormDefault="qualified" attributeFormDefault="unqualified">

  <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="http://ws.openecard.org/schema/xmldsig-core-schema.xsd" />
  <xs:import namespace="http://uri.etsi.org/01903/v1.3.2#"
    schemaLocation="http://ws.openecard.org/schema/XAdES-1-3-2.xsd" />
  <xs:import namespace="urn:ietf:params:xml:ns:ers"
    schemaLocation="http://ws.openecard.org/schema/xml-ers-rfc6283.xsd" />
  <xs:import namespace="urn:oasis:names:tc:dss-x:1.0:profiles:verificationreport:schema#"
    schemaLocation="http://ws.openecard.org/schema/oasis-dssx-1.0-profiles-verification-report-cs1.xsd" />
  <xs:import namespace="urn:oasis:names:tc:dss:1.0:core:schema"
    schemaLocation="http://ws.openecard.org/schema/oasis-dss-core-schema-v1.0-os.xsd" />
  <xs:import namespace="http://www.bsi.bund.de/ecard/api/1.1"
    schemaLocation="http://ws.openecard.org/schema/eCard.xsd" />
  <xs:import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
    schemaLocation="http://ws.openecard.org/schema/saml-schema-assertion-2.0.xsd" />

  <xs:element name="XAIP" type="xaip:XAIPType"/>

  <xs:complexType name="XAIPType">
    <xs:sequence>
      <xs:element name="packageHeader" type="xaip:packageHeaderType" />
      <xs:element name="metaDataSection" type="xaip:metaDataSectionType" maxOccurs="1" minOccurs="0" />
    </xs:sequence>

</xs:complexType>
```

<xs:element name="dataObjectsSection"
  type="xaip:dataObjectsSectionType" maxOccurs="1"
  minOccurs="0"/>
</xs:element>

<xs:element name="credentialsSection"
  type="xaip:credentialsSectionType" maxOccurs="1"
  minOccurs="0"/>
</xs:element>
</xs:sequence>
</xs:complexType>

<!------------------------------------------------------------------->
<!-- packageHeaderType -->
<!------------------------------------------------------------------->
<xs:complexType name="packageHeaderType">
  <xs:sequence>
    <xs:element name="AOID" type="xs:string"
      maxOccurs="1" minOccurs="0"/>
    <xs:element name="packageInfo" type="xs:string"
      minOccurs="0"/>
    <xs:element name="versionManifest"
      type="xaip:versionManifestType"
      maxOccurs="unbounded" minOccurs="1" />
    <xs:element ref="ds:CanonicalizationMethod"
      maxOccurs="1" minOccurs="0"/>
    <xs:element name="extension" type="xaip:extensionType"
      maxOccurs="1" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="packageID" type="xs:ID" use="required"/>
</xs:complexType>

<xs:complexType name="extensionType">
  <xs:sequence>
    <xs:any processContents="lax" maxOccurs="unbounded" />  
  </xs:sequence>
</xs:complexType>

<xs:complexType name="versionManifestType">
  <xs:sequence>
    <xs:element name="versionInfo" type="xs:string"
      maxOccurs="1" minOccurs="0"/>
    <xs:element name="preservationInfo"
      type="xaip:preservationInfoType" />
    <xs:element name="submissionInfo"
      type="xaip:submissionInfoType"
      maxOccurs="1" minOccurs="0"/>
    <xs:element name="packageInfoUnit"
      type="xaip:packageInfoUnitType"
      maxOccurs="unbounded" minOccurs="1"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="extension" type="xaip:extensionType"
  maxOccurs="1" minOccurs="0">
</xs:element>
</xs:sequence>
<xs:attribute name="VersionID" type="xs:ID" use="required"/>
</xs:complexType>

<xs:complexType name="preservationInfoType">
  <xs:sequence>
    <xs:element name="retentionPeriod" type="xs:date" />
    <xs:element name="status" type="xs:string"
      maxOccurs="1" minOccurs="0">
    </xs:element>
    <xs:element name="extension" type="xaip:extensionType"
      maxOccurs="1" minOccurs="0">
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="otherContentType">
  <xs:complexContent>
    <xs:extension base="xs:anyType">
      <xs:attribute name="Type" type="xs:anyURI" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="submissionInfoType">
  <xs:sequence>
    <xs:element name="clientID" type="saml:NameIDType">
    </xs:element>
    <xs:element name="submissionUnit" type="saml:NameIDType"
      maxOccurs="1" minOccurs="0">
    </xs:element>
    <xs:element name="submissionAuthor" type="saml:NameIDType"
      maxOccurs="1" minOccurs="0">
    </xs:element>
    <xs:element name="submissionTime" type="xs:dateTime"
      maxOccurs="1" minOccurs="0">
    </xs:element>
    <xs:element name="extension" type="xaip:extensionType"
      maxOccurs="1" minOccurs="0">
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="packageInfoUnitType">
  <xs:sequence>
    <xs:element name="unitType" type="xs:string"
      minOccurs="0">
    </xs:element>
    <xs:element name="textInfo" type="xs:string"
      minOccurs="0">
    </xs:element>
    <xs:element name="protectedObjectPointer" type="xs:IDREF"
      minOccurs="1" maxOccurs="unbounded">
    </xs:element>
    <xs:element name="unprotectedObjectPointer" type="xs:IDREF"
      minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:element type="xaip:packageInfoUnitType">
  <minOccurs="0" maxOccurs="unbounded"/>
</xs:element>
<xs:element type="xaip:extensionType">
  <maxOccurs="1" minOccurs="0"/>
</xs:element>
</xs:element>
<xs:element name="packageUnitID" type="xs:ID" use="required"/>
</xs:complexType>

<!---------------------------------------------------------------------------------------------------------------------------------->
<!- MetadataSection -->
<!---------------------------------------------------------------------------------------------------------------------------------->
<xs:complexType name="metaDataSectionType">
  <xs:sequence>
    <xs:element ref="xaip:metaDataObject">
      <maxOccurs="unbounded" minOccurs="1"/>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:element name="metaDataObject" type="xaip:metaDataObjectType"/>
<xs:complexType name="metaDataObjectType">
  <xs:complexContent>
    <xs:extension base="xs:anyType">
      <xs:attribute name="metaDataID" type="xs:ID" use="required"/>
      <xs:attribute name="dataObjectID" type="xs:IDREF" use="required" />
      <xs:attribute name="category" type="xs:string" />
      <xs:attribute name="classification" type="xs:string" />
      <xs:attribute name="type" type="xs:string" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="dataObjectCheckSum" type="xaip:checkSumType"/>

<!---------------------------------------------------------------------------------------------------------------------------------->
<!- DataObjectsSection -->
<!---------------------------------------------------------------------------------------------------------------------------------->
<xs:complexType name="dataObjectsSectionType">
  <xs:sequence>
    <xs:element ref="xaip:dataObject">
      <maxOccurs="unbounded" minOccurs="1"/>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:element name="dataObject" type="xaip:dataObjectType"/>
<xs:complexType name="dataObjectType">
  <xs:complexContent>
</xs:complexContent>
</xs:complexType>
<xs:choice>
    <xs:element name="binaryData">
        <xs:complexType>
            <xs:simpleContent>
                <xs:extension base="xs:base64Binary"/>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="xmlData" type="dss:AnyType"/>
</xs:choice>

<xs:element name="checkSum" type="xaip:checkSumType" minOccurs="0"/>
<xs:element name="transformInfo" type="xaip:tranformInfoType" minOccurs="0"/>
<xs:attribute name="dataObjectID" type="xs:ID" use="required"/>

<xs:complexType name="checkSumType">
    <xs:sequence>
        <xs:element name="checkSumAlgorithm" type="xs:anyURI"/>
        <xs:element name="checkSum" type="xs:hexBinary"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="tranformInfoType">
    <xs:sequence>
        <xs:element name="transformObject" type="xaip:transformObjectType" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="transformObjectType">
    <xs:sequence>
        <xs:element name="transformAlgorithm" type="xs:anyURI"/>
        <xs:element name="Parameters" type="xs:anyType" maxOccurs="1" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="transformObjectID" type="xs:ID" use="required"/>
    <xs:attribute name="order" type="xs:string"/>
</xs:complexType>

<!---------------------------------------------------------------------------------->
<!-- CredentialsSection -->
<!---------------------------------------------------------------------------------->
<xs:complexType name="credentialsSectionType">
<xs:sequence>
    <xs:element ref="xaip:credential"
        maxOccurs="unbounded" minOccurs="1">
    </xs:element>
</xs:sequence>
</xs:complexType>

<xs:element name="credential" type="xaip:credentialType"/>

<xs:complexType name="credentialType">
    <xs:choice>
        <xs:element ref="dss:SignatureObject"/>
        <xs:element name="certificateValues" type="xades:CertificateValuesType"/>
        <xs:element name="revocationValues" type="xades:RevocationValuesType"/>
        <xs:element ref="xaip:evidenceRecord"/>
        <xs:element ref="vr:VerificationReport"/>
        <xs:element name="other" type="xaip:extensionType"/>
    </xs:choice>
    <xs:attribute name="relatedObjects" type="xs:IDREFS"/>
    <xs:attribute name="credentialID" type="xs:ID" use="required"/>
</xs:complexType>

<xs:element name="evidenceRecord" type="xaip:EvidenceRecordType"/>

<xs:complexType name="EvidenceRecordType">
    <xs:complexContent>
        <xs:extension base="ec:EvidenceRecordType">
            <xs:attribute name="AOID" type="xs:string"/>
            <xs:attribute name="VersionID" type="xs:string"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:element name="DXAIP" type="xaip:DXAIPType"/>

<xs:complexType name="DXAIPType">
    <xs:complexContent>
        <xs:extension base="xaip:XAIPType">
            <xs:sequence type="xaip:updateSectionType"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="updateSectionType">
    <xs:sequence>
        <xs:element name="prevVersion" type="xs:string"/>
        <xs:element name="placeHolder" type="xaip:placeHolderType" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="placeHolderType">
    <xs:attribute name="objectID" type="xs:ID" use="required"/>
</xs:complexType>
</xs:schema>