Federal Office
for Information Security

BSI Technical Guideline 03125 Preservation of Evidence of Cryptographically Signed Documents

Annex TR-ESOR-AeSV: AIP-eIDAS-Signature-Validator-Tool (AeSV)

Designation	ArchiveInformationPackage(AIP)-eIDAS-Signature-Validator
Abbreviation	BSI TR-ESOR-AIP-eIDAS-SigV
Version	1.3 (on base of the eIDAS-Regulation and the ETSI Preservation Standards with a new certification scheme)
Date	14.11.2022

Document history

Version	*Date*	*Editor*	*Description*
1	11.07.2022	procilon Group	Adoption of template
2	29.07.2022	procilon Group	Added additional information
3	25.08.2022	procilon Group	Added information for ASiC containers
4	06.10.2022	procilon Group	Processing of received comments
5	25.10.2022	procilon Group	Extended version, corrections and processing of received comments
6	27.10.2022	procilon Group	Extending description of the test data supplied with the validator
7	28.10.2022	procilon Group	Extending description of the test data, minor corrections in the text
8	14.11.2022	procilon Group	Spell check, introduction of Annex D
9	22.11.2022	procilon Group	Minor improvements, added a hint to Annex F of TR-ESOR for dealing with XAdES signatures in (L)XAIP

Table 1: Document History

Federal Office for Information Security
P.O. Box 20 03 63
53133 Bonn
Tel:+49 229 99 9582-0
E-Mail: tresor@bsi.bund.de
Internet: https://www.bsi.bund.de

Table of Contents

1 Introduction

1.1 Overview

This document contains an overview of a validation component AIP-eIDAS-Signature-Validator-Tool (AeSV) [1] for (L)XAIP and ASiC-AIP containers. These containers may contain preservation objects, which are listed in notice 2 below this paragraph. The AIP-validator contains functionality for the structural and syntactical validation of AIP objects. It offers in addition the possibility of using an external digital signature verification service for the validation of electronic signatures, seals, and timestamps.

NOTICE 1

In this TR-ESOR- Version, the word "AIP" means in all TR-ESOR-Annexes :

a) *the XML-based archival information package "XAIP" pursuant to* ***[TR-ESOR-F], clause 3.1*** *as well as*
b) *the logical XAIP "LXAIP" pursuant to* ***[TR-ESOR-F], clause 3.2*** *as well as*
c) *the "ASiC-AIP" pursuant to* ***[TR-ESOR-F], clause 3.3*** *on base of* ***[ETSI EN 319162-1].***

In general, this TR-ESOR Version differentiates in detail between "XAIP", "LXAIP" and "ASiC-AIP". In this document, in some places XAIP means XAIP or LXAIP or ASiC-AIP.

To improve readability, at several places in this document, the insertion of "LXAIP" and "ASiC-AIP" is renounced, and the footnote ***"****"XAIP" shall be supported, "LXAIP" and "ASiC-AIP, if implemented" may be supported****"*** *is added to each "XAIP".*

In this document, the word "(L)XAIP" represents "XAIP" or "LXAIP" pursuant to ***[TR-ESOR-F]*** *and the word "DXAIP" represents "DXAIP" or "DLXAIP"" pursuant to* ***[TR-ESOR-F].***

NOTICE 2

In this TR-ESOR- Version, ***BIN*** *is restricted to the following preservation object formats:*

- *ASiC-ERS (in TR-ESOR v1.3 called ASiC-AIP) pursuant to* ***[ETSI TS 119 512]*** *Annex A.3.1 and A.3.1.3 (http://uri.etsi.org/ades/ASiC/type/ASiC-ERS) and pursuant to* ***[TR-ESOR-F]*** *Clause 3.3;*
- *CAdES pursuant to* ***[ETSI TS 119 512]*** *Annex A.1.1 (http://uri.etsi.org/ades/CAdES). If there is no MIME type filled, then the default application/cms is used;*
- *XAdES pursuant to* ***[ETSI TS 119 512]*** *Annex A.1.2 (http://uri.etsi.org/ades/XAdES). If there is no MIME type filled, then the default application/xml is used*
- *PAdES pursuant to* ***[ETSI TS 119 512]*** *Annex A.1.3 (http://uri.etsi.org/ades/PAdES). If there is no MIME Type filled, then the default application/pdf is used;*
- *ASiC-E pursuant to* ***[ETSI TS 119 512]*** *Annex A.1.4 (http://uri.etsi.org/ades/ASiC/type/ASiC-E). If there is no MIME type filled, then the default application/vnd.etsi.asic-e+zip is used;*
- *ASiC-S pursuant to* ***[ETSI EN 319 162]*** *(http://uri.etsi.org/ades/ASiC/type/ASiC-S). If there is no MIME type filled, then the default application/vnd.etsi.asic-s+zip is used;*

This version of TR-ESOR supports DigestList pursuant to ***[ETSI TS 119 51****2] Annex A.1.6 (http://uri.etsi.org/19512/format/DigestList). This format is not supported in AIP-eIDAS-Signature-Validator-Tool (AeSV).*

Attention 1.

In TR-ESOR V1.3, actually, ASiC-AIP is announced but it is still not released and does not lead to a certification.

NOTICE 3

*In the following text the term "****digital signature****" covers "advanced electronic signatures" pursuant to* ***[eIDAS-VO]****, Article 3(11), "qualified electronic signatures" pursuant to* ***[eIDAS-VO]****, Article 3(12), "advanced*

[1] Further named with the short name "AIP-Validator".

*electronic seals" pursuant to **[eIDAS-VO]**, Article 3(26) and "qualified electronic seals" pursuant to **[eIDAS-VO]**, Article 3(27). Insofar, the term "digital signed document" covers documents signed by advanced electronic signatures or seals as well as documents signed by qualified electronic signatures or seals.*

*In this TR the term "**cryptographic signed documents**" covers not only qualified signed documents pursuant to **[eIDAS-VO]**, Article 3(12) or qualified sealed documents pursuant to **[eIDAS-VO]**, Article 3(27) or qualified time-stamped documents pursuant to **[eIDAS-VO]**, Article 3(34) (within the meaning of the **eIDAS** regulation), but also documents with advanced electronic signatures pursuant to **[eIDAS-VO]**, Article 3(11) or with advanced electronic seals pursuant to **[eIDAS-VO]**, Article 3(26) or with electronic time-stamps pursuant to **[eIDAS-VO]**, Article 3(33), as they are often used in the internal communication of public authorities. The documents with simple signatures or seals based on other (e.g. non-cryptographic) technologies are not meant here.*

1.2 Scope of the document

The document covers the following aspects:

- Installation and configuration
- Parameterization and usage
- Integration of signature verification components
- Major design decisions
- Known limitations

This documentation applies to the code located on https://github.com/de-bund-bsi-tr-esor/tr-esor-AIP-eIDAS-SigValidator marked with tag tresor-1.3.

At the time of publication of this document, support for TR-ESOR 1.3 compliant validation is also available on the main branch.

2 Introduction

The Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik, BSI) is the responsible German authority for information security.

The BSI has published the technical guideline for the long-term preservation of evidence of cryptographically signed documents (TR-03125/TR-ESOR). A key aspect of this technical guideline is the preservation of electronic documents in structures named AIP-containers[2]. AIP-containers provide the abilities of storing documents, meta-data and credentials containing the supplemental evidence data (signatures, seals, timestamps, certificates, certificate revocation lists, OCSP responses, signature- or seal- or timestamp verification information, etc.) and evidence records. The technical guideline TR-03125 introduces adapters, which are responsible for the creation of AIP-containers. Since the containers need to be interoperable, there is a need for tools that can verify the structural and syntactical correctness as well as electronic signatures, seals, and timestamps, which may be part of the container.

The AIP-validator provides the required verification capabilities:

- Syntactical analysis: verification of well-formedness of AIP objects
- Structural analysis: verification of required and optional elements in the AIP object
- Signature and timestamp validation: Validation of electronic signatures, seals, and timestamps by using a verification service conformant to the signature verification functionality defined by the technical guideline TR-03112
- Creation of verification protocols conformant to TR-ESOR 03125, Annex TR-ESOR-VR integrating signature validation reports conforming to OASIS DSS v1.0 profile for comprehensive multi-signature verification reports version 1.0

The main building blocks of the AIP-validator are the following specifications:

- TR-03125, Anlage TR-ESOR-F: Formate Version 1.3 [TR-ESOR-F]
- TR-03125, Anlage TR-ESOR-VR: Verification Reports for Selected Data Structures, Version 1.3 [TR-ESOR-VR]

The validator does not depend on any archiving solution or any other third-party system with exception of the signature verification service. Figure 1 shows the general design overview. The following interfaces and components are part of the AIP-validator:

Table 2 Design Elements of the AIP-validator

Component	*Purpose*
CLI	Interface for accessing the functionality from command line
SOAP	Interface for accessing the functionality using SOAP
AIP-Validator	The AIP-validating component comprising modules for syntax validation, signature identification, signature verification and protocol generation.
Dispatcher	The dispatcher module is responsible for configuration evaluation as well as configuring other validator modules. It manages the calls to the modules with validation functionality.
SyntaxValidator	A module responsible for syntax validation. Checks the AIP-container presented to the validator for syntactical correctness and provides validation results to the Dispatcher.

[2] Please consider notice 1 in section 1.1

Component	*Purpose*
SignatureFinder	A module responsible for identification of electronic signatures and timestamps in the AIP-container. Provides information about electronic signatures, seals and timestamps to the SignatureVerifier[3].
SignatureVerifier	This component is responsible for calling the external Signature Verification Service with the signatures, seals and timestamps identified during the AIP inspection.
ProtocolAssembler	The ProtocolAssembler is responsible for the generation of the protocol stating the results of AIP inspection and signature validation. NOTICE 1: If the signature verification service is not configured or used, the protocol will contain the results of the structural verification of the AIP-container. Digital signatures will be reported but with status indetermined. Moreover, the hint is given that the signature verification has been omitted. The overall result of the AIP-container verification will contain the status urn:oasis:names:tc:dss:1.0:resultmajor:InsufficientInformation.
Signature Verification Service	The external Service for verification of cryptographically signed objects (electronic signatures, seals, timestamps, and evidence records) that are part of the AIP-container. The component provides the interface named 'Signature Verification' that is compliant to the specification of 'VerifyRequest' in TR-03112. All signature verification reports including those for electronic timestamps embedded into the validation report are generated by the external signature validation service. The AIP-validators capabilities are limited to the identification of signatures and timestamps and requesting the validation by the external verification component. See Notice 1 in the section of the Protocol Assembler.

The source code of the AIP-validator is located on GitHub under *de-bund-tr-esor/tr-esor-AIP-eIDAS-SigValidator*.

[3] The SignatureFinder searches for specified signature objects in the AIP container. No distinction is made between signature, timestamp and seal.

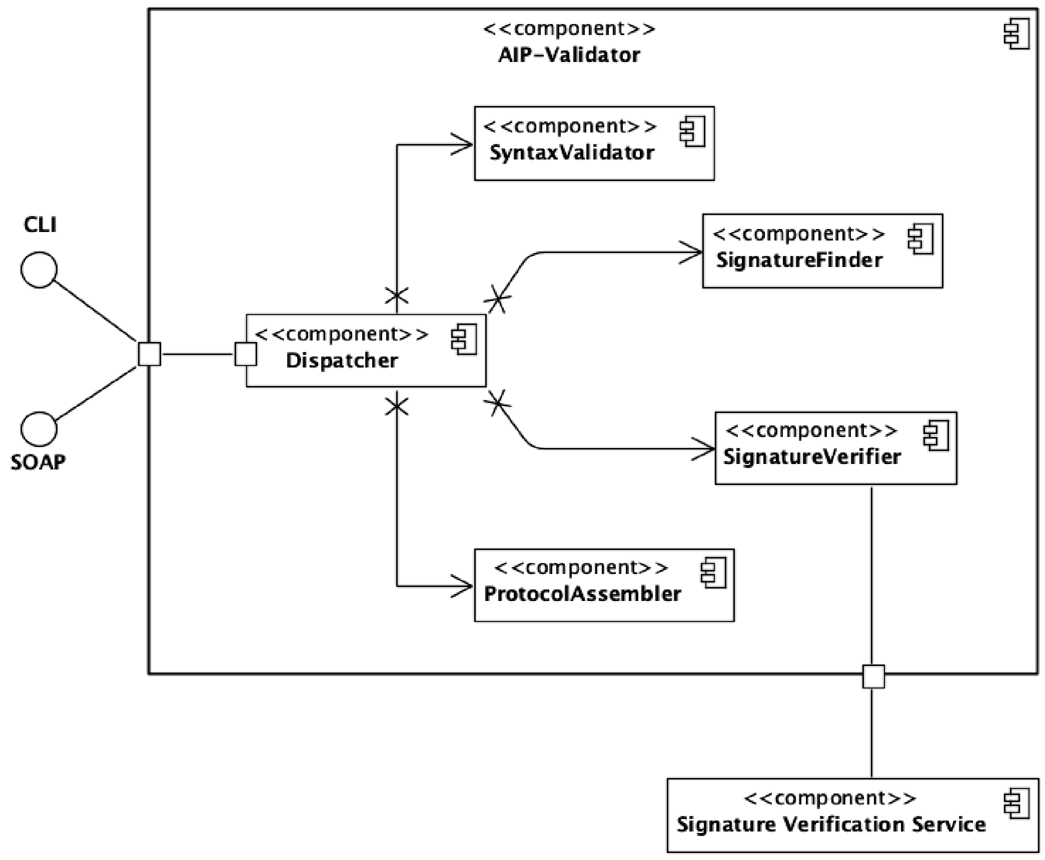

Figure 1General Design Overview

3 Legal and other information

The AIP-eIDAS-Signature-Validator-Tool (AeSV) itself as well as this documentation are provided under the Apache License Version 2.0, which is enclosed to the product:

4 Installation and Configuration

4.1 Prerequisites

General information

The following documentation assumes that the system used has a current patch status and that applications and libraries are kept up to date. It must be ensured that these requirements are met to ensure the functionality of the components to be installed.

Java 11

First of all, a current Java SDK version must be installed. To build and run the XAIP validator, at least Java 11 from version 11.0.10 is required[4]. The currently installed version can be obtained on the command line using the command **java** --version.

Git-Client

If the code is to be retrieved directly from the repository, the GIT command line tools and a GIT client - for example GitHub Client or Source Tree or another client of your choice - are required. A current GitHub account may be mandatory.

You can find further information at this point: https://github.com/git-guides/install-git.

A list of graphical git clients can be found here: https://git-scm.com/downloads/guis.

Maven

To build the AIP-validator, Maven is required. Maven and its documentation can be downloaded here: http://maven.apache.org/index.html.

4.2 Repository access

Depending on the selected access method, the repository can be cloned. The following call shows an example with stored SSH keys. Alternatively, the code archive can be downloaded as a ZIP file. In this case, it is not required to access the repository via a git client.

```
git clone git@github.com:de-bund-bsi-tr-esor/tr-esor-AIP-eIDAS-SigValidator.git
```

More information about cloning git repositories can be found here: https://github.com/git-guides/git-clone.

[4] Nevertheless the use of the latest Java 11 patch level is recommended.

4.3 Maven Build

Building the project requires the following steps:

- Open a terminal and change the current directory to the project root directory
- Call the Maven build command **mvn clean** package

The following lines demonstrate the procedure when the project root is named xaipvalidator[5].

```
cd xaipvalidator
```

```
mvn clean package
```

The readme file of the XAIPValidator project on GitHub contains a complete example of how to build the tool including the JavaDoc documentation under https://github.com/de-bund-bsi-tr-esor/tr-esor-AIP-eIDAS-SigValidator/blob/master/README.md.

The results of the build procedure will be located `xaip-validator-cli/target/`.

4.4 Creating JavaDoc API documentation

The Javadoc API documentation can be built calling:

```
mvn javadoc:aggregate -Ddoclint=none
```

on the root project. The javadoc will be available under tr-esor-AIP-eIDAS-SigValidator/target/site/apidocs/index.html which should be opened using an internet browser.

[5] The standard root directory of the project is identical to name of the project on GitHub (tr-esor-AIP-eIDAS-SigValidator).

4.5 CLI Options

Description: The cli version of the AIPValidator is being used for validation of AIP's via the command line.

Usage: `java -jar aip-validator.jar [OPTION [ARG]*]*`

-M=`<moduleConfig>`

Passing a single module configuration property to the validator. The property and requirements for any module configuration properties are defined by the implementing module.

Example:

```
-Mverifier.wsdlUrl=http://localhost:8080/eCard?wsdl
```

This example is using the property config verifier.wsdlUrl which is a required config property of the DefaultVerifierModule.

Instead of passing multiple module properties as an argument, a config file which contains the module properties can be passed instead. For more information see

`–c, --config <file>`

-c, --config `<file>`

Passing a configuration file in form of a property file to the validator. This file can contain any module configuration property. The configuration file is being passed to every module so they can scan the content and retrieve any configuration properties they define.

The handling is just the same as passing all configuration properties via a separate command line argument.

This argument can also be used together with single command line module property arguments.

Example:

```
-c config.properties
```

Content of `config.properties:`

```
validator.schemaDir=/tmp/xaip/definition

verifier.wsdlUrl=http://localhost:8080/eCard?wsdl
```

-i, --in, --input `<file>`

Passing a <file> as a source for the xaip validation. Omitting this argument will take the standard <inputStream> for the validation.

Example:

```
-i /tmp/sample.xaip
```

-o, --out, --output <file>

Defining a definition for the validation result. Omitting this argument will write the result to the standard <OutputStream> instead.

Example:

```
-o /tmp/report.xml
```

-v, --verify

This flag enables the signature verification which is being executed by the [SignatureVerifierModule]. Omitting this flag will only execute the syntax validation.

-d, --debug

Flag to enable debug output for a better analysis of the validator behaviour. This output can contain stack traces or other kinds of errors even when everything works fine.

-l, --log <file>

Since this tool is not only creating a report but also log output this argument can be used to separate the log output of the validator into a dedicated document.

Example:

```
-l validator.log
```

-h, --help

Displays information for using the validator.

Important Notes

- The location of the schema files must be set by using the parameter -Mvalidator.schemaDir is required when using the command line version of the AIP-validator with the default syntax validation module. Otherwise, the validator will not start.
- The argument of parameter -o must specify a directory. Currently the validator cannot write a plain file located in the same directory.
- The URL of the signature verification service must be specified using the parameter -v unless you are using a custom verification module. The signature verification service must be compliant to the TR-03112 / OASIS interface definition (using VerifyRequest).

4.6 Server Options

`-M=``<moduleConfig>`

Passing a single module configuration property to the validator. The property and requirements for any module configuration properties are defined by the implementing module.

Example:

```
-Mverifier.wsdlUrl=http://localhost:8080/eCard?wsdl
```

This example is using the property config `verifier.wsdlUrl` which is a required config property of the `DefaultVerifierModule`. Instead of passing multiple module properties as an argument, a config file which contains the module properties can be passed instead.

For more information see `-c, --config <file>`

`-c, --config` `<file>`

Passing a configuration file in form of a property file to the validator. This file can contain any module configuration property. The configuration file is being passed to every module so they can scan the content and retrieve any configuration properties they define.

The handling is just the same as passing all configuration properties via a separate command line argument. This argument can also be used together with single command line module property arguments.

Example:

```
-c config.properties
```

Content of `config.properties`:

```
validator.schemaDir=/tmp/xaip/definition
verifier.wsdlUrl=http://localhost:8080/eCard?wsdl
```

`-p, --port` `<port>`

Port the server should be published to, 8080 by default

`-P, --protocol` `<protocol>`

Protocol to be used, `http` by default

`-H, --host` `<hostname>`

Hostname the server is published to

`--path` `<path>`

Custom path the service should be used, `/xaip-validate` by default

`-d, --debug`

Flag to enable debug output for a better analysis of the validator behavior. This output may contain stack traces or other types of error messages even when everything works fine.

`-l, --log` `<file>`

Since this tool is not only creating a report but also log output this argument can be used to separate the log output of the validator into a dedicated document

Example: -l validator.log

`-h, --help`

Displays information for using the validator.

4.7 Available parameters of the validator modules

Without any additional configuration, the AIP-validator is using the default module implementations. Every module implementation can specify their own configuration which can be passed from outside using the param -M or –config. They can also be specified as a requirement to use the module.

All configuration options labelled with an asterisk (*) are mandatory configuration properties which must be provided as a minimum requirement to use the module.

Table 3: Default Syntax Validator Module

Parameter name	*Example*	*Description*
*validator.schemaDir	/tmp/xaip-schema	Path to the schema directory containing the xaip schema files

Table 4: Default Verifier Module

Parameter name	*Example*	*Description*
*validator.wsdlUrl	https://host:port/VerificationService/eCard?wsdl	URL to the WSDL of the VerificationService which should be used

5 Usage

5.1 Command Line Interface (CLI) usage

```
java

    -jar xaip-validator-cli/target/xaip-validator-cli.jar

    -i ~/example.xaip

    -Mvalidator.schemaDir=<schema-directory>
```

5.2 SOAP-Server mode usage

5.2.1 MacOS and Linux

```
java

    -cp            "xaip-validator-soap/target/xaip-validator-soap-1.1.0-
    1.jar:target/dependency/*"
    de.bund.bsi.tresor.xaip.validator.soap.Server

    -Mvalidator.schemaDir=<schema-director>

    -Mverifier.wsdlUrl=<verification-service-url>
```

Example of the verification service URL:

```
"https://host:port/VerificationService/eCard?wsdl"
```

5.2.2 Windows

```
java

    -cp            "xaip-validator-soap/target/xaip-validator-soap-1.1.0-
    1.jar;target/dependency/*"
    de.bund.bsi.tresor.xaip.validator.soap.Server

    -Mvalidator.schemaDir=<schema-directory>

    -Mverifier.wsdlUrl=<verification-service-url>
```

Example of the verification service URL:

```
"https://host:port/VerificationService/eCard?wsdl"
```

6 Integration of Verification Components

By default, the AIP-validator utilizes a component for the verification of digital signatures, seals, timestamps and evidence records that is compliant to the eCard-Framework (TR-03112) by implementing the web service method VerifyRequest. This web service method is part of the web service definition file ‘eCard.wsdl’, which is part of the web service and schema definitions provided together with the eCard-Framework.

```
<wsdl:operation name="VerifyRequest">
<soap:operation soapAction="http://www.bsi.bund.de/ecard/api/1.1#VerifyRequest" />
        <wsdl:input>
                <soap:body use="literal" />
                        </wsdl:input>
                        <wsdl:output>
                <soap:body use="literal" />
        </wsdl:output>
</wsdl:operation>
```

Implementations compliant to the specification of the eCard-Framework [TR-03112] can be utilized for signature verification by setting the service url as a configuration parameter of the AIP-validator.

6.1 Recommendation

For configuration of the signature verification webservice to be used; the parameter `verifier.wsdlUrl` must be set. For adjustment of the utilized signature verification implementation the use of an eCard-Framework compliant verification service is recommended.

6.2 Extending the implementation

In case that the use of an eCard-Framework compliant service is not possible; the module implementation must be extended. This section explains a possible approach.

A custom signature verification module should be implemented for supporting a custom signature verification service. The signature verification module must implement the interface `SignatureVerifier` from package `de.bund.bsi.tresor.xaip.validator.api.boundary`.

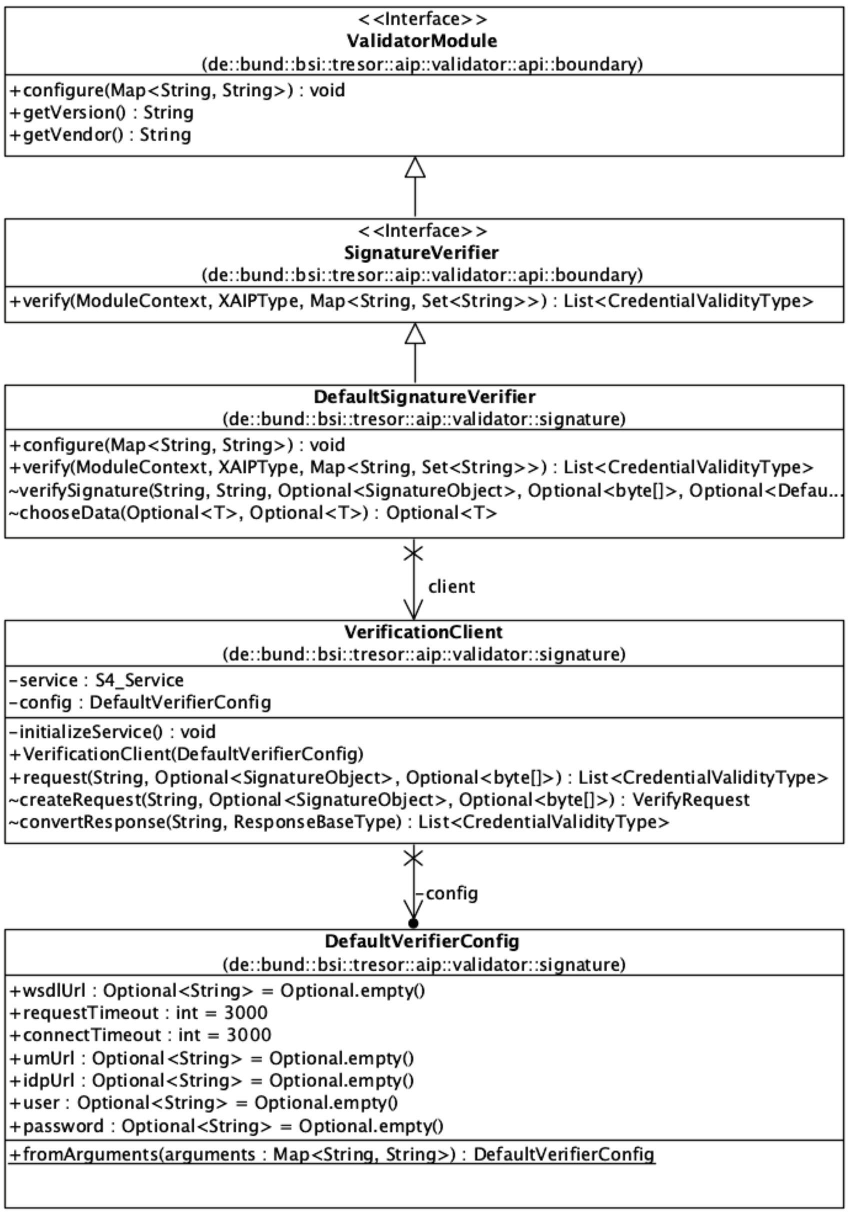
<<Interface>>
ValidatorModule
(de::bund::bsi::tresor::aip::validator::api::boundary)
+configure(Map<String, String>) : void
+getVersion() : String
+getVendor() : String
<<Interface>>
SignatureVerifier
(de::bund::bsi::tresor::aip::validator::api::boundary)
+verify(ModuleContext, XAIPType, Map<String, Set<String>>) : List<CredentialValidityType>
DefaultSignatureVerifier
(de::bund::bsi::tresor::aip::validator::signature)
+configure(Map<String, String>) : void
+verify(ModuleContext, XAIPType, Map<String, Set<String>>) : List<CredentialValidityType>
~verifySignature(String, String, Optional<SignatureObject>, Optional<byte[]>, Optional<Defau...
~chooseData(Optional<T>, Optional<T>) : Optional<T>
client
VerificationClient
(de::bund::bsi::tresor::aip::validator::signature)
-service : S4_Service
-config : DefaultVerifierConfig
-initializeService() : void
+VerificationClient(DefaultVerifierConfig)
+request(String, Optional<SignatureObject>, Optional<byte[]>) : List<CredentialValidityType>
~createRequest(String, Optional<SignatureObject>, Optional<byte[]>) : VerifyRequest
~convertResponse(String, ResponseBaseType) : List<CredentialValidityType>
-config
DefaultVerifierConfig
(de::bund::bsi::tresor::aip::validator::signature)
+wsdlUrl : Optional<String> = Optional.empty()
+requestTimeout : int = 3000
+connectTimeout : int = 3000
+umUrl : Optional<String> = Optional.empty()
+idpUrl : Optional<String> = Optional.empty()
+user : Optional<String> = Optional.empty()
+password : Optional<String> = Optional.empty()
+fromArguments(arguments : Map<String, String>) : DefaultVerifierConfig

Figure 2 Class hierarchy and dependencies of DefaultSignatureVerifier

6.2.1 Required Methods

The following methods need to be implemented by a signature verification module.

```
public String getVersion()
```

Provides a printable string with the version of the module, e.g., 1.0.0.

```
public String getVendor()
```

Provides a printable string with the name of the module vendor, e.g., “BSI”.

```
default public void configure(
      Map<String, String> properties )
```

Provides a function that receives configuration parameters and performs internal configuration tasks.

```
public List<CredentialValidityType> verify(
      ModuleContext context,
      XAIPType xaip,
      Map<String, Set<String>> credIdsByDataId );
```

This method is called automatically by the implementation of the AIP-validator.

`ModuleContext context`

The `ModuleContext` instance is an object that can be used to transfer additional information that needs to be exchanged between different modules.

`XAIPType xaip`

The xaip-object is an object representation of the XAIP-container processed by the AIP-validator.

`Map<String, Set<String>> credIdsByDataId`

This map contains the detached signatures and timestamps found by the AIP-validator during XAIP-container analysis.

Figure 2 shows a dependency to class `VerificationClient`. This client is responsible for sending the verification request to the signature verification web service. A possible strategy for the integration of a different signature validation component is the modification of this implementation.

7 Logging

The AIP-validator implements the following behavior:

- If parameters -l and -o are not set, the resulting validation report is printed to stdout and log messages are printed to stderr.
- Parameter -o may be used to redirect the generated validation report to the file specified as an argument
- Parameter -l may be used to redirect the log messages into a file specified as an argument
- Parameter -d or –debug may be used to activate verbose logging

8 Limitations

The following limitations apply:

- The AIP-validator has a strong dependency to the external signature validation service. The validation of electronic signatures, seals and timestamps depends on this external service. The signature verification report in the AIP-validation report is generated by the external signature verification service. Not all variants of *AdES-compliant signatures may be verified completely by the verification service. Please consider notice 1 in Table 2.
- Parallel XML-signatures are not supported for signature verification by the default signature verification service. In case of parallel XML-signatures a different signature verification service must be utilized.
- Extensions are not evaluated due to their dependency to specific profiles.
- TransformationInfos are identified by the validator but not validated

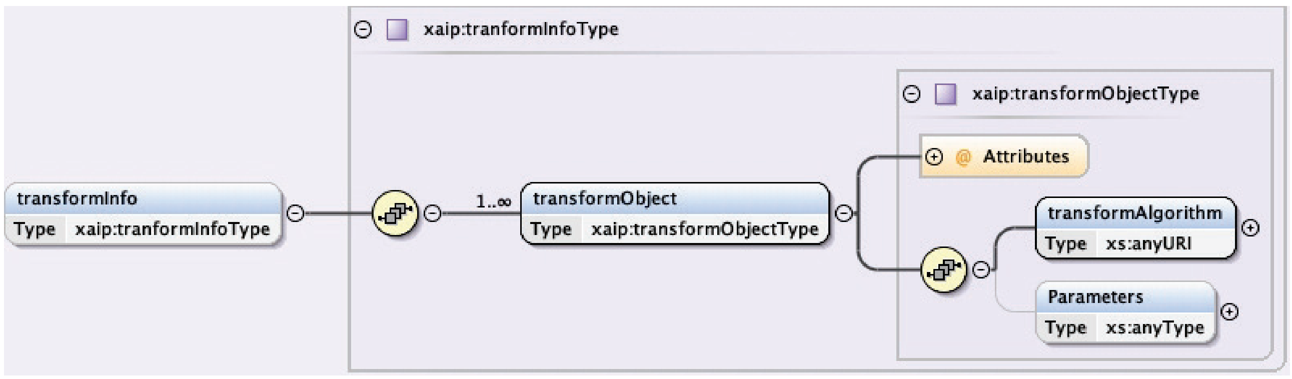

Figure 3 xaip:transformInfoType

The following fragement provides a sample of a TransformationInformation.

```
<dataObjectsSection>
      <dataObject dataObjectID="obj-42bd184c-7ebf-4877-9b81-79220d7457ff">
         <binaryData MimeType="application/octet-stream">Zm9v</binaryData>
         <transformInfo>
            <transformObject transformObjectID="sampleOid" order="sampleOrder">
               <transformAlgorithm>http://sample.org/sample</transformAlgorithm>
            </transformObject>
         </transformInfo>
      </dataObject>
   </dataObjectsSection>
```

Figure 4 xaip:transformInfo example

Be aware of the following issues:

- [XVAL-1] When using the parameter -o to specify the argument must be a file which is not in the current directory

9 Annex A: Special handling for XML signatures

The AIP-validator implements a special treatment for XML-signatures to circumvent canonicalization and transformation issues. Figure 5 shows the schema of the `dataObject` element, Figure 6 shows the structure of the encoding of the signature object in an XAIP[6]-container.

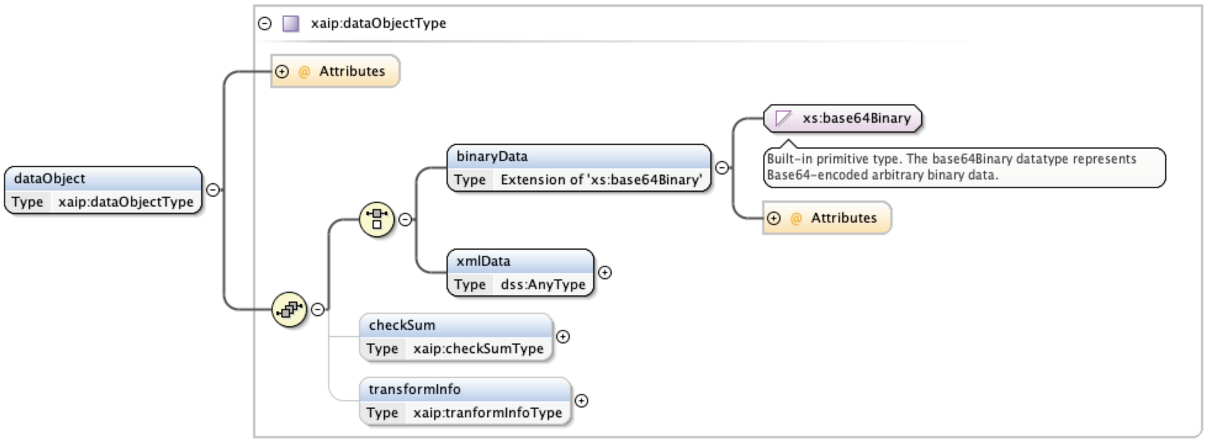

Figure 5 dataObject schema

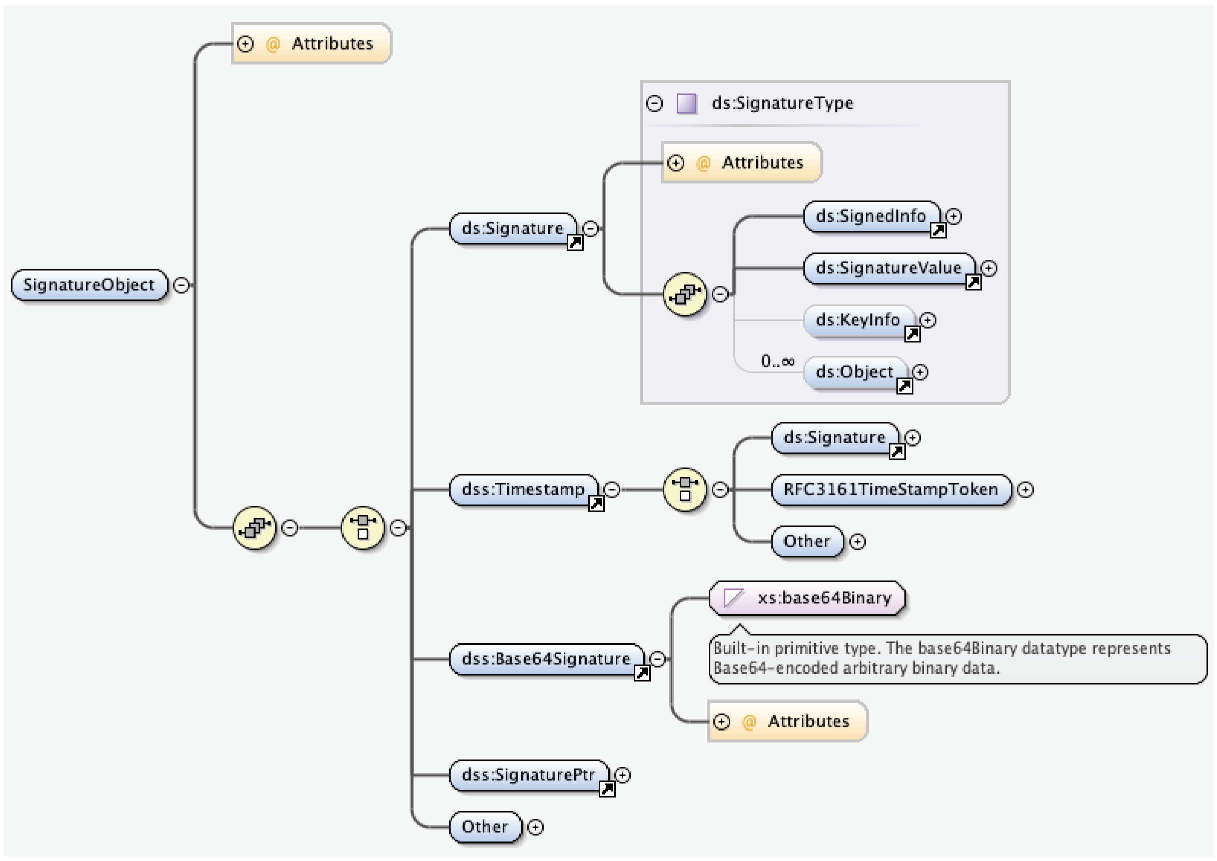

Figure 6 SignatureObject schema

[6] Please consider notice 1 in section 1.1. In this case XAIP also covers LXAIP.

NOTICE: Annex F of TR-ESOR contains hints for dealing with XAdES signatures and associated XML documents. It is recommended to store an XML document in the dataObjects section as `xaip:binaryData`. Any XML signatures should be stored in the `dss:Base64Signature` element in the `dss:SignatureObject`.

The AIP-validator implements the following strategy:

Enveloped and Enveloping XML-signature in xaip-element dataObject->binaryData

The signature may be encoded using base64-encoding in the `dataObject->binaryData` element of the XAIP[6]-container.

1. The AIP-validator decodes the data object.
2. If the result of step 1) appears to be an XML-document, the validator evaluates the XML-document for the appearance of an XML-signature object.
3. If an XML-signature object is detected, the base64-encoded document will be submitted to the signature verification service in base64-encoding using the field `dss:Base64Data` in the `VerifyRequest`

Enveloped and Enveloping XML-signature in xaip-element dataObject->xmlData

The signature may be part of a `dataObject->xmlData` element of the XAIP-container.

1. When detecting an element of type `dataObject->xmlData`, the validator evaluates the xmlData for the appearance of an enveloped or enveloping signature.
2. In case of a detected signature, the XML data will be submitted to the verification service in base64-encoding using the field `dss:Base64Data` in the `VerifyRequest`.

XML-signature in element dss:SignatureObject

1. A detached XML-signature may appear in the `dss:Base64Signature` element as part of the SignatureObject. In this case, the signature is submitted to the signature verification service in Base64 encoding.
2. A detached XML-signature may appear in element `ds:Signature` as part of the SignatureObject. In this case, the validator will extract the signature from the XAIP-object and submit the signature in Base64 encoding simulating a `dss:Base64Signature` element.
3. An enveloped or enveloping signature may appear in the `dss:Base64Signature` element as part of the SignatureObject. In this case, the signature is submitted to the signature verification service in Base64 encoding (as part of a `dss:Base64Signature` element).

Important Notice: If XML-coded data is embedded as a payload, existing canonicalization's in the AIP container do not apply to this data. The signature object can also contain a canonicalization rule, which is only to be applied to the `SignedProperties` element. The XML payload is passed to the signature verification component as it was embedded in the AIP container. If the digital signature cannot be validated successfully (especially with errors in the cryptographic validation), the encoding of the XML payload may be the cause therefore.

10 ASIC-S and ASIC-E

Basically, ASIC signatures are ZIP-compressed containers. ASiC-* signatures may be part of the following elements in the XAIP-container:

- `dataObject:binaryData` with no `dss:SignaturePtr`
- `dataObject:binaryData` with a `dss:SignaturePtr`
- `metaDataObject:binaryMetaData` with no `dss:SignaturePtr`
- `metaDataObject:binaryMetaData` with a `SignaturePtr`
- `dss:SignatureObject` with `dss:Base64Signature`
- as part of an LXAIP-container (LXAIP containing a file reference to an external ASiC-* signature)

The AIP-validator verifies the ASiC-* signatures, if one of the combinations listed above appears in AIP-container. AsiC-* signature containers may contain not only one cryptographic signature but also multiple signatures, timestamps, seals, and evidence records.

11 Annex B: Supported Archival Information Package Types and Archive Data Object Types

11.1 Archive Data Object Container Formats

The Archive Data Object Container Type “XAIP” pursuant to [TR-ESOR-F], clause 3.1 is supported and in addition, the Archive Data Object Container Type “LXAIP” pursuant to [TR-ESOR-F], clause 3.2.

Moreover, the AIP-Validator supports the syntax validation of Archive Containers of Type “ASiC-AIP” pursuant to [TR-ESOR-F], clause 3.3.

The validator supports both variants of ASIC-AIP:

- ASiC-AIP with data object embedded into the XAIP (see [TR-ESOR-F], clause 3.3.2) and
- ASiC-AIP with LXAIP-objects and referenced objects (see [TR-ESOR-F], clause 3.3.1).

Test data for supported AIPs is documented in Annex C.

11.2 Archive Data Object Types

Furthermore, the Preservation Product may also support the Archive Data Object Types:

- CAdES pursuant to [ETSI TS 119 512] Annex A.1.1 (http://uri.etsi.org/ades/CadES). If there is no MIME type filled, then the default application/cms is used;
- XAdES pursuant to [ETSI TS 119 512] Annex A.1.2 (http://uri.etsi.org/ades/XadES). If there is no MIME type filled, then the default application/xml is used;
- PAdES pursuant to [ETSI TS 119 512] Annex A.1.3 (http://uri.etsi.org/ades/PadES). If there is no MIME Type filled, then the default application/pdf is used;
- ASiC-E pursuant to [ETSI TS 119 512] Annex A.1.4 (http://uri.etsi.org/ades/AsiC-E);
- ASiC-S pursuant to [ETSI EN 319 162] (http://uri.etsi.org/ades/ASiC/type/ASiC-S).

Test data for supported archive data object types is documented in Annex C.

12 Annex C

12.1 Test data specification

The following table provides an overview about the test data supplied with the AIP-validator. The test data is published on the GitHub project. This overview provides basic information about structure and content of the test data. The review of the published test data is recommended for further information.

The test data provided with version 1.1.0 of the AIP-validator is compliant to the schemas provided with TR-ESOR 1.3 and may not be compatible with TR-ESOR 1.2.x compliant implementations.

The test material provided includes various signature encodings. These encoding are explained in Table 5 and used inTable 6.

Table 5 Signature encodings used in the test material

.Short	*Description*	*Identifier*
CMS detached	An ASN.1 encoded signature object conformant to CMS encoding but not compliant to CAdES. The signature object is stored separately from the signed data object.	A
CMS attached	An ASN.1 encoded signature object conformant to CMS encoding but not compliant to CAdES. The signed data and the signature object are stored together in the same file object. The signature container is an enveloping signature.	B
CAdES detached	An ASN.1 encoded signature object conforming to the CAdES standard. The signature object is stored separately from the signed data object.	C
CAdES attached	An ASN.1 encoded signature object conforming to the CAdES standard. The signed data and the signature object are stored together in the same file object. This format is also known as an enveloping signature. The signature container is an enveloping signature.	D
XAdES detached	An XML signature conforming to the XAdES standard. The signature object is stored separately from the signed data object.	E
XAdES enveloping	An XML signature conforming to the XAdES standard. The signed data and the signature object are stored together in the same file object. The signature is an envelope around the signed data. This format is also known as an enveloping signature.	F
XAdES enveloped	An XML signature conforming to the XAdES standard. The signed data and the signature object are stored together in the same file object. The signature is an enveloped by the signed data. This format is also known as an enveloped signature.	G
PDF	A signature embedded into a PDF document. The signature may be compliant to the PAdES standard containing either the ETSI.CAdES.detached entry in the signature subfilter or be non PAdES compliant PDF signature using adbe.pkcs7.detached signature subfilter.	H
ASiC-S	A ZIP compressed signature container (ASiC-Simple) with one signature. The ASiC-S encoding identifier is combined with the CAdES or XAdES identifiers.	I

.Short	*Description*	*Identifier*
ASiC-E	A ZIP compressed signature container (ASiC-Extended) with one ore more signature and/or seals and/or timestamps and/or evidence records. The ASiC-E encoding identifier is combined with the CAdES or XAdES identifiers.	J
RFC 3161 Timestamp	A timestamp compliant to RFC 3161	TS
Evidence Record	An evidence record compliant to RFC 4998	ER

The digital certificates used in the test data are either

- verifiable against a trusted service list (TSL) or a
- root certificate known to the signature validating component or
- unknown to the signature validating component (e. g. the root certificate is a self-generated test root certificate etc.)

The result generated by the signature validation service may therefore provide one of the following validation results:

- **valid:** all checks were successful including the cryptographic validation and the check of the certificate validity using OCSP or CRLs
- **invalid:** the signature is invalid, mostly due to failures in the cryptographic validation
- **indetermined:** the signature is cryptographically ok but the validity of the certificate cannot be determined, mostly due to missing OCSP responses or CRLs. This validation result is typically achieved when using test certificate chains, signer certificates with no OCSP responder URL or certificates derived from roots which are not part of the trusted service lists or configured into the signature validation component

Hint: The AIP-validator performs a structural validation of the presented archive containers. It makes use of an external signature validation component. The following table shows the specification of the test material. A more detailed description is given in Table 7.

Hint: Both the abbreviation ER and the abbreviation ERS are used for an Evidence Record.

Table 6 Test data specification overview

Testcase	*AIP-Format*	*Data*	*Signature formats*	*Timestamp (RFC3161)*	*Evidence Record (RFC 4998)*	*Format Validation Result*	*Signature Validation Result*
C-1	XAIP	binary	E			valid	valid
C-2	XAIP	binary		x		valid	valid
C-3	XAIP	binary	C, D, E, I with C, I with E, I with ER, J with C, J with E, J with ER, J with TS	x	x	valid	valid
C-4	XAIP	binary	H			valid	valid
C-5	XAIP	binary	A			valid	valid

Testcase	*AIP-Format*	*Data*	*Signature formats*	*Timestamp (RFC3161)*	*Evidence Record (RFC 4998)*	*Format Validation Result*	*Signature Validation Result*
C-6	XAIP	binary	B			valid	valid
C-7	XAIP		J with E			valid	valid
C-8	XAIP		J with TS	Part of ASiC-E		valid	valid
C-9	XAIP		J with ER		Part of ASiC-E	valid	valid
C-10	XAIP		J with C			valid	valid
C-11	XAIP	binary	C, TS	x		valid	valid
C-12	LXAIP	external	C (external reference)			valid	valid
C-13	ASiC-AIP	external	C			valid	valid
C-14	ASiC-AIP	external	C (external reference)			valid	valid
C-15	LXAIP	external				invalid	N/A
C-16	LXAIP	external	A, H			valid	invalid (A), valid (H)
C-17	LXAIP	external	ER		x	valid	invalid
C-18	XAIP	binary				invalid	N/A
C-19	XAIP	binary				invalid	N/A
C-20	XAIP	binary	C, TS	x		valid	valid
C-21	XAIP	binary	C, TS	x		invalid	invalid
C-22	XAIP		F			valid	valid
C-23	XAIP	xml	G			valid	valid
C-24	XAIP		J with TS	Part of ASiC-E		valid	invalid

Table 7 Test data specification overview

Remark: The test data specification contains the location of the signature object in the XAIP container. For better readability the namespace qualifiers are not shown.

Table 8 Test data supplied with the AIP-validator

No.	***Test file***	***Specification***
C-1	XAIP13-xades-det-single.xml	**Description:** An XAIP container with a detached xml signature. **AIP-Format:** XAIP **Data Objects:** a base64-encoded XML-document embedded into the XAIP-container (dataObject/binaryData) (object id DO-01) **Credential:** Detached XML-signature (object id CR-01) Data object location *dataObjectsSection/dataObject/binaryData* Signature object location *credentialsSection/credential/SignatureObject/Signature*
C -2	XAIP13-tsp-det-pdf.xml	**Description:** An XAIP container with data object and timestamp token pursuant to RFC 3161. **AIP-Format:** XAIP **Data Objects:** a PDF document with no signatures (object id DO-01). **Credentials:** Timestamp pursuant to RFC 3161 (object id CR-01) Data object location *dataObjectsSection/dataObject/binaryData* RFC-3161 timestamp token *credentialSection/credential/SignatureObject/Timestamp/RFC3161TimeStampToken*

No.	*Test file*	*Specification*
C -3	XAIP13-sig-all.xml	**Description:** XAIP container with 12 signatures, timestamps and evidence records. - CAdES attached - CAdES detached - XAdES detached - Multiple ASiC-S with single detached CAdES - ASiC-S with single detached XAdES - ASiC-S with single detached ER - ASiC-E with single detached CAdES - ASiC-E with single detached XAdES - ASiC-E with single detached ER - ASiC-E with single detached TSP - RFC 3161 timestamp token **AIP-Format:** XAIP **Data Objects:** - DO-02: ASCII-String (base64-encoded) - DO-04: PDF document (base64-encoded) - DO-13: ASCII String (base64-encoded) **Credentials:** - CR-01: CAdES-signature with merged payload - CR-02: Detached CAdeS-signature for data object DO-02 - CR-03: Detached XAdES-signature for data object DO-02 - CR-05: ASiC-S signature with single detached CAdES signature and mimetype - CR-06: ASiC-S signature with single detached XAdES signature and mimetype - CR-07: ASiC-S signature with single detached ER and mimetype - CR-08: ASiC-S signature with detached CAdES signature and mimetype - CR-09: ASiC-E signature with single detached TSP and mimetype - CR-10: ASiC-E with single detached XAdES and mimetype - CR-11: ASiC-E with single detached ER and mimetype - CR-12: ASiC-E with single detached TSP and mimetype - CR-13: RFC 3161 timestamp for data object DO-13 <u>XAdES detached signature location</u> *credentialsSection/credential/SignatureObject/Signature* <u>timestamp token location</u> *credentialSection/credential/SignatureObject/Timestamp/RFC3161TimeStampToken* <u>All other signature objects</u> *credentialsSection/credential/SignatureObject/Base64Signature*

No.	*Test file*	*Specification*
C-4	XAIP13-pdf-att-vis-single.xml	**Description:** XAIP container with PDF document including a visible signature **AIP-Format:** XAIP **Data Objects:** PDF document (object id DO-01) with embedded signature. No signature pointer is specified in the archive container. **Credentials:** N/A Data object location *dataObjectsSection/dataObject/binaryData*
C-5	XAIP13-cades-det-single.xml	**Description:** XAIP container with detached CMS signature The term 'detached CMS' signature is used to specify that the ASN.1 encoded signature object does not include the data object. The data object is stored separately. **AIP-Format:** XAIP **Data Objects:** ASCII text in base64 encoding (object id: DO-01) **Credentials:** detached CMS signature (object id: CR-01) Data object location *dataObjectsSection/dataObject/binaryData* Signature object location *credentialsSection/credential/SignatureObject/Base64Signature*
C-6	XAIP13-cades-att-single.xml	**Description:** XAIP container with attached CMS signature The term 'attached CMS' signature is used to specify that the ASN.1 encoded signature object includes the data object. **AIP-Format:** XAIP **Data Objects:** N/A **Credentials:** attached CMS signature (object id: CR-01) Signature object location credentialsSection/credential/SignatureObject/Base64Signature

No.	*Test file*	*Specification*
C-7	XAIP13-asice-xades-det-single.xml	**Description:** XAIP container with ASiC-E signature containing a single detached XAdES signature **AIP-Format:** XAIP **Data Objects:** N/A **Credentials:** ASiC-E signature containing a single detached XAdES signature (object id: CR-01) Signature object location *credentialsSection/credential/SignatureObject/Base64Signature*
C-8	XAIP13-asice-tsp-det-single.xml	**Description:** XAIP container with ASiC-E signature containing a single detached timestamp in the ASiC-E container **AIP-Format:** XAIP **Data Objects:** N/A **Credentials:** ASiC-E signature container with a single detached RFC 3161 timestamp Signature Object location *credentialsSection/credential/SignatureObject/Base64Signature*
C-9	XAIP13-asice-ers-det-single.xml	**Description:** XAIP container with ASiC-E signature and a single detached evidence record (ER) **AIP-Format:** XAIP **Data Objects:** N/A **Credentials:** ASiC-E signature and a single detached evidence record (ER) (object id: CR-01) Signature Object location *credentialsSection/credential/SignatureObject/Base64Signature*

No.	*Test file*	*Specification*
C-10	XAIP13-asice-cades-det-single.xml	**Description:** XAIP container with ASiC-E signature and a single detached CAdES signature **AIP-Format:** XAIP **Data Objects:** N/A **Credentials:** ASiC-E signature and a single detached CAdES signature (object id: CR-01) Signature Object location *credentialsSection/credential/SignatureObject/Base64Signature*
C-11	XAIP13-V1_V2_cades-det-txt_tsp-det-txt.xml	**Description:** An XAIP with two versions: - Version 1: V001: detached CAdES of a text file - Version 2: V002: a timestamp of a binary data **AIP-Format:** XAIP **Data Objects:** - ASCII text in base64-encoding (object id: DO-01) - ASCII text in base64-encoding (object id: DO-02) **Credentials:** - Detached CAdES signature (object id: CR-01) - RFC 3161 timestamp (object id: CR-02) Data object location *dataObjectsSection/dataObject/binaryData* Signature object location *credentialsSection/credential/SignatureObject/Base64Signature* V002: a timestamp of a binary data Data object location *dataObjectsSection/dataObject/binaryData* RFC-3161 timestamp token *credentialSection/credential/SignatureObject/Timestamp/RFC3161TimeStampToken*

No.	*Test file*	*Specification*
C-12	LXAIP13-cades-det-single.xml	**Description:** A LXAIP with detached CAdES of text file, addressed via references and not contained directly in XAIP itself: **AIP-Format:** LXAIP **Data Objects:** External file referenced by an URI (object id: DO-01) **Credentials:** External file referenced by an URI (object id: CR-01) Data object reference location *dataObjectsSection/dataObject/xmlData/DataObjectReference* Signature object reference location *credentialsSection/credential/other/DataObjectReference*
C-13	ASIC_AIP_XAIP_OK_SIG_OK.asice	**Description:** An ASiC-AIP container with an embedded XAIP containing two detached CAdES and unsigned metadata: **AIP-Format:** ASiC-AIP with embedded XAIP **Data Objects:** - base64-encoded ASCII document (object id: DO-01) - base64-encded PDF document (object id: DO-02) **Credentials:** - detached CAdES signature files (object ids: CR-01, CR-02) Detached CAdES of text file Data object location in embedded XAIP *dataObjectsSection/dataObject/binaryData* Signature object location in embedded XAIP *credentialsSection/credential/SignatureObject/Base64Signature* Detached CAdES signature of PDF file Data object location in embedded XAIP *dataObjectsSection/dataObject/binaryData* Signature object location in embedded XAIP *credentialsSection/credential/SignatureObject/Base64Signature* Unsigned XML-based metadata Metadata object location in embedded XAIP *metaDataSection/metaDataObject/binaryMetaData*

No.	***Test file***	***Specification***
C-14	ASIC_AIP_LXAIP_OK_ SIG_OK.asice	**Description:** An ASiC-AIP container with an embedded LXAIP referencing two detached CAdES and unsigned metadata: **AIP-Format:** ASiC-AIP with embedded LXAIP **Data Objects:** - reference to external object (URI) (object id: DO-01) - reference to external object (URI) (object id: DO-02) **Credentials:** - detached CAdES signature files (object ids: CR-01, CR-02) <u>Detached CAdES signature of text file</u> Data object reference location in embedded LXAIP *dataObjectsSection/dataObject/xmlData/DataObjectReference* Signature object reference location in embedded LXAIP *credentialsSection/credential/other/DataObjectReference* <u>Detached CAdES of PDF file</u> Data object reference location in embedded LXAIP *dataObjectsSection/dataObject/xmlData/DataObjectReference* Signature object reference location in embedded LXAIP *credentialsSection/credential/other/DataObjectReference* <u>Unsigned XML-based metadata</u> Metadata object reference location in embedded LXAIP *metaDataSection/metaDataObject/xmlMetaData/DataObjectReference*
C-15	LXAIP_NOK.xml	**Description:** An invalid LXAIP container with missing elements. **AIP-Format:** LXAIP **Data-Objects:** - reference to external xml file (object id: DO-01) - reference to external PDF file (object id: DO-02) **Credentials:** N/A <u>Reference to external xml file</u> *dataObjectsSection/dataObject/xmlData/DataObjectReference* <u>Reference to external PDF file</u> *dataObjectsSection/dataObject/xmlData/DataObjectReference*

No.	*Test file*	*Specification*
C-16	LXAIP_NOK_SIG.xml	**Description:** An LXAIP container with two document references, one with an invalid signature **AIP-Format:** LXAIP **Data Objects:** - reference to external text file (object id: DO-03) - reference to external PDF file (object id: DO-04) **Credentials:** - reference to external signature file (object id: CR-03) - reference to external signature file (object id: CR-04) Reference to external data objects *dataObjectsSection/dataObject/xmlData/DataObjectReference* Reference to external credentials *credentialsSection/credential/other/DataObjectReference*
C-17	LXAIP_NOK_ER.xml	**Description:** An LXAIP container with two document references and an invalid evidence record **AIP-Format:** LXAIP **Data Objects:** - reference to external text file (object id: DO-01) - reference to external PDF file (object id: DO-02) **Credentials:** - reference to external evidence record (object id: ER-01) Reference to external data objects *dataObjectsSection/dataObject/xmlData/DataObjectReference* Reference to external credentials *credentialsSection/credential/evidenceRecord/asn1EvidenceRecord*
C-18	XAIP_NOK.xml	**Description:** An XAIP container with two documents embedded in the data objects section. The container causes a schema validation error. **AIP-Format:** XAIP **Data-Objects:** - base64 encoded text (object id: DO-01) - base64 encoded PDF (object id: DO-02) **Credentials:** N/A Data object location *dataObjectsSection/dataObject/binaryData*

No.	*Test file*	*Specification*
C-19	XAIP_NOK_VERSION	**Description:** An XAIP container with the wrong XAIP version causing a validation error. **AIP-Format:** XAIP **Data-Objects:** - base64 encoded text (object id: DO-01) - base64 encoded PDF (object id: DO-02) **Credentials:** N/A Data object location *dataObjectsSection/dataObject/binaryData*
C-20	XAIP13-cades-det-txt_tsp-det-txt_idAsList.xml	**Description:** An XAIP container with two documents and two credentials. The data objects are protected by hash values stored in the id assignment list with object id IDAL-01. **AIP-Format:** XAIP **Data-Objects:** - base64 encoded ASCII text (object id's DO-01 and DO-02) **Credentials:** - detached CAdES signature (object id CR-01) - RFC 3161 timestamp (object id CR-02) Data object location *dataObjectsSection/dataObject/binaryData* Signature location *credentialsSection/credential/SignatureObject/Base64Signature* Timestamp location *credentialsSection/credential/SignatureObject/RFC3161TimestampToken* Assignment List location *packageHeader/versionManifest/idAssignmentList*

No.	*Test file*	*Specification*
C-21	XAIP13-cades-det-txt_tsp-det-txt_idAsList_NOK.xml	**Description:** An XAIP container with two documents and two credentials. The data objects are protected by hash values stored in the id assignment list with object id IDAL-01. One of the hash values has been manipulated so the validation of the XAIP container fails. **AIP-Format:** XAIP **Data-Objects:** - base64 encoded ASCII text (object id's DO-01 and DO-02) **Credentials:** - detached CAdES signature (object id CR-01) - RFC 3161 timestamp (object id CR-02) <u>Data object location</u> *dataObjectsSection/dataObject/binaryData* <u>Signature location</u> *credentialsSection/credential/SignatureObject/Base64Signature* <u>Timestamp location</u> *credentialsSection/credential/SignatureObject/RFC3161TimestampToken* <u>Assignment List location</u> *packageHeader/versionManifest/idAssignmentList*
C-22	XAIP13-xades-att-env-txt-single.xml	**Description:** An XAIP container with an eveloping XAdES signature. The signature is embedded in the credential section. **AIP-Format:** XAIP **Data-Objects:** N/A **Credentials:** - Enveloping XAdES signature (object id: CR-01) <u>Signature location</u> *credentialsSection/credential/SignatureObject*
C-23	XAIP13-xades-att-enveloped-xml-single.xml	**Description:** An XAIP container with an enveloped XAdES signature. The signed document is embedded in the data objects section. **AIP-Format:** XAIP **Data-Objects:** XML document with embedded XAdES signature (XAdES enveloped) (object id: DO-01) **Credentials:** N/A <u>Data object location</u> *dataObjectsSection/dataObject/xmlData*

No.	*Test file*	*Specification*
C-24	XAIP13-asice-tsp-det-single_NOK.xml	**Description:** An XAIP container with an ASiC-E signature and invalid timestamp **AIP-Format:** XAIP **Data-Objects**: N/A **Credentials:** ASiC-E signature container with single detached timestamp (object id: CR-01) <u>Signature location</u> *credentialsSection/credential/SignatureObject/Base64Signature*

13 Annex D: Structure of the validation report

The AIP-validator generates protocols according to the validation report schema, which is part of the TR-ESOR schema files. This section explains the structure of the validation report that is generated by the AIP-validator.

The root element of the report is an xml element of type *<VerificationReport>* as specified by OASIS in OASIS DSS v1.0 Profile for Comprehensive Multi-Signature Verification Reports Version 1.0.

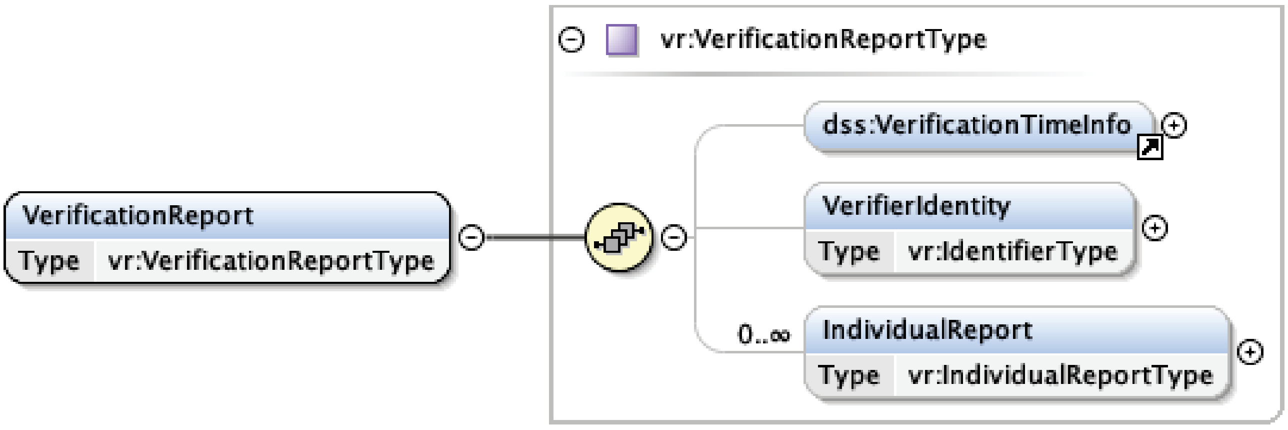

Figure 7 Schema of vr:VerificationReportType

The validation report will contain the elements *<VerificationTimeInfo>* and *<IndividualReport>*. The *<VerifierIdentity>* is not generated. The result of the AIP-container validation is stored in the <Result> Element. The *<Result>* element is specified as follows:

- **ResultMajor:** contains the cumulated validation result of the AIP-container and its signatures
- **ResultMinor:** not used by the AIP-validator
- **ResultMessage:** contains a human readable message string with the summary of the validation process

The element *<ResultMajor>* may contain one of the following values[7]:

- **Success** – The validation of the AIP-container and its signatures was successful. No invalid signatures. Seals, timestamps etc. were found and all AIP-container elements are conformant according to the TR-ESOR schema specifications.
- **InsufficientInformation** – The validation of signatures, timestamps, seals and evidence records was not performed due to an offline validation. The schema validation was successful.
- **RequesterError** – The validation of the AIP-container has failed due to format errors or the signature validation service responded with an error. This status will also be given if a digital signature, seal, evidence record or timestamp, which were part of the AIP-container, were invalid.
- **ResponderError** – The response of the verification service cannot be processed., e.g. an unknown schema has been used in the response structure

[7] The namespace for the values given in the list is “urn:oasis:names:tc:dss:1.0:resultmajor:”

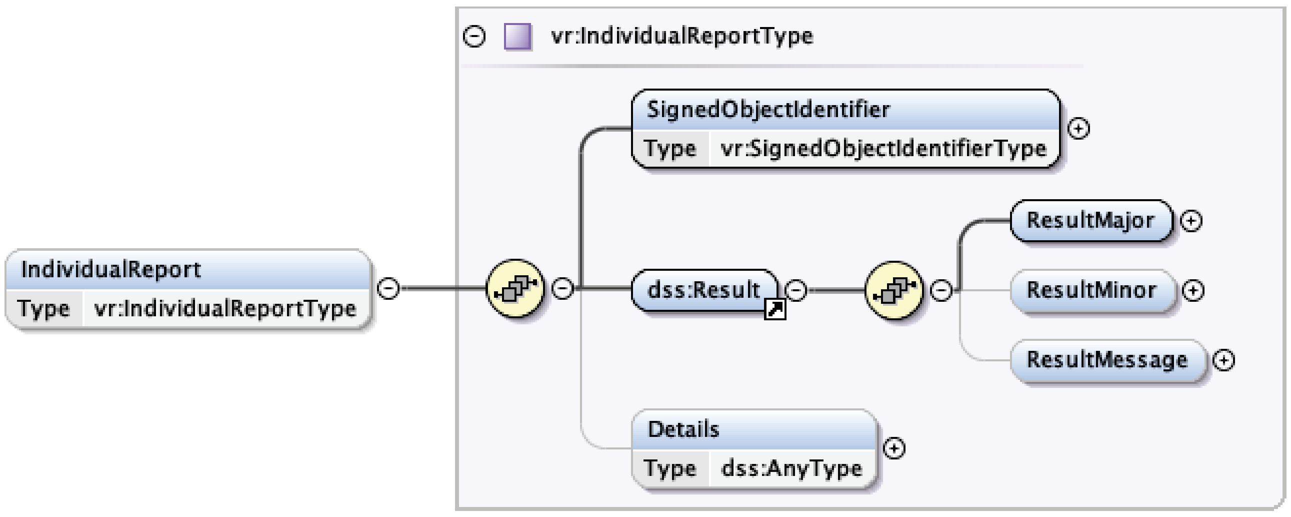

Figure 8 Schema of IndividualReport

The element *<IndividualReport>* is generated exactly once. All validation information will be stored in the Details element. The Details-element will contain a node of type *<XAIPReport>*.

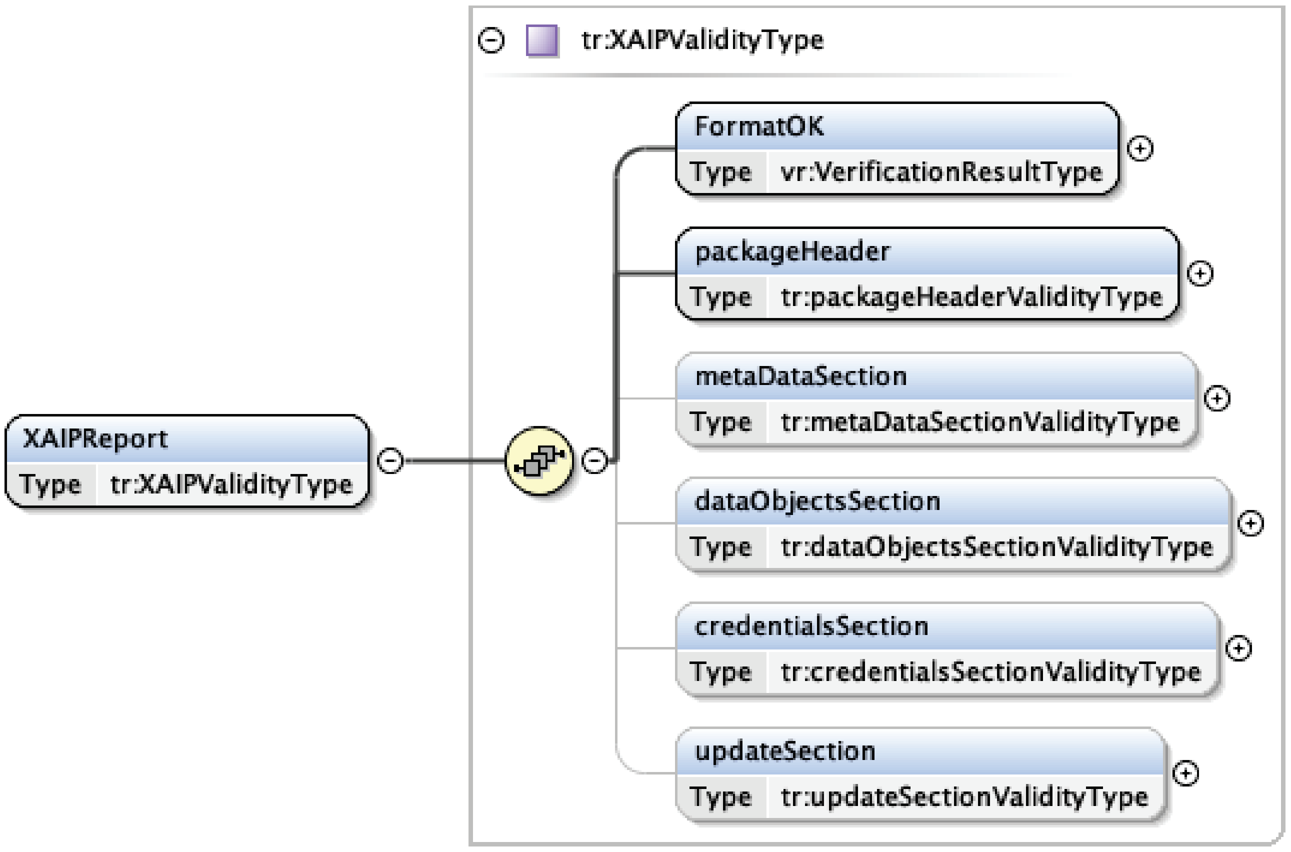
tr:XAIPValidityType
XAIPReport
Type tr:XAIPValidityType
FormatOK
Type vr:VerificationResultType
packageHeader
Type tr:packageHeaderValidityType
metaDataSection
Type tr:metaDataSectionValidityType
dataObjectsSection
Type tr:dataObjectsSectionValidityType
credentialsSection
Type tr:credentialsSectionValidityType
updateSection
Type tr:updateSectionValidityType

Figure 9 Schema of XAIPReport

The specification of the *<XAIPReport>* element and its sub-elements is given in Annex TR-ESOR-VR: Verification Reports for Selected Data Structures.

Hint: In case of digital signatures, seals, timestamps or evidence records, which are not embedded as credentials into the AIP-container, a virtual credential id is generated by the AIP-validator and used in the validation report.

Furthermore, the rules defined in Annex TR-ESOR-VR apply to the validation report generated by the AIP-validator.