Bundesamt
für Sicherheit in der
Informationstechnik

BSI Technical Guideline 03125
Preservation of Evidence of Cryptographically Signed Documents

Annex TR-ESOR-C.2:
Conformity Test Specification (Level 2 - Technical Conformity)

Designation	Technical Conformity Test Specification (Level 2)
Abbreviation	BSI TR-ESOR-C.2
Version	1.2
Date	30.06.15

Part of the document is not barrier free.

Federal Office for Information Security
Post Box 20 03 63
53133 Bonn
Phone: +49 228 99 9582-0
E-Mail: tresor@bsi.bund.de
Internet: https://www.bsi.bund.de

Table of Contents

List of Figures

List of Tables

1. Introduction

The goal of the Technical Guideline "Preservation of Evidence of Cryptographically Signed Documents" is to specify technical security requirements for the long-term preservation of evidence of cryptographically signed electronic documents and data along with associated electronic administrative data (meta data).
A Middleware defined for this purpose (TR-ESOR-Middleware) in the sense of this Guideline includes all of the modules (**M**) and interfaces (**S**) [for the German "*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 in the following:

- The input interface S.4 of the TR-ESOR-Middleware serves to embed the TR-ESOR-Middleware in the existing IT and infrastructure landscape;
- The central Middleware module (**[TR-ESOR-M.1]**), which regulates the flow of information in the Middleware, that implements the security requirements for the interfaces with the IT applications and which ensures that the application systems are decoupled from the ECM/long-term storage;
- The "Cryptographic" module (**[TR-ESOR-M.2]**) and the associated interfaces S.1 and S.3 that provide the functions needed for the creation (optional) and verification of electronic signatures, the post-verification of electronic certificates, and for the obtainment of qualified time stamps for the Middleware. Furthermore, it can provide the functions for the encryption and decryption of data and documents;
- The "ArchiSig" module (**[TR-ESOR-M.3]**) with the interface S.6 that provides the functions needed for the preservation of evidence of the digitally signed documents;
- An ECM/long-term storage with the interfaces S.2 and S.5 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 may be induced through the two interfaces that are still part of the TR-ESOR-Middleware.
  The application layer that can include an XML-adapter is not a direct part of this Technical Guideline, either, even though this XML-adapter can be implemented as part of a Middleware.

The IT Reference Architecture depicted in 1 is based on the ArchiSafe[1] 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] For more information, see http://www.archisafe.de.

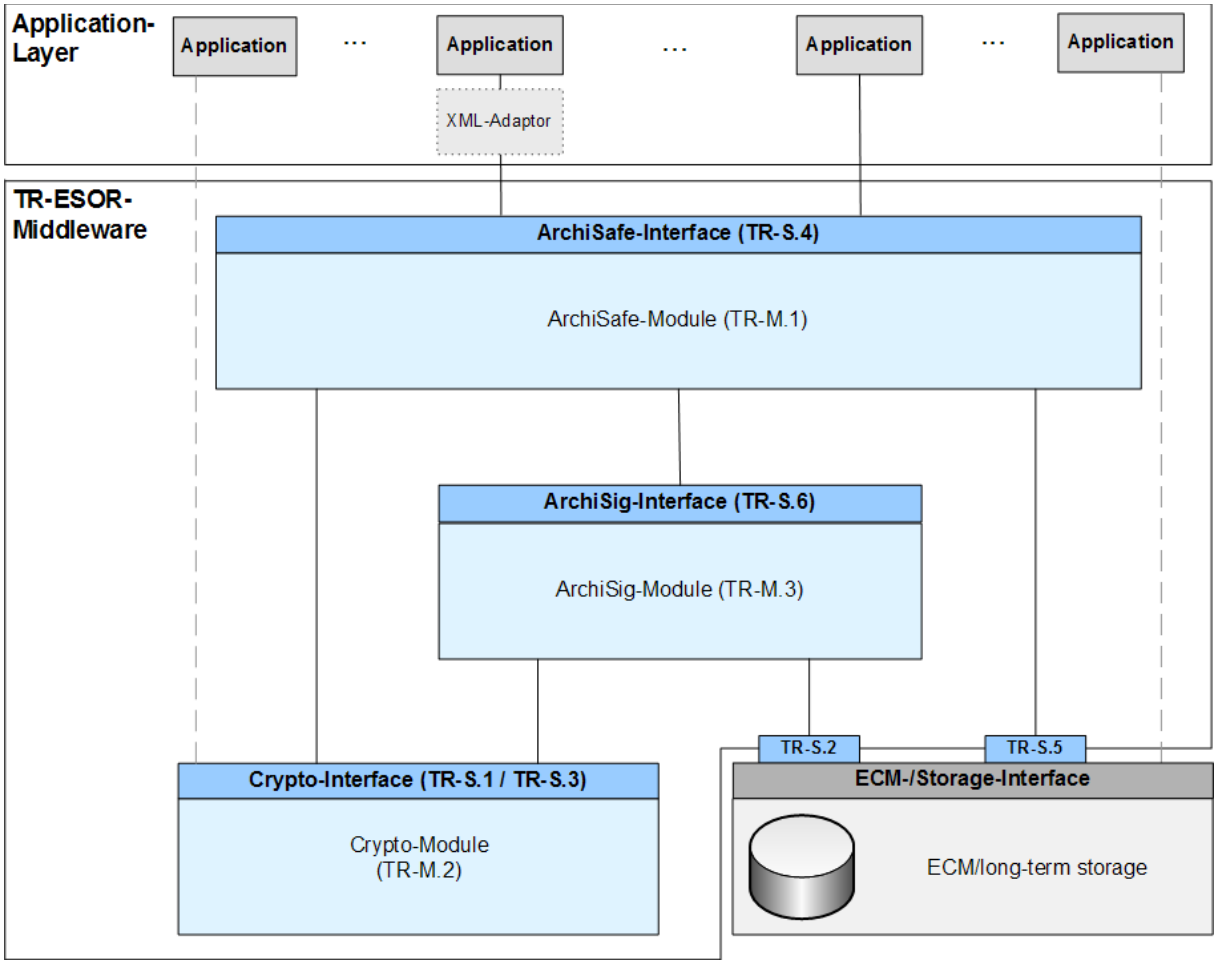

Figure 1: Schematic Depiction of the IT Reference Architecture

This Technical Guideline is modularly structured, and the individual annexes to the Main Document specify the functional and technological security requirements for the needed IT components and interfaces of the TR-ESOR-Middleware. The specifications are strictly platform, product and manufacturer independent.

The document at hand bears the designation “Annex TR-ESOR-C.2” and describes and specifies the conformity tests for the Conformity Level 2 (Technical Conformity).

2. Overview

Products or systems, which want to get certified according to this Technical Guideline, have to demonstrate their conformance to the specifications. There are three defined conformance levels, which mainly differ in the technical detail specifications of interfaces and data formats used.

- Conformity Level 1 – Functional Conformity
- Conformity Level 2 – Technical Conformity
- Conformity Level 3 – Recommendations for Federal Agencies

The three levels are built on top of each other. This means e.g. in order to demonstrate conformity to level 2 all conformance criteria for level 1 have to be passed in addition to the conformance criteria for level 2.
This document specifies the test criteria for reaching Conformity Level 2 (Technical Conformity), which are derived from the requirements and web service interfaces specified in Annex [TR-ESOR-E], [TR-ESOR-F], [TR-ESOR-ERS] and [TR-ESOR-VR] and complement the mandatory (red) tests for interface functions specified in section 5.5 of Annex [TR-ESOR-C.1] of the present Technical Guideline.
The tests for Conformity Level 2 aim at verifying technical interoperability between different components within the reference architecture as depicted in Figure 1. For this purpose the present document specifies test cases for the different web service based interfaces (see section 3) and verification report structures for central data structures (see section [TR-ESOR-VR]).
Each test case is identified by a unique ID and specified in a semi-formal pseudo-code, which provides the basis for the technical implementation of the test cases in an appropriate test environment.
In general the optional elements of the request – and response structures of the functional calls according Annex E are omitted. The usage or omission of these optional elements has no impact on the fulfilling on the conformance requirements of this specification.
The tests of the web service based interfaces can be performed using the freely available SoapUI[2] test tool.
All tests, which are required to be passed are marked with the color red. Tests, which are only applicable in certain situations are marked grey. All tests use the same configuration, which which is denoted by CONFIG_Implicit in the following.
In order to become certified according to Conformity Level 2, a product or system must pass all mandatory (red) conformity criteria (tests) for this conformity level and for all lower conformity levels. If one or more tests are not successful, the conformity cannot be certified.

[2] See http://www.soapui.org .

3. Web Service Interfaces

As shown in Figure 1 the reference architecture comprises different interfaces, which can be implemented by a TR-ESOR-compliant middleware product. The present specification defines test suites for the interfaces S.1, S.3, S.4 and S.6 (see section 3.1), which comprise test cases for different functions (see section 3.3 - 3.9) using the different standard test objects as defined in section 3.2.

3.1 Test suites for interfaces S.*x*

3.1.1 Test suite for S.1

In addition to the test cases for VerifyRequest and SignRequest as defined in the scope of the eCard-API-Framework (see [eCard-Test]) the test suite for the interface S.1 contains the test cases for VerifyRequest defined in section 3.9.

3.1.2 Test suite for S.3

In addition to the test cases for VerifyRequest, SignRequest as defined in the scope of the eCard-API-Framework (see [eCard-Test]) the test suite for the interface S.3 contains the test cases for VerifyRequest defined in section 3.9.

3.1.3 Test suite for S.4

The test suite for S.4 contains the test cases for the following functions:

- ArchiveSubmissionRequest (see section 3.3)
- ArchiveUpdateRequest (see section 3.4)
- ArchiveRetrievalRequest (see section 3.5)
- ArchiveEvidenceRequest (see section 3.6)
- ArchiveDeletionRequest (see section 3.7)
- ArchiveDataRequest (see section 3.8)

3.1.4 Test suite for S.6

The test suite for S.6 contains the test cases for the following functions:

- ArchiveSubmissionRequest (see section 3.3)
- ArchiveUpdateRequest (see section 3.4)
- ArchiveEvidenceRequest (see section 3.6)

3.2 Standard Test Objects

Container Name	Description
XAIP_OK	The XAIP is syntactically correct and passes the defined consistency checks.
XAIP_NOK	The schema validation of the XAIP fails.
XAIP_NOK_EXPIRED	The schema validation for the XAIP succeeds, but the `preservationInfo`-element indicates a preservation date, which is already exceeded.
XAIP_NOK_SUBMTIME	The schema validation for the XAIP succeeds, but the `submissionTime`-element deviates from the current time beyond a reasonable tolerance range. The documentation of the middleware or the module, which shall be tested, **shall** contain some assertions and related conditions or constraints indicating when the submissionTime

Container Name	Description
	contained in the provided XAIP deviates too much from the current time.
XAIP_NOK_SIG	The XAIP is syntactically correct and passes the defined consistency checks, but the XAIP contains an invalid signature. Invalid signature means that the signature is syntactically not correct or at least one of the evidence relevant data, for example a signature, timestamp, certificate, certificate revocation list or OCSP-response, etc., is wrong.
XAIP_NOK_ER	The XAIP is syntactically correct and passes the defined consistency checks, but the XAIP contains an invalid Evidence Record. An invalid Evidence Record means, that the Evidence Record is syntactically not correct or does not fulfil the requirements according to Annex ERS.
XAIP_NOK_SIG_OK_ER	The XAIP is syntactically correct and passes the defined consistency checks and there is a correct Evidence Record or a number of correct Evidence Records, but the XAIP contains an invalid signature.
XAIP_OK_SIG_OK_ER	The XAIP is syntactically correct and passes the defined consistency checks and there is a valid signature and a valid Evidence Record or a number of correct Evidence Records.
BIN	This test object is a binary document, which is provided in the `ArchiveData`-element.
XAIP(BIN)	The XAIP(BIN) is a XAIP, which is part of the response of a successful ArchiveRetrievalRequest concerning an archive data object, which was previously inserted as a BIN in the long-term storage by an ArchiveSubmissionRequest.
DXAIP_OK	The XAIP is syntactically correct and represents a valid update container (“Delta XAIP”) for XAIP_OK, which contains the corresponding AOID.
DXAIP_NOK	The XAIP is syntactically not correct because the schema validation fails.
DXAIP_NOK_AOID	The schema validation for the Delta XAIP succeeds, but the update container (“Delta XAIP”) contains a not yet assigned AOID.
DXAIP_NOK_EXPIRED	The schema validation for the Delta XAIP succeeds, but the `preservationInfo`-element indicates a point in time in the past.
DXAIP_NOK_SUBMTIME	The schema validation for the Delta XAIP succeeds, but the `submissionTime`-element deviates from the current time beyond a reasonable tolerance range. The documentation of the middleware or the module, which

Container Name	Description
	shall be tested, **shall** contain some assertions and related conditions or constraints indicating when the submissionTime contained in the provided XAIP deviates too much from the current time.
DXAIP_NOK_SIG	The schema validation for the Delta XAIP succeeds, but the XAIP contains an invalid signature.
DXAIP_NOK_ER	The schema validation for the Delta XAIP succeeds, but the XAIP contains an invalid Evidence Record.
DXAIP_NOK_VERSION	The schema validation for the XAIP succeeds, but there is a syntactical collision with the original XAIP such that the schema validation for the compound XAIP fails, for example the element prevVersion in the updateSection of the DXAIP is not the latest version of this XAIP.
DXAIP_NOK_ID	The DXAIP contains no or an invalid AOID.
TST_OK	The time stamp token is syntactically correct and based on a valid signature.
TST_OK_VALINFO	This time stamp token is based on TST_OK and contains the validation information, which has been collected during verification.
TST_NOK	The time stamp token is syntactically incorrect.
TST_NOK_SIG	The time stamp token is syntactically correct, but the signature does not verify correctly.
TST_NOK_VALINFO	This time stamp token is based on TST_OK and contains validation information, which has been collected during verification, but are not complete.
ER_OK_INIT	The Evidence Record according [RFC4998] and [TR-ESOR-ERS]/Basic-ERS-Profile and based on XAIP_OK contains only an initial archive timestamp.
ER_NOK_INIT	The initial archive timestamp of the Evidence Record according [RFC4998] and [TR-ESOR-ERS]/Basic-ERS-Profile and based on XAIP_OK can not be validated.
ER_OK_CHAIN	The Evidence Record according to [TR-ESOR-ERS]/Basic-ERS-Profile is based on XAIP_OK and includes a sequence of archive timestamps within the archive timestamp chain according to [RFC4998].
ER_NOK_CHAIN	The Evidence Record according to [TR-ESOR-ERS]/Basic-ERS-Profile and based on XAIP_OK includes a sequence of archive timestamps within the archive timestamp chain according to [RFC4998] which can not be validated.
ER_OK_SEQ	The Evidence Record according to

Container Name	Description
	[TR-ESOR-ERS]/Basic-ERS-Profile is based on XAIP_OK and includes a sequence of archive timestamp chains in the archive timestamp sequence according to [RFC4998].
ER_NOK_SEQ	The Evidence Record according to [TR-ESOR-ERS]/Basic-ERS-Profile and based on XAIP_OK includes a sequence of archive timestamp chains in the archive timestamp sequence according to [RFC4998] which can not be validated.

Table 1 Definition of test data

Furthermore, data elements of the type “anyType” are defined in the namespace "urn:oasis:names:tc:dss:1.0:core:schema" which can be found at http://ws.openecard.org/schema/oasis-dss-core-schema-v1.0-os.xsd.

3.3 ArchiveSubmissionRequest

All test cases defined in this section are derived from the requirement M.1:A4.1-1 and the general requirement E:A2.0-2 together with the interface specification of the ArchiveSubmissionRequest function in section 3.1 of [TR-ESOR-E].

3.3.1 SU-01 – XAIP_OK without AOID

Identifier	SU-01
Test Purpose	The test shall verify that it is possible to submit XAIP_OK without AOID.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists.

Step	Test sequence	Expected Results
1	ArchiveSubmissionRequest (XAIP_OK)	ArchiveSubmissionResponse (dss:Result(resultmajor#ok), AOID_SU-01)
Observations:		
Verdict:		

3.3.2 SU-02 – XAIP_OK with not yet assigned AOID

This test is only applicable, if the middleware allows that the client application specifies the AOID[3].

Identifier	SU-02
Test Purpose	The test shall verify that it is possible to submit an XAIP_OK and an explicitly specified AOID, which has not been assigned yet.
Configuration	CONFIG_Implicit
Pre-test conditions	Requirements (A3.1.1-01) and (A3.1.2-01): The usage of the dss:OptionalInputs "ImportEvidence" in order to import an evidence record ER_OK_INIT, ER_OK_CHAIN or ER_OK_SEQ is optional. The usage of the verificationReport-element is optional. Make sure that the middleware allows to retrieve archive data and to submit archive data with AOIDs specified by the client application (cf. Step 1 and Step 2). Authenticated connection to middleware exists.

Step	Test sequence	Expected Results
1	Repeat this step until the expected result has been achieved or skip the test case as failed after ten trials. Randomly generate AOID_SU-02 within the possible range and make sure that it has not been assigned yet. This is realised by the following call: ArchiveRetrievalRequest (AOID_SU-02)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
2	ArchiveSubmissionRequest (dss:OptionalInputs(AOID(AOID_SU-02)), XAIP_OK)	ArchiveSubmissionResponse (dss:Result(resultmajor#ok), AOID_SU-02)
3	Repeat this step until the expected result has been achieved or skip the test case as failed after ten trials. Randomly generate AOID_SU-03 within the possible range and make sure that it has not been assigned yet. This is realised by the following call: ArchiveRetrievalRequest (AOID_SU-03)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
4	ArchiveSubmissionRequest (dss:OptionalInputs(AOID(AOID_SU-03), ReturnVerificatonReport, ImportEvidence(ER_NOK_INIT)), XAIP_OK[4])	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/XAIP_NOK_ER), dss:OptionalOutput(VerificationReport))
5	ArchiveSubmissionRequest (dss:OptionalInputs(AOID(AOID_SU-03), ReturnVerificatonReport, ImportEvidence(ER_OK_INIT)),	ArchiveSubmissionResponse (dss:Result(resultmajor#ok), dss:OptionalOutput(VerificationReport), AOID_SU-03

[3] **NOTE:** The client application may generate an AOID. Therefore the <AOID>-element is optional. See also Annex F v1.2, chapter 3.2, page 10.

[4] The Evidence Record is stored in dss:OptionalInputs.

	XAIP_OK)	)
6	Repeat this step until the expected result has been achieved or skip the test case as failed after ten trials. Randomly generate AOID_SU-04 within the possible range and make sure that it has not been assigned yet. This is realised by the following call: ArchiveRetrievalRequest (AOID_SU-04)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
7	Prepare a XAIP_OK with a retentionPeriod “one day after the day of this submission”.	
8	ArchiveSubmissionRequest (dss:OptionalInputs(AOID(AOID_SU-04), ReturnVerificatonReport, ImportEvidence(CredentialID_ER)), xaip:XAIP(XAIP_OK+ER_NOK_INIT))	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/XAIP_NOK_ER), dss:OptionalOutput(VerificationReport))
9	ArchiveSubmissionRequest (dss:OptionalInputs(AOID(AOID_SU-04), ReturnVerificatonReport, ImportEvidence(CredentialID_ER)), xaip:XAIP(XAIP_OK+ER_OK_INIT)[5])	ArchiveSubmissionResponse (dss:Result(resultmajor#ok), dss:OptionalOutput(VerificationReport), AOID_SU-04)
10	Repeat this step until the expected result has been achieved or skip the test case as failed after ten trials. Randomly generate AOID_SU-05 within the possible range and make sure that it has not been assigned yet. This is realised by the following call: ArchiveRetrievalRequest (AOID_SU-05)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
11	ArchiveSubmissionRequest (dss:OptionalInputs(AOID(AOID_SU-05), ReturnVerificatonReport, ImportEvidence(ER_NOK_CHAIN)), XAIP_OK)	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/XAIP_NOK_ER), dss:OptionalOutput(VerificationReport))
12	ArchiveSubmissionRequest (dss:OptionalInputs(AOID(AOID_SU-05), ReturnVerificatonReport, ImportEvidence(ER_OK_CHAIN)), XAIP_OK)	ArchiveSubmissionResponse (dss:Result(resultmajor#ok), dss:OptionalOutput(VerificationReport), AOID_SU-05)
13	ArchiveRetrievalRequest(AOID(AOID_SU-05), VersionID(v1))	ArchiveRetrievalResponse(dss:Result(resultmajor#OK), XAIP_OK(VID_SU-05)[6])

[5] The Evidence Record is stored in the credential section of XAIP_OK.

[6] The original first version (v1) is returned → XAIP_OK.

14	ArchiveEvidenceRequest(AOID(AOID_SU_05), VersionID(v1))	ArchiveEvidenceResponse(dss:Result(resultmajor#ok), xaipERS(AOID_SU-05,VID_SU-01, ASN.1-ERS(XAIP_OK)[7])))
15	VerifyRequest (dss:OptionalInputs (VerifyUnderSignaturePolicy (DefaultPolicy (SignaturePolicyIdentifier (http://www.bsi.bund.de/ tr-esor/sigpolicy/verify-xaip)))), dss:InputDocuments (dss:Document (Base64XML(XAIP_OK(VID_SU-05) + xaipERS(AOID_SU-05,VID_SU-01, ASN.1-ERS(XAIP_OK)))))	VerifyResponse(dss:Result(resultmajor#ok,))
16	Repeat this step until the expected result has been achieved or skip the test case as failed after ten trials. Randomly generate AOID_SU-06 within the possible range and make sure that it has not been assigned yet. This is realised by the following call: ArchiveRetrievalRequest (AOID_SU-06)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
17	ArchiveSubmissionRequest (dss:OptionalInputs(AOID(AOID_SU-06), ReturnVerificatonReport, ImportEvidence(ER_NOK_SEQ)), XAIP_OK)	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/XAIP_NOK_ER), dss:OptionalOutput(VerificationReport))
18	ArchiveSubmissionRequest (dss:OptionalInputs(AOID(AOID_SU-06), ReturnVerificatonReport, ImportEvidence(ER_OK_SEQ)), XAIP_OK)	ArchiveSubmissionResponse (dss:Result(resultmajor#ok), dss:OptionalOutput(VerificationReport), AOID_SU-06)
19	ArchiveRetrievalRequest(AOID(AOID_SU-06), VersionID(v1))	ArchiveRetrievalResponse(dss:Result(resultmajor#OK), XAIP_OK(VID_SU-06)[8])
20	ArchiveEvidenceRequest(	ArchiveEvidenceResponse(

[7] The Evidence Record belonging to version v1 is returned.
[8] The original first version v1 is returned → XAIP_OK.

	AOID(AOID_SU_06), VersionID(v1))	dss:Result(resultmajor#ok), xaipERS(AOID_SU-06,VID_SU-01, ASN.1-ERS(XAIP_OK)[9])))
21	VerifyRequest (dss:OptionalInputs (VerifyUnderSignaturePolicy (DefaultPolicy (SignaturePolicyIdentifier (http://www.bsi.bund.de/ tr-esor/sigpolicy/verify-xaip)))), dss:InputDocuments (dss:Document (Base64XML(XAIP_OK(VID_SU-06) + xaipERS(AOID_SU-06,VID_SU-01, ASN.1-ERS(XAIP_OK)))))	VerifyResponse(dss:Result(resultmajor#ok,))
Observations:		
Verdict:		

[9] The Evidence Record belonging to version v1 is returned.

3.3.3 SU-03 – XAIP_OK with already assigned AOID

This test is only applicable, if the middleware allows that the client application specifies the AOID[10].

Identifier	SU-03	
Test Purpose	The test shall check that the submission of an XAIP_OK with an explicitly specified AOID, which is already assigned yields an error.	
Configuration	CONFIG_Implicit	
Pre-test conditions	The usage of the verificationReport-element is optional. Make sure that the middleware allows to submit archive data with AOIDs specified by the client application (cf. Step 1). Furthermore make sure that the test case SU-02 was successfully performed using AOID(AOID_SU-02). Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1	ArchiveSubmissionRequest (dss:OptionalInputs(AOID(AOID_SU-02), ReturnVerificatonReport), XAIP_OK)	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/existingAOID), dss:OptionalOutput(VerificationReport))
2	ArchiveSubmissionRequest (dss:OptionalInputs(AOID(AOID_SU-02), ReturnVerificatonReport, ImportEvidence(ER_OK_INIT)), XAIP_OK)	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/existingAOID), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

[10] **NOTE:** The client application may generate an AOID. Therefore the <AOID>-element is optional. See also Annex F (Version 1.2, Chapter 3.2, page 10).

3.3.4 SU-04 – XAIP_NOK

Identifier	SU-04	
Test Purpose	The test shall verify that there will be an error, if a (syntactically incorrect) XAIP_NOK is submitted.	
Configuration	CONFIG_Implicit	
Pre-test conditions	The usage of the verificationReport-element is optional. Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1	ArchiveSubmissionRequest (dss:OptionalInputs(ReturnVerificatonReport)), XAIP_NOK)	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/XAIP_NOK), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

3.3.5 SU-05 – XAIP_NOK_EXPIRED

Identifier	SU-05	
Test Purpose	The test shall verify that there will be an error, if the semantically incorrect XAIP_NOK_EXPIRED is submitted.	
Configuration	CONFIG_Implicit	
Pre-test conditions	The usage of the verificationReport-element is optional. Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1	ArchiveSubmissionRequest (dss:OptionalInputs(ReturnVerificatonReport)), XAIP_NOK_EXPIRED)	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/XAIP_NOK_EXPIRED), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

3.3.6 SU-06 – XAIP_NOK_SUBMTIME

Identifier	SU-06	
Test Purpose	The test shall verify that there will be an error, if the semantically incorrect XAIP_NOK_SUBMTIME is submitted.	
Configuration	CONFIG_Implicit	
Pre-test conditions	The usage of the verificationReport-element is optional. Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1	ArchiveSubmissionRequest (dss:OptionalInputs(ReturnVerificatonReport), XAIP_NOK_SUBMTIME)	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/XAIP_NOK_SUBMTIME), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

3.3.7 SU-07 – XAIP_NOK_SIG

Identifier	SU-07	
Test Purpose	The test shall verify that there will be an error, if the submitted XAIP_NOK_SIG contains signatures, which do not verify correctly[11].	
Configuration	CONFIG_Implicit	
Pre-test conditions	The usage of the verificationReport-element is optional. Furthermore the usage of the OptionalInputs “ImportEvidence” with a correct evidence record ER_OK is optional. Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1	ArchiveSubmissionRequest (dss:OptionalInputs(ReturnVerificatonReport, ImportEvidence(ER_OK)), XAIP_NOK_SIG)	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/XAIP_NOK_SIG), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

[11] See also Annex E (Version 1.2, Chapter 3.7.1 (`VerifyUnderSignaturePolicy`).

3.3.8 SU-08 – XAIP_NOK_ER

Identifier	SU-08	
Test Purpose	The test shall verify that there will be an error, because the submitted XAIP_NOK_ER contains an incorrect Evidence Record.	
Configuration	CONFIG_Implicit	
Pre-test conditions	The usage of the verificationReport-element is optional. Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1	ArchiveSubmissionRequest (dss:OptionalInputs(ReturnVerificatonReport), XAIP_NOK_ER)	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/XAIP_NOK_ER), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

3.3.9 SU-09 – XAIP_NOK_SIG_OK_ER

Identifier	SU-09	
Test Purpose	The test shall verify that there will be an error, if the XAIP_NOK_SIG_OK_ER contains signatures, which do not verify correctly, even if there is a valid Evidence Record.	
Configuration	CONFIG_Implicit	
Pre-test conditions	The usage of the verificationReport-element is optional. Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1	ArchiveSubmissionRequest (dss:OptionalInputs(ReturnVerificatonReport), XAIP_NOK_SIG_OK_ER)	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/XAIP_NOK_SIG), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

3.3.10 SU-10 – XAIP_OK_SIG_OK_ER

Identifier	SU-10	
Test Purpose	The test shall verify that it is possible to submit an XAIP_OK_SIG_OK_ER, which contains signatures, which have been valid at the archiving time and a valid Evidence Record.	
Configuration	CONFIG_Implicit	
Pre-test conditions	Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1	ArchiveSubmissionRequest (XAIP_OK_SIG_OK_ER)	ArchiveSubmissionResponse (dss:Result(resultmajor#ok), AOID_SU-10)
Observations:		
Verdict:		

3.3.11 SU-11 – BIN without AOID

Identifier	SU-11	
Test Purpose	The test shall verify that it is possible to submit BIN without AOID.	
Configuration	CONFIG_Implicit	
Pre-test conditions	This test is only applicable, if the middleware allows to submit binary documents as archive data. Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1	ArchiveSubmissionRequest (ArchiveData(BIN))	ArchiveSubmissionResponse (dss:Result(resultmajor#ok), AOID_SU-11)
Observations:		
Verdict:		

3.3.12 SU-12 – BIN with not yet assigned AOID

This test is only applicable, if the middleware allows that the client application specifies the AOID.[12]

Identifier	SU-12	
Test Purpose	The test shall verify that it is possible to retrieve archive data and to submit a binary object and an explicitly specified AOID, which has not been assigned yet.	
Configuration	CONFIG_Implicit	
Pre-test conditions	This test is only applicable, if the middleware allows to submit binary documents as archive data. Make sure that the middleware allows to submit archive data with AOIDs specified by the client application (cf. Step 1 and Step 2). Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1	Repeat this step until the expected result has been achieved or skip the test case as failed after ten trials. Randomly generate AOID_SU-12 within the possible range and make sure that it has not been assigned yet. This is realised by the following call: ArchiveRetrievalRequest (AOID_SU-12)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
2	ArchiveSubmissionRequest (dss:OptionalInputs(AOID(AOID_SU-12)), ArchiveData(BIN))	ArchiveSubmissionResponse (dss:Result(resultmajor#ok), AOID_SU-12)
Observations:		
Verdict:		

[12] **NOTE:** The client application may generate an AOID. Therefore the <AOID>-element is optional. See also Annex F (Version 1.2, Chapter 3.2, page 10).

3.3.13 SU-13 – BIN with already assigned AOID

This test is only applicable, if the middleware allows that the client application specifies the AOID.[13]

Identifier	SU-13	
Test Purpose	The test shall check that the submission of a binary object with an explicitly specified AOID, which is already assigned yields an error.	
Configuration	CONFIG_Implicit	
Pre-test conditions	This test is only applicable, if the middleware allows to submit binary documents as archive data. The usage of the verificationReport-element is optional. Make sure that the middleware allows to submit archive data with AOIDs specified by the client application (cf. Step 1). Furthermore make sure that the test case SU-12 was successfully performed using AOID(AOID_SU-12). Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1	ArchiveSubmissionRequest (dss:OptionalInputs(AOID(AOID_SU-12) dss:OptionalInputs(ReturnVerificatonReport)), ArchiveData(BIN))	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/existingAOID), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

[13] **NOTE:** The client application may generate an AOID. Therefore the <AOID>-element is optional. See also Annex F (Version 1.2 , Chapter 3.2, page 10).

3.3.14 SU-14 – Unknown control in `OptionalInputs`

Identifier	SU-14	
Test Purpose	The test shall verify that there will be an error, if the request contains unknown controls in the `OptionalInputs`-element and the archive data object will not be imported.	
Configuration	CONFIG_Implicit	
Pre-test conditions	The usage of the verificationReport-element is optional. Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1	ArchiveSubmissionRequest (dss:OptionalInputs(ReturnVerificatonReport, SomethingUnknown), XAIP_OK)	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/notSupported), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

3.3.15 SU-15 – Unknown control in `OptionalInputs` with AOID

This test is only applicable, if the middleware allows that the client application specifies the AOID.[14]

Identifier	SU-15
Test Purpose	The test shall verify that there will be an error, if the request contains unknown controls in the `OptionalInputs`-element and the archive data object will not be imported.
Configuration	CONFIG_Implicit
Pre-test conditions	Make sure that the middleware allows to retrieve archive data and to submit archive data with AOIDs specified by the client application. The usage of the verificationReport-element is optional. Authenticated connection to middleware exists.

Step	**Test sequence**	**Expected Results**
1	Repeat this step until the expected result has been achieved or skip the test case as failed after ten trials. Randomly generate AOID_SU-15 within the possible range and make sure that it has not been assigned yet. This is realised by the following call: ArchiveRetrievalRequest (AOID_SU-15)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
2	ArchiveSubmissionRequest (dss:OptionalInputs(AOID(AOID_SU-15), ReturnVerificatonReport, SomethingUnknown), XAIP_OK)	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/notSupported), dss:OptionalOutput(VerificationReport))
3	ArchiveRetrievalRequest (dss:OptionalInputs(ReturnVerificatonReport)), AOID_SU-15)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

[14] **NOTE:** The client application may generate an AOID. Therefore the <AOID>-element is optional. See also Annex F (Version 1.2 , Chapter 3.2, page 10).

3.3.16 SU-16 – Unknown ArchiveDataType

Identifier	SU-16	
Test Purpose	The test shall verify that there will be an error, if the request contains an unknown NEW_ArchiveData_Type (not XAIP and not BIN) and the archive data object will not be imported.	
Configuration	CONFIG_Implicit	
Pre-test conditions	The usage of the verificationReport-element is optional. Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1	ArchiveSubmissionRequest (dss:OptionalInputs(ReturnVerificatonReport), ArchiveData(New_ArchiveData_Type))	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/unknownArchiveDataType), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

3.3.17 SU-17 – Unknown ArchiveDataType with AOID

This test is only applicable, if the middleware allows that the client application specifies the AOID.[15]

Identifier	SU-17
Test Purpose	The test shall verify that there will be an error, if the request contains an unknown NEW_ArchiveData_Type (not XAIP and not BIN) and the archive data object will not be imported.
Configuration	CONFIG_Implicit
Pre-test conditions	Make sure that the middleware allows allows to retrieve archive data and to submit archive data with AOIDs specified by the client application. The usage of the verificationReport-element is optional. Authenticated connection to middleware exists.

Step	**Test sequence**	**Expected Results**
1	Repeat this step until the expected result has been achieved or skip the test case as failed after ten trials. Randomly generate AOID_SU-17 within the possible range and make sure that it has not been assigned yet. This is realised by the following call: ArchiveRetrievalRequest (AOID_SU-17)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
2	ArchiveSubmissionRequest (dss:OptionalInputs(AOID(AOID_SU-17), ReturnVerificatonReport), New_ArchiveData_Type)	ArchiveSubmissionResponse (dss:Result(resultmajor#error, resultminor/arl/unknownArchiveDataType), dss:OptionalOutput(VerificationReport))
3	ArchiveRetrievalRequest (AOID_SU-17)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
Observations:		
Verdict:		

[15] **NOTE:** The client application may generate an AOID. Therefore the <AOID>-element is optional. See also Annex F (Version 1.2 , Chapter 3.2, page 10).

3.4 ArchiveUpdateRequest

All test cases defined in this section are derived from the requirement A4.2-3 and the general requirement A2.0-2 together with the interface specification of the ArchiveUpdateRequest function in section 3.2 of [TR-ESOR-E]. According to section 2 of [TR-ESOR-E], ArchiveUpdateRequest and ArchiveUpdateResponse **should** be supported.
If ArchiveUpdateRequest and ArchiveUpdateResponse are not supported, the following test cases are obsolete, otherwise they have to be fulfilled.

3.4.1 UP-01 – DXAIP_OK

Identifier	UP-01
Test Purpose	The test shall verify that it is possible to submit a correct DXAIP_OK with ArchiveUpdateRequest.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that test case SU-01 was successfully performed with AOID_SU-01. Requirements (A3.2.1-1) and (A3.2.2-1): The usage of the dss:OptionalInputs “ImportEvidence” in order to import an evidence record is optional. The usage of the verificationReport-element is optional.

Step	**Test sequence**	**Expected Results**
1	Insert AOID_SU-01 into the prepared DXAIP_OK-template.	
2	ArchiveUpdateRequest (DXAIP_OK)	ArchiveUpdateResponse (dss:Result(resultmajor#OK), VersionID(VID_UP-01))
3	Insert AOID_SU-04 into the prepared DXAIP_OK-template.	
4	ArchiveUpdateRequest (dss:OptionalInputs(ReturnVerificatonReport, ImportEvidence(ER_OK)), DXAIP_OK)	ArchiveUpdateResponse (dss:Result(resultmajor#OK), dss:OptionalOutput(VerificationReport), VersionID(VID_UP-01(AOID_SU-04)))
Observations:		
Verdict:		

3.4.2 UP-02 – DXAIP_NOK_AOID

Identifier	UP-02
Test Purpose	The test shall verify that there will be an error, if a DXAIP_NOK_AOID with a not yet assigned AOID is submitted.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. The usage of the verificationReport-element is optional.

Step	Test sequence	Expected Results
1	Repeat this step until the expected result has been achieved or skip the test case as failed after ten trials. Randomly generate AOID_UP-02 within the possible range and make sure that it has not been assigned yet. This is realised by the following call: ArchiveRetrievalRequest (AOID_UP-02)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
2	Insert AOID_UP-02 into DXAIP_NOK_AOID.	
3	ArchiveUpdateRequest (dss:OptionalInputs(ReturnVerificatonReport)), DXAIP_NOK_AOID)	ArchiveUpdateResponse (dss:Result(resultmajor#error, resultminor/arl/DXAIP_NOK_AOID), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

3.4.3 UP-03 – DXAIP_NOK

Identifier	UP-03	
Test Purpose	The test shall verify that there will be an error, if a (syntactically incorrect) DXAIP_NOK is submitted.	
Configuration	CONFIG_Implicit	
	Authenticated connection to middleware exists. The usage of the verificationReport-element is optional.	
Step	**Test sequence**	**Expected Results**
1	ArchiveUpdateRequest (dss:OptionalInputs(ReturnVerificationReport, DXAIP_NOK))	ArchiveUpdateResponse (dss:Result(resultmajor#error, resultminor/arl/DXAIP_NOK), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

3.4.4 UP-04 – DXAIP_NOK_EXPIRED

Identifier	UP-04	
Test Purpose	The test shall verify that there will be an error, if the invalid DXAIP_NOK_EXPIRED is submitted.	
Configuration	CONFIG_Implicit	
Pre-test conditions	Authenticated connection to middleware exists. The usage of the verificationReport-element is optional.	
Step	**Test sequence**	**Expected Results**
1	ArchiveUpdateRequest (dss:OptionalInputs(ReturnVerificatonReport)), DXAIP_NOK_EXPIRED)	ArchiveUpdateResponse (dss:Result(resultmajor#error, resultminor/arl/DXAIP_NOK_EXPIRED), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

3.4.5 UP-05 – XAIP_NOK_SUBMTIME

Identifier	UP-05
Test Purpose	The test shall verify that there will be an error, if the invalid XAIP_NOK_SUBMTIME is submitted.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. The usage of the verificationReport-element is optional.

Step	Test sequence	Expected Results
1	ArchiveUpdateRequest (dss:OptionalInputs(ReturnVerificatonReport)), XAIP_NOK_SUBMTIME)	ArchiveUpdateResponse (dss:Result(resultmajor#error, resultminor/arl/DXAIP_NOK_SUBMTIME), dss:OptionalOutput(VerificationReport))

Observations:

Verdict:

3.4.6 UP-06 – DXAIP_NOK_SIG

Identifier	UP-06	
Test Purpose	The test shall verify that there will be an error, if the submitted DXAIP_NOK_SIG contains signatures, which do not verify correctly[16].	
Configuration	CONFIG_Implicit	
Pre-test conditions	Authenticated connection to middleware exists. The usage of the verificationReport-element is optional.	
Step	**Test sequence**	**Expected Results**
1	ArchiveUpdateRequest (dss:OptionalInputs(ReturnVerificatonReport)), DXAIP_NOK_SIG)	ArchiveUpdateResponse (dss:Result(resultmajor#error, resultminor/arl/DXAIP_NOK_SIG), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

[16] See Annex E (Version 1.2, Chapter 3.7.1, (`VerifyUnderSignaturePolicy`)

3.4.7 UP-07 – DXAIP_NOK_ER

Identifier	UP-07	
Test Purpose	The test shall verify that there will be an error, because the submitted DXAIP_NOK_ER contains an invalid Evidence Record.	
Configuration	CONFIG_Implicit	
Pre-test conditions	Authenticated connection to middleware exists. The usage of the verificationReport-element is optional.	
Step	**Test sequence**	**Expected Results**
1	ArchiveUpdateRequest (dss:OptionalInputs(ReturnVerificatonReport)), DXAIP_NOK_ER)	ArchiveUpdateResponse (dss:Result(resultmajor#error, resultminor/arl/DXAIP_NOK_ER), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

3.4.8 UP-08 – DXAIP_NOK_VERSION

Identifier	UP-08	
Test Purpose	The test shall verify that there will be an error, if a DXAIP_NOK_VERSION is submitted with ArchiveUpdateRequest, which produces a collision[17] with the already existing XAIP.	
Configuration	CONFIG_Implicit	
Pre-test conditions	Authenticated connection to middleware exists. The usage of the verificationReport-element is optional.	
Step	**Test sequence**	**Expected Results**
1	ArchiveUpdateRequest (dss:OptionalInputs(ReturnVerificatonReport)), DXAIP_NOK_VERSION)	ArchiveUpdateResponse (dss:Result(resultmajor#error, resultminor/arl/DXAIP_NOK_VERSION), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

[17] A wrong „version“ (DXAIP_NOK_Version) may be caused by inserting a DXAIP where the value in the element `prevVersion` in the `updateSection` of the DXAIP is not the latest version of this XAIP.

3.4.9 UP-09 – Update of a previously submitted BIN

Identifier	UP-09	
Test Purpose	The test shall verify that it is possible to submit a correct DXAIP_OK with ArchiveUpdateRequest based on a previously inserted BIN-archive data object.	
Configuration	CONFIG_Implicit	
Pre-test conditions	Authenticated connection to middleware exists. The usage of the verificationReport-element is optional.	
Step	**Test sequence**	**Expected Results**
1	ArchiveRetrievalRequest (AOID_SU-11)	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK(BIN, AOID_SU-11, VID_SU-11))
2	Insert AOID_SU-11 into the prepared DXAIP_OK-template	
3	ArchiveUpdateRequest (dss:OptionalInputs(ReturnVerificatonReport)), DXAIP_OK)	ArchiveUpdateResponse (dss:Result(resultmajor#OK), dss:OptionalOutput(VerificationReport), VersionID(VID_UP-09))
Observations:		
Verdict:		

3.4.10 UP-10 – DXAIP_NOK_ID

Identifier	UP-10
Test Purpose	The test shall verify that there will be a wrong ID in DXAIP_NOK_ID, when the invalid DXAIP_NOK_ID is submitted.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. The usage of the `verificationReport`-element is optional.

Step	**Test sequence**	**Expected Results**
1	Repeat this step until the expected result has been achieved or skip the test case as failed after ten trials. Randomly generate AOID_UP-10 within the possible range and make sure that it has not been assigned yet. This is realised by the following call: ArchiveRetrievalRequest (AOID_UP-10)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
2	Insert AOID_UP-10 into the prepared DXAIP_OK-template.	
3	ArchiveUpdateRequest (dss:OptionalInputs(ReturnVerificatonReport)), DXAIP_NOK_ID)	ArchiveUpdateResponse (dss:Result(resultmajor#error, resultminor/arl/DXAIP_NOK_ID), dss:OptionalOutput(VerificationReport))
Observations:		
Verdict:		

3.4.11 UP-11 – Unknown control in `OptionalInputs`

Identifier	UP-11
Test Purpose	The test shall verify that there will be an error, if the request contains unknown controls in the `OptionalInputs`-element.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. The usage of the verificationReport-element is optional.

Step	Test sequence	Expected Results
1	ArchiveRetrievalRequest (AOID_SU-02)	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK(VID_SU-02)[18])
2	Insert AOID_SU-02 into the prepared DXAIP_OK-template.	
3	ArchiveUpdateRequest (dss:OptionalInputs(SomethingUnknown, ReturnVerificatonReport), DXAIP_OK)	ArchiveUpdateResponse (dss:Result(resultmajor#error, resultminor/arl/notSupported), dss:OptionalOutput(VerificationReport))
4	ArchiveRetrievalRequest (AOID_SU-02)	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK(VID_SU-02)[19])
Observations:		
Verdict:		

[18] After the successful ArchiveSubmissionRequest in test case UP-02 (see section 3.3.2), retrieving AOID_SU-02 yields the latest version of the respective XAIP, which is "VID_SU-02".

[19] After the successful ArchiveSubmissionRequest in test case UP-02 (see section 3.3.2), and the not successful ArchiveUpdateRequest retrieving AOID_SU-02 yields the latest version of the respective XAIP, which is still "VID_SU-02".

3.5 ArchiveRetrievalRequest

All test cases defined in this section are derived from the general requirement A2.0-2 together with the interface specification of the ArchiveRetrievalRequest function in section 3.3 of [TR-ESOR-E].

3.5.1 RE-01 – Retrieval of previously archived XAIPs

Identifier	RE-01
Test Purpose	The test shall verify that it is possible to retrieve previously archived XAIPs .
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that the test cases SU-01, SU-02, SU-10 and UP-01 were successfully performed. Requirements (A3.3.1-1) and (A3.3.1-2); The usage of the dss:OptionalInputs "IncludeERS" in order to retrieve an evidence record is optional.

Step	Test sequence	Expected Results
1	ArchiveRetrievalRequest (AOID_SU-01[20])	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK+DXAIP_OK(VID_UP-01)[21] [22])
2	ArchiveRetrievalRequest (dss:OptionalInputs(IncludeERS), AOID_SU-02)	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK(ER_OK)[23])
3	The fulfilment of the requirement A3.3.1-2[24] is checked.	
4	ArchiveRetrievalRequest (AOID_SU-10)	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK_SIG_OK_ER)
5	ArchiveRetrievalRequest (AOID_SU-01, VID_UP-01)	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK+DXAIP_OK[25]

[20] AOID_SU-01 denotes the AOID, which has been returned within test case SU-01 (see Section 3.3.1, page 13).

[21] After the successful `ArchiveUpdateRequest` in test case UP-01 (see section 3.4.1), retrieving AOID_SU-01 yields the latest version of the respective XAIP, which has been created by merging XAIP_OK and DXAIP_OK.

[22] VID_UP-01 denotes the `VersionID`, which has been returned within test case UP-01 (see Section 3.4.1, page 32).

[23] After the successful `ArchiveSubmissionRequest` in test case UP-02 (see Section 3.3.2), retrieving AOID_SU-02 yields the latest version of the respective XAIP, which is "VID_SU-02" and the evidence record(s) is(are) included in the credential section.

[24] See Annex E (Section 3.3.1).

[25] After the successful `ArchiveUpdateRequest` in test case UP-01 (see section 3.4.1), retrieving AOID_SU-01 with VID_UP-01 yields the XAIP, which has been created by merging XAIP_OK and

		)
6	ArchiveRetrievalRequest (AOID_SU-01, VID_SU-01[26])	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK(VID_SU-01)[27])
7	ArchiveRetrievalRequest (AOID_SU-01, ALL[28])	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK(VID_SU-01, VID_UP-01)[29])
8	ArchiveRetrievalRequest (dss:OptionalInputs(IncludeERS), AOID_SU-10[30])	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK_SIG_OK_ER)
9	ArchiveRetrievalRequest (dss:OptionalInputs(IncludeERS), AOID_SU-02)	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK_SIG_OK_ER(AOID_SU-02)[31])
Observations:		
Verdict:		

DXAIP_OK.

[26] VID_SU-01 denotes the `VersionID` of XAIP_OK with AOID_SU-01 after the successful `ArchiveSubmissionRequest` in test case SU-01 (see section 3.3.1 page 13).

[27] Before the successful `ArchiveUpdateRequest` in test case UP-01 (see section 3.4.1), retrievingAOID_SU-01 with VID_SU-01 yields the XAIP_OK, which was submitted in test case SU-01 (see section 3.3.1) .

[28] "ALL" denotes that all existing versions of XAIP_OK have to be retrieved in the ArchiveRetrievalResponse (see Annex -E, Section 3.3.1, page 18).

[29] XAIP_OK contains two versions VID_SU-01 and VID_UP-01, which are retrieved by the `ArchiveRetrievalResponse`.

[30] "IncludeERS" denotes that the retrieved XAIP_OK_SIG_OK_ER has to contain the Evidence Record(s) according to the retrieved versions of the XAIP_OK (see Annex E, section 3.3.1, page 18).

[31] After the sucessful ArchiveRetrievalRequest the returned XAIP_OK_SIG_OR_ER includes the requested EvidenceRecord(s) .

3.5.2 RE-02 – Retrieval of previously archived BINs

Identifier	RE-02	
Test Purpose	The test shall verify that it is possible to retrieve a XAIP after a successful ArchiveRetrievalRequest concerning an archive data object, which was previously inserted as a BIN in the long-term storage by an ArchiveSubmissionRequest.	
Configuration	CONFIG_Implicit	
Pre-test conditions	Authenticated connection to middleware exists. Make sure that the test cases SU-11, SU-12, UP-09 were successfully performed.	
Step	**Test sequence**	**Expected Results**
1	ArchiveRetrievalRequest (AOID_SU-11[32])	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK+DXAIP_OK(VID_UP-09)[33] [34])
2	ArchiveRetrievalRequest (AOID_SU-11, VID_SU-11[35])	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK(VID_SU-11)[36])
3	ArchiveRetrievalRequest (AOID_SU-11, ALL[37])	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK(VID_SU-11, VID_UP-09)[38])
Observations:		
Verdict:		

[32] See test case SU-11, section 3.3.11.

[33] After sucessfully importing BIN in test case 3.3.11 and the successful ArchiveUpdateRequest in test case UP-09 (see section 3.4.9), retrieving AOID_SU-11 yields the latest version of the respective XAIP.

[34] VID_SU-11 denotes the VersionID, which has been returned within test case UP-11 (see section 3.3.11, page 25).

[35] VID_SU-11 denotes the VersionID of XAIP_OK with AOID_SU-11 after the sucessful ArchiveSubmissionRequest in test case SU-11 (see section 3.3.11, page 25).

[36] Before the successful ArchiveUpdateRequest in test case UP-09 (see section 3.4.9), retrievingAOID_SU-11 with VID_SU-11 yields the XAIP_OK, which was imported in test case SU-11 (see section 3.3.11) .

[37] "ALL" denotes that all existing versions of XAIP_OK have to be retrieved in the ArchiveRetrievalResponse (see [TR-ESOR-E], section 3.3.1, page 18).

[38] XAIP_OK contains two versions VID_SU-11 and VID_UP-09, which are retrieved by the ArchiveRetrievalResponse.

3.5.3 RE-03 – Retrieval of XAIPs for known and unknown AOIDs

Identifier	RE-03
Test Purpose	The test shall verify that it is possible to retrieve XAIPs for known AOIDs and there will be an error in case of unknown AOIDs.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that the test cases SU-01 and SU-10 were successfully performed.

Step	**Test sequence**	**Expected Results**
1	Repeat this step until the expected result has been achieved or skip the test case as failed after ten trials. Randomly generate AOID_RE-03 within the possible range and make sure that it has not been assigned yet. This is realised by the following call: ArchiveRetrievalRequest (AOID_RE-03)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
2	ArchiveRetrievalRequest (AOID_SU-01)	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK+DXAIP_OK[39])
3	ArchiveRetrievalRequest (AOID_SU-10)	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK_SIG_OK_ER)
4	ArchiveRetrievalRequest (AOID_RE-03)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
Observations:		
Verdict:		

[39] After the successful ArchiveUpdateRequest in test case UP-01 (see section 3.4.1), retrieving AOID_SU-01 yields the latest version of the respective XAIP, which has been created by merging XAIP_OK and DXAIP_OK.

3.5.4 RE-04 – Retrieval of XAIPs for known and unknown VersionIDs

Identifier	RE-04
Test Purpose	The test shall verify that there will be a requestOnlyPartlySuccessfulWarning, if it is not possible to retrieve XAIPs for all indicated VersionIDs.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that the test case SU-01 was successful performed.

Step	Test sequence	Expected Results
1	Repeat this step until the expected result has been achieved or skip the test case as failed after ten trials. Randomly generate VID_RE-04 within the possible range and make sure that it has not been assigned yet. This is realised by the following call: ArchiveRetrievalRequest (AOID_SU-01, VID_RE-04)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownVersionID))
2	ArchiveRetrievalRequest (AOID_SU-01, VID_UP-01)	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK+DXAIP_OK(VID_UP-01))
3	ArchiveRetrievalRequest (AOID_SU-01, VID_RE-04)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownVersionID))
4	ArchiveRetrievalRequest (AOID_SU-01, VID_UP-01, VID_RE-04)	ArchiveRetrievalResponse (dss:Result(resultmajor#warning, resultminor/arl/requestOnlyPartlySuccessfulWarning), XAIP_OK+DXAIP_OK(VID_UP-01))
Observations:		
Verdict:		

3.5.5 RE-05 – Unknown control in `OptionalInputs`

Identifier	RE-05	
Test Purpose	The test shall verify that there will be an error, if the request contains unknown controls in the `OptionalInputs`-element.	
Configuration	CONFIG_Implicit	
Pre-test conditions	Authenticated connection to middleware exists. Make sure that test case SU-01 was successfully performed.	
Step	**Test sequence**	**Expected Results**
1	ArchiveRetrievalRequest (dss:OptionalInputs(SomethingUnknown), AOID_SU-01)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/notSupported))
Observations:		
Verdict:		

3.6 ArchiveEvidenceRequest

All test cases defined in this section are derived from the general requirement A2.0-2 together with the interface specification of the ArchiveEvidenceRequest function in section 3.4 of [TR-ESOR-E].

3.6.1 EV-01 – Retrieval of Evidence Records of previously archived XAIPs without specifying the desired ERS Format

Identifier	EV-01
Test Purpose	The test shall verify that it is possible to retrieve Evidence Records for previously archived XAIPs.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that test cases SU-01, SU-02, SU-10 and UP-01 were successfully performed.

Step	Test sequence	Expected Results
1	ArchiveEvidenceRequest (AOID_SU-01)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-1,VID_UP-01, ASN.1-ERS(XAIP_OK+DXAIP_OK))[40])
2	ArchiveEvidenceRequest (AOID_SU-02)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-2,VID_SU-02, ASN.1-ERS(XAIP_OK)))
3	ArchiveEvidenceRequest (AOID_SU-10)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-10,VID_SU-10, ASN.1-ERS(XAIP_OK_SIG_OK_ER)))
4	ArchiveEvidenceRequest (AOID_SU-01, v1)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-01,v1[41], ASN.1-ERS(XAIP_OK)))
5	ArchiveEvidenceRequest (AOID_SU-01, VID_UP-01)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-01,VID_UP-01, ASN.1-ERS(XAIP_OK+DXAIP_OK)))

[40] Latest version
[41] VID_SU-01:=v1

6	ArchiveEvidenceRequest (AOID_SU-01, VID_UP-01, VID_SU-01)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-01,VID_UP-01, ASN.1-ERS(XAIP_OK+DXAIP_OK)), xaipERS(AOID_SU-01,VID_SU-01, ASN.1-ERS(XAIP_OK)))
7	ArchiveEvidenceRequest (AOID_SU-01, ALL)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-01,VID_SU-01, ASN.1-ERS(XAIP_OK)), xaipERS(AOID_SU-01,VID_UP-01, ASN.1-ERS(XAIP_OK+DXAIP_OK)))
Observations:		
Verdict:		

3.6.2 EV-02 – Retrieval of Evidence Records of previously archived BINs without specifying the desired ERS Format

Identifier	EV-02
Test Purpose	The test shall verify that it is possible to retrieve Evidence Records for previously archived BINs.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that test cases SU-11, SU-12 and UP-09 were successfully performed.

Step	Test sequence	Expected Results
1	ArchiveEvidenceRequest (AOID_SU-11)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-11,VID_UP-09, ASN.1-ERS(XAIP(BIN)_OK+DXAIP_OK))[42])
2	ArchiveEvidenceRequest (AOID_SU-12)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-12,VID_SU-12, ASN.1-ERS(XAIP(BIN)_OK)))
3	ArchiveEvidenceRequest (AOID_SU-11, v1)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-11,v1[43], ASN.1-ERS(XAIP(BIN)_OK)))
4	ArchiveEvidenceRequest (AOID_SU-11, VID_UP-09)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-11,VID_UP-09, ASN.1-ERS(XAIP(BIN)_OK+DXAIP_OK)))
5	ArchiveEvidenceRequest (AOID_SU-11, VID_UP-09, VID_SU-11)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-11,VID_UP-09, ASN.1-ERS(XAIP(BIN)_OK+DXAIP_OK)), xaipERS(AOID_SU-11,VID_SU-11, ASN.1-ERS(XAIP_OK)))
6	ArchiveEvidenceRequest (AOID_SU-11, ALL)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-11,VID_SU-11, ASN.1-ERS(XAIP(BIN)_OK)), xaipERS(AOID_SU-11,VID_UP-091, ASN.1-ERS(XAIP(BIN)_OK+DXAIP_OK)))
Observations:		
Verdict:		

[42] Latest version
[43] VID_SU-01:=v1

3.6.3 EV-03 – Retrieval of ASN.1-based Evidence Records for previously archived XAIPs specifying the desired ERS Format

Identifier	EV-03
Test Purpose	The test shall verify that it is possible to retrieve ASN.1-based Evidence Records for previously archived XAIPs.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that test cases SU-01, SU-02, SU-10 and UP-01 were successfully performed. The usage of the dss:OptionalInputs "ERSFormat" is optional. Requirements (A3.4.1-1) and (A3.4.2-1): RFC6283 as an option is not supported .

Step	Test sequence	Expected Results
1	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:4998)), AOID_SU-01)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-1,VID_UP-01, ASN.1-ERS(XAIP_OK+DXAIP_OK))[44])
2	Check ASN.1-ERS(XAIP_OK+DXAIP_OK) according to (A3.4.2-1)[45].	
3	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:4998)), AOID_SU-02)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-2,VID_SU-02, ASN.1-ERS(XAIP_OK)))
4	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:4998)), AOID_SU-10)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-10,VID_SU-10, ASN.1-ERS(XAIP_OK_SIG_OK_ER)))
5	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:4998)), AOID_SU-01, v1)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-01,v1[46], ASN.1-ERS(XAIP_OK)))
6	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:4998)), AOID_SU-01, VID_UP-01)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-01,VID_UP-01, ASN.1-ERS(XAIP_OK+DXAIP_OK)))
7	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(	ArchiveEvidenceResponse (dss:Result(resultmajor#OK),

[44] actual version
[45] See Annex E (Section 3.4.2).
[46] VID_SU-01:=v1

	urn:ietf:rfc:4998)), AOID_SU-01, VID_UP-01, VID_SU-01)	xaipERS(AOID_SU-01,VID_UP-01, ASN.1-ERS(XAIP_OK+DXAIP_OK)), xaipERS(AOID_SU-01,VID_SU-01, ASN.1-ERS(XAIP_OK)))
8	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:4998)), AOID_SU-01, ALL)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-01,VID_SU-01, ASN.1-ERS(XAIP_OK)), xaipERS(AOID_SU-01,VID_UP-01, ASN.1-ERS(XAIP_OK+DXAIP_OK)))
9	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:6283)), AOID_SU-01, ALL)	ArchiveEvidenceResponse (dss:Result(resultmajor#error, resultminor/arl/notSupported)),
Observations:		
Verdict:		

3.6.4 EV-04 – Retrieval of ASN.1-based Evidence Records for previously archived BINs specifying the desired ERS Format

Identifier	EV-04
Test Purpose	The test shall verify that it is possible to retrieve ASN.1-based Evidence Records for previously archived BINs.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that test cases SU-11, SU-12 and UP-09 were successfully performed. The usage of the dss:OptionalInputs "ERSFormat" is optional.

Step	Test sequence	Expected Results
1	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:4998)), AOID_SU-11)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-11,VID_UP-09, ASN.1-ERS(XAIP(BIN)_OK+DXAIP_OK)))
2	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:4998)), AOID_SU-12)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-12,VID_SU-12, ASN.1-ERS(XAIP(BIN)_OK)))
3	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:4998)), AOID_SU-11, v1)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-11,v1[47], ASN.1-ERS(XAIP(BIN)_OK)))
4	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:4998)), AOID_SU-11, VID_UP-09)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-11,VID_UP-09, ASN.1-ERS(XAIP(BIN)_OK+DXAIP_OK)))
5	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:4998)), AOID_SU-11, VID_UP-09, VID_SU-11)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-11,VID_UP-09, ASN.1-ERS(XAIP(BIN)_OK+DXAIP_OK)), xaipERS(AOID_SU-11,VID_SU-11, ASN.1-ERS(XAIP(BIN)_OK)))

[47] VID_SU-01:=v1

6	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:4998)), AOID_SU-11, ALL)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-11,VID_SU-11, ASN.1-ERS(XAIP(BIN)_OK)), xaipERS(AOID_SU-11,VID_UP-091, ASN.1-ERS(XAIP(BIN)_OK+DXAIP_OK)))
Observations:		
Verdict:		

3.6.5 EV-05 – Retrieval of XML-based Evidence Records for previously archived XAIPs specifying the desired ERS Format

This test is only applicable, if the middleware supports XML-based Evidence Records.

Identifier	EV-05
Test Purpose	The test shall verify that it is possible to retrieve XML-based Evidence Records for previously archived XAIPs.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that test cases SU-01, SU-02, SU-10 and UP-01 were successfully performed. The usage of the dss:OptionalInputs "ERSFormat" is supported. RFC6283 as an option is supported .

Step	Test sequence	Expected Results
1	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:6283)), AOID_SU-01)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-1,VID_UP-01, XML-ERS(XAIP_OK+DXAIP_OK))[48])
2	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:6283)), AOID_SU-02)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-2,VID_SU-02, XML-ERS(XAIP_OK)))
3	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:6283)), AOID_SU-10)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-10,VID_SU-10, XML- ERS(XAIP_OK_SIG_OK_ER)))
4	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:6283)), AOID_SU-01, v1)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-01,v1[49], XML-ERS(XAIP_OK)))
5	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:6283)), AOID_SU-01, VID_UP-01)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-01,VID_UP-01, XML-ERS(XAIP_OK+DXAIP_OK)))
6	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(	ArchiveEvidenceResponse (dss:Result(resultmajor#OK),

[48] actual version
[49] VID_SU-01:=v1

	urn:ietf:rfc:6283)), AOID_SU-01, VID_UP-01, VID_SU-01)	xaipERS(AOID_SU-01,VID_UP-01, XML--ERS(XAIP_OK+DXAIP_OK)), xaipERS(AOID_SU-01,VID_SU-01, XML-ERS(XAIP_OK)))
7	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:6283)), AOID_SU-01, ALL)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-01,VID_SU-01, XML-ERS(XAIP_OK)), xaipERS(AOID_SU-01,VID_UP-01, XML-ERS(XAIP_OK+DXAIP_OK)))
Observations:		
Verdict:		

3.6.6 EV-06 – Retrieval of XML-based Evidence Records for previously archived BINs specifying the desired ERS Format

This test is only applicable, if the middleware supports XML-based Evidence Records.

Identifier	EV-06
Test Purpose	The test shall verify that it is possible to retrieve XML-based Evidence Records for previously archived BINs.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that test cases SU-11, SU-12 and UP-09 were successfully performed. The usage of the dss:OptionalInputs "ERSFormat" is supported. RFC6283 as an option is supported.

Step	Test sequence	Expected Results
1	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:6283)), AOID_SU-11)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-11,VID_UP-09, XML-ERS(XAIP(BIN)_OK+DXAIP_OK)))
2	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:6283)), AOID_SU-12)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-12,VID_SU-12, XML-ERS(XAIP(BIN)_OK)))
3	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:6283)), AOID_SU-11, v1)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-11,v1[50], XML-ERS(XAIP(BIN)_OK)))
4	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:6283)), AOID_SU-11, VID_UP-09)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-11,VID_UP-09, XML-ERS(XAIP(BIN)_OK+DXAIP_OK)))
5	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:6283)), AOID_SU-11, VID_UP-09, VID_SU-11	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-11,VID_UP-09, XML-ERS(XAIP(BIN)_OK+DXAIP_OK)), xaipERS(AOID_SU-11,VID_SU-11,

[50] VID_SU-01:=v1

	) 	XML-ERS(XAIP(BIN)_OK)))
6	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:6283)), AOID_SU-11, ALL)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-11,VID_SU-11, XML-ERS(XAIP(BIN)_OK)), xaipERS(AOID_SU-11,VID_UP-091, XML-ERS(XAIP(BIN)_OK+DXAIP_OK)))
Observations:		
Verdict:		

3.6.7 EV-07 – Retrieval of Evidence Records in an unknown Evidence Record-Format (ERSFormat) for previously archived XAIPs

Identifier	EV-07
Test Purpose	The test shall verify that there will be an error in case of an unknown ERSFormat.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that the test case SU-01 and SU-10 was successfully performed. The usage of the dss:OptionalInputs “ERSFormat” is supported.

Step	**Test sequence**	**Expected Results**
1	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:5000)), AOID_SU-01)	ArchiveEvidenceResponse (dss:Result(resultmajor#error), resultminor/arl/notSupported))
2	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:5000)), AOID_SU-10)	ArchiveEvidenceResponse (dss:Result(resultmajor#error), resultminor/arl/notSupported))
Observations:		
Verdict:		

3.6.8 EV-08 – Retrieval of Evidence Records in an unknown ERSFormat for previously archived BINs

Identifier	EV-08
Test Purpose	The test shall verify that there will be an error in case of an unknown ERSFormat.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that the test case SU-11 and SU-12 was successfully performed.

Step	**Test sequence**	**Expected Results**
1	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:5000)), AOID_SU-11,)	ArchiveEvidenceResponse (dss:Result(resultmajor#error), resultminor/arl/notSupported))
2	ArchiveEvidenceRequest (dss:OptionalInputs(ERSFormat(urn:ietf:rfc:5000)), AOID_SU-12)	ArchiveEvidenceResponse (dss:Result(resultmajor#error), resultminor/arl/notSupported))
Observations:		
Verdict:		

3.6.9 EV-09 – Retrieval of Evidence Records for known and unknown AOIDs

Identifier	EV-09
Test Purpose	The test case verifies that that it is possible to retrieve an EvidenceRecord for known AOIDs and there will be an error in case of unknown AOIDs.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that the test cases SU-01 and SU-02, were successfully performed. Make sure, that the randomly generated AOID_RE-02 is within the possible range but has not been assigned yet.

Step	**Test sequence**	**Expected Results**
1	ArchiveEvidenceRequest (AOID_SU-01)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-01, VID_UP-01, ASN.1-ERS(XAIP_OK+DXAIP_OK))[51])
2	ArchiveEvidenceRequest (AOID_SU-10)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-10,VID_SU-10, ASN.1-ERS(XAIP_OK_SIG_OK_ER)))
3	ArchiveEvidenceRequest (AOID_RE-03)	ArchiveEvidenceResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
Observations:		
Verdict:		

[51] actual version

3.6.10 EV-10 – Retrieval of Evidence Records for partly known VersionIDs

Identifier	EV-10
Test Purpose	The test case verifies that there is a requestOnlyPartlySuccessfulWarning, if an Evidence Record for an unknown VersionID is requested.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that the test case SU-01 was successfully performed. Make sure, that conderning AOID_SU-01 VID_RE-03 is within the possible range but is an unknown VersionID.

Step	**Test sequence**	**Expected Results**
1	ArchiveEvidenceRequest (AOID_SU-01)	ArchiveEvidenceResponse (dss:Result(resultmajor#OK), xaipERS(AOID_SU-01, VID_UP-01, ASN.1-ERS(XAIP_OK+DXAIP_OK)))
2	ArchiveEvidenceRequest (AOID_SU-01, VID_RE-03)	ArchiveEvidenceResponse (dss:Result(resultmajor#error, resultminor/arl/unknownVersionID))
3	ArchiveEvidenceRequest (AOID_SU-01, VID_RE-03, VID_UP-01)	ArchiveEvidenceResponse (dss:Result(resultmajor#warning, resultminor/arl/requestOnlyPartlySucessfulWarning), xaipERS(AOID_SU-01, VID_UP-01, ASN.1-ERS(XAIP_OK+DXAIP_OK)))
Observations:		
Verdict:		

3.6.11 EV-11 – Unknown control in `OptionalInputs`

Identifier	EV-11	
Test Purpose	The test shall verify that there will be an error, if the request contains unknown controls in the `OptionalInputs`-element.	
Configuration	CONFIG_Implicit	
Pre-test conditions	Authenticated connection to middleware exists. Make sure that the test case SU-01 was successfully performed.	
Step	**Test sequence**	**Expected Results**
1	ArchiveEvidenceRequest (dss:OptionalInputs(SomethingUnknown), AOID_SU-01)	ArchiveEvidenceResponse (dss:Result(resultmajor#error, resultminor/arl/notSupported)
Observations:		
Verdict:		

3.7 ArchiveDeletionRequest

All test cases defined in this section are derived from the general requirement A2.0-2 together with the interface specification of the ArchiveDeletionRequest function in section 3.5 of [TR-ESOR-E].

3.7.1 DE-01 – Deletion of XAIP without ReasonOfDeletion

Identifier	DE-01
Test Purpose	The test shall verify that it will yield an error, if ArchiveDeletionRequest is called without providing a ReasonOfDeletion if the element "retentionPeriod" in the XAIP contains a predetermined future date.
Configuration	CONFIG_Implicit
Pre-test conditions	In one test case the element "retentionPeriod" în the XAIP to be deleted contains a predetermined future date. In the other test case the element "retentionPeriod" contains a date in the past. Authenticated connection to middleware exists. Requirement (A3.5.1-1): The ArchiveDeletionRequest of SU-10 with the end of the "retentionPeriod" in the future should not be successful without a reason for the deletion.

Step	**Test sequence**	**Expected Results**
1	ArchiveDeletionRequest (AOID_SU-10)	ArchiveDeletionResponse (dss:Result(resultmajor#error, resultminor/arl/missingReasonOfDeletion))
2	ArchiveRetrievalRequest (AOID_SU-10)	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK_SIG_OK_ER)
3	ArchiveDeletionRequest (AOID_SU-03[52])	ArchiveDeletionResponse (dss:Result(resultmajor#ok))
4	ArchiveRetrievalRequest (AOID_SU-03)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
Observations:		
Verdict:		

[52] The XAIP with AOID_SU-03 was imported some days ago. The retentionPeriod is finished.

3.7.2 DE-02 – Deletion of XAIP with ReasonOfDeletion

Identifier	DE-02
Test Purpose	The test shall verify that it is possible to delete an XAIP by calling ArchiveDeletionRequest with ReasonOfDeletion among the dss:OptionalInputs.
Configuration	CONFIG_Implicit
Pre-test conditions	The element “retentionPeriod” in the XAIP to be deleted contains a predetermined future date. Make sure that the test case SU-10 was successfully performed. Authenticated connection to middleware exists.

Step	Test sequence	Expected Results
1	ArchiveDeletionRequest (dss:OptionalInputs(ReasonOfDeletion(RequestorName(SomeName), RequestInfo(SomeInfo)), AOID_SU-10)	ArchiveDeletionResponse (dss:Result(resultmajor#ok))
2	ArchiveRetrievalRequest (AOID_SU-10)	ArchiveRetrievalResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
3	Check requirement A3.5.1-2[53].	
Observations:		
Verdict:		

[53] See Annex E (Section 3.5.1).

3.7.3 DE-03 – Deletion of unknown XAIPs without ReasonOfDeletion

Identifier	DE-03
Test Purpose	The test shall verify that it will yield an error, if ArchiveDeletionRequest is called without providing a ReasonOfDeletion. As the present call also contains the unknown AOID_RE-03, the AOID-specific dss:Result needs to indicate that the AOID is unknown.
Configuration	CONFIG_Implicit
Pre-test conditions	The element “retentionPeriod” in the XAIP to be deleted contains a predetermined future date. Make sure that the test case SU-02 was successfully performed. Make sure, that the randomly generated AOID_RE-02 is within the possible range but has not been assigned yet. Authenticated connection to middleware exists.

Step	Test sequence	Expected Results
1	ArchiveDeletionRequest (AOID_SU-02) 	ArchiveDeletionResponse (dss:Result(resultmajor#error, r resultminor/arl/missingReasonOfDeletion),,)
2	ArchiveDeletionRequest (AOID_RE-03)	ArchiveDeletionResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
3	ArchiveRetrievalRequest (AOID_SU-02)	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK)
Observations:		
Verdict:		

3.7.4 DE-04 – Deletion of unknown formerly BINs without ReasonOfDeletion

Identifier	DE-03
Test Purpose	The test shall verify that it will yield an error, if ArchiveDeletionRequest is called without providing a ReasonOfDeletion. As the present call also contains the unknown AOID_RE-03, the AOID-specific dss:Result needs to indicate that the AOID is unknown.
Configuration	CONFIG_Implicit
Pre-test conditions	The element “retentionPeriod” în the XAIP to be deleted, which was formerly integrated as a BIN, contains a predetermined future date. Make sure that the test case SU-11 was successfully performed. Make sure, that the randomly generated AOID_RE-02 is within the possible range but has not been assigned yet. Authenticated connection to middleware exists.

Step	**Test sequence**	**Expected Results**
1	ArchiveDeletionRequest (AOID_SU-11))	ArchiveDeletionResponse (dss:Result(resultmajor#error, resultminor/arl/missingReasonOfDeletion))
2	ArchiveDeletionRequest (AOID_RE-03)	ArchiveDeletionResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID))
3	ArchiveRetrievalRequest (AOID_SU-11)	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP(BIN)_OK)
Observations:		
Verdict:		

3.7.5 DE-05 – Deletion of unknown XAIPs with ReasonOfDeletion

Identifier	DE-05	
Test Purpose	The test shall verify that it will yield an error, if ArchiveDeletionRequest is called with an unknown AOIDs with providing a ReasonOfDeletion.	
Configuration	CONFIG_Implicit	
Pre-test conditions	The element “retentionPeriod” în the XAIP to be deleted contains a predetermined future date. Make sure, that the randomly generated AOID_RE-02 is within the possible range but has not been assigned yet. Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1	ArchiveDeletionRequest (dss:OptionalInputs(ReasonOfDeletion(RequestorName(SomeName), RequestInfo(SomeInfo))), AOID_RE-03)	ArchiveDeletionResponse (dss:Result(resultmajor#error, resultminor/arl/unknownAOID) AOID_RE-03)
Observations:		
Verdict:		

3.7.6 DE-06 – Unknown control in `OptionalInputs`

Identifier	DE-05	
Test Purpose	The test shall verify that there will be an error , if the request contains unknown controls in the `OptionalInputs`-element and the ArchiveDeletionRequest will not be successful.	
Configuration	CONFIG_Implicit	
Pre-test conditions	Authenticated connection to middleware exists. The element “retentionPeriod” in the XAIP to be deleted contains a predetermined future date. Make sure that the test case SU-02 was successfully performed.	
Step	**Test sequence**	**Expected Results**
1	ArchiveDeletionRequest (dss:OptionalInputs(SomethingUnknown), AOID_SU-02))	ArchiveDeletionResponse (dss:Result(resultmajor#error, resultminor/arl/notSupported))
2	ArchiveRetrievalRequest (AOID_SU-02))	 ArchiveRetrievalResponse (dss:Result(resultmajor#OK), XAIP_OK[54])
Observations:		
Verdict:		

[54] After the successful ArchiveSubmissionRequest in test case UP-02 (see section 3.3.2), retrieving AOID_SU-02 yields the latest version of the respective XAIP, which is “VID_SU-02”.

3.8 ArchiveDataRequest

All test cases defined in this section are derived from the general requirement A2.0-2 together with the interface specification of the ArchiveDataRequest function in section 3.6 of [TR-ESOR-E]. According to section 2 of [TR-ESOR-E], ArchiveDataRequest and ArchiveDataResponse **should** be supported.
If ArchiveDataRequest and ArchiveDataResponse are not supported, the following test cases are absolet, otherwise they have to be fulfilled.

3.8.1 Specification of DataLocation elements

The following test cases require the specification of DataLocation elements.

ID	Type and Purpose
DataLoc-P-01	http://www.w3.org/TR/2007/REC-xpath20-20070123/ All existing VersionManifest elements shall be returned.
DataLoc-P-02	http://www.w3.org/TR/2007/REC-xpath20-20070123/ The entire packageHeader element shall be returned.
DataLoc-P-NOK	http://www.w3.org/TR/2007/REC-xpath20-20070123/ Any syntactically invalid XPath expression.
DataLoc-Q-01	http://www.w3.org/TR/2007/REC-xquery-20070123/ All protected data objects of the first version of the XAIP shall be returned.
DataLoc-Q-02	http://www.w3.org/TR/2007/REC-xquery-20070123/ All unprotected data objects of the latest version of the XAIP shall be returned.
DataLoc-Q-NOK	http://www.w3.org/TR/2007/REC-xquery-20070123/ Any syntactically incorrect XQuery expression.
DataLoc-L-01	http://www.w3.org/TR/2003/REC-xptr-framework-20030325[55]
DataLoc-NOK	http://www.unknown.uri.org

[55] **NOTE:** The support of the XPointer-Language will be removed in the next version of the specification and hence there are no corresponding test cases specified.

3.8.2 DA-01 – Requesting Data within known XAIPs

Identifier	DA-01
Test Purpose	The test shall verify that it is possible to request certain pieces of data within a sequence of XAIPs identified by known AOIDs.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that test cases SU-01 was successfully performed.

Step	Test sequence	Expected Results
1	ArchiveDataRequest (AOID_SU-01, DataLocation(DataLoc-P-01), DataLocation(DataLoc-Q-01))	ArchiveDataResponse (dss:Result(resultmajor#ok), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-P-01), Value(DataLoc-P-01)), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-Q-01), Value(DataLoc-Q-01)))
2	ArchiveDataRequest (AOID_SU-01, DataLocation(DataLoc-P-02), DataLocation(DataLoc-Q-02))	ArchiveDataResponse (dss:Result(resultmajor#ok), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-P-02), Value(DataLoc-P-02)), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-Q-02), Value(DataLoc-Q-02)))
Observations:		
Verdict:		

3.8.3 DA-02 – Requesting Data within known XAIPs, which were formerly imported as BINs

Identifier	DA-02
Test Purpose	The test shall verify that it is possible to request certain pieces of data within a sequence of XAIPs identified by known AOIDs, which were formerly imported as BINs..
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that test cases SU-12 was successfully performed.

Step	**Test sequence**	**Expected Results**
1	ArchiveDataRequest (AOID_SU-12, DataLocation(DataLoc-P-01), DataLocation(DataLoc-Q-01))	ArchiveDataResponse (dss:Result(resultmajor#ok), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-P-01), Value(DataLoc-P-01)), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-Q-01), Value(DataLoc-Q-01)))
2	ArchiveDataRequest (AOID_SU-12, DataLocation(DataLoc-P-02), DataLocation(DataLoc-Q-02))	ArchiveDataResponse (dss:Result(resultmajor#ok), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-P-02), Value(DataLoc-P-02)), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-Q-02), Value(DataLoc-Q-02)))
Observations:		
Verdict:		

3.8.4 DA-03– Requesting Data within unknown XAIPs

Identifier	DA-03
Test Purpose	The test shall verify that there is an error, if a request contains an unknown XAIP.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that the test case SU-01 was successfully performed. Make sure, that the randomly generated AOID_RE-02 is within the possible range but has not been assigned yet.

Step	Test sequence	Expected Results
1	ArchiveDataRequest (AOID_SU-01, DataLocation(DataLoc-P-01), DataLocation(DataLoc-Q-01))	ArchiveDataResponse (dss:Result(resultmajor#ok), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-P-01), Value(DataLoc-P-01)), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-Q-01), Value(DataLoc-Q-01)))
2	ArchiveDataRequest (AOID_RE-03, DataLocation(DataLoc-P-01), DataLocation(DataLoc-Q-01))	ArchiveDataResponse (dss:Result(resultmajor#error resultminor/arl/unknownAOID))
Observations:		
Verdict:		

3.8.5 DA-04– Requesting Data within unknown XAIPs as formerly imported BINs

Identifier	DA-04
Test Purpose	The test shall verify that there is an error, if a request contains an unknown XAIP as formerly imported BINs.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that the test case SU-12 was successfully performed. Make sure, that the randomly generated AOID_RE-02 is within the possible range but has not been assigned yet.

Step	**Test sequence**	**Expected Results**
1	ArchiveDataRequest (AOID_SU-12, DataLocation(DataLoc-P-01), DataLocation(DataLoc-Q-01))	ArchiveDataResponse (dss:Result(resultmajor#ok), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-P-01), Value(DataLoc-P-01)), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-Q-01), Value(DataLoc-Q-01)))
2	ArchiveDataRequest (AOID_RE-03, DataLocation(DataLoc-P-01), DataLocation(DataLoc-Q-01))	ArchiveDataResponse (dss:Result(resultmajor#error resultminor/arl/unknownAOID))
Observations:		
Verdict:		

3.8.6 DA-05 – Requesting Data within known XAIPs and partly invalid DataLocations

Identifier	DA-05
Test Purpose	The test shall verify that there will be an error, if there is an ArchiveDataRequest with known AOIDs and invalid DataLocation-elements.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that test cases SU-01 was successfully performed.

Step	Test sequence	Expected Results
1	ArchiveDataRequest (AOID_SU-01, DataLocation(DataLoc-P-NOK), DataLocation(DataLoc-Q-NOK))	ArchiveDataResponse (dss:Result(resultmajor#error), XAIPData(dss:Result(resultmajor#error, resultminor/arl/notSupported,, DataLocation(DataLoc-P-NOK)), XAIPData(dss:Result(resultmajor#error, resultminor/arl/notSupported, DataLocation(DataLoc-Q-NOK)))
2	ArchiveDataRequest (AOID_SU-01, DataLocation(DataLoc-P-01), DataLocation(DataLoc-Q-01))	ArchiveDataResponse (dss:Result(resultmajor#ok), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-P-01), Value(DataLoc-P-01)), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-Q-01), Value(DataLoc-Q-01)))
3	ArchiveDataRequest (AOID_SU-01, DataLocation(DataLoc-P-01), DataLocation(DataLoc-Q-NOK))	ArchiveDataResponse (dss:Result(resultmajor#warning, resultminor/arl/requestOnlyPartlySucessfulWarning), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-P-01), Value(DataLoc-P-01)), XAIPData(dss:Result(resultmajor#error, resultminor/arl/notSupported, DataLocation(DataLoc-Q-NOK)))
Observations:		
Verdict:		

3.8.7 DA-06 – Requesting Data within known XAIPs as formerly imported BINs and partly invalid DataLocations

Identifier	DA-06
Test Purpose	The test shall verify that there will be an error, if there is an ArchiveDataRequest with known AOIDs and invalid DataLocation-elements concerning formerly imported BINs.
Configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exists. Make sure that test cases SU-12 was successfully performed.

Step	**Test sequence**	**Expected Results**
1	ArchiveDataRequest (AOID_SU-12, DataLocation(DataLoc-P-NOK), DataLocation(DataLoc-Q-NOK))	ArchiveDataResponse (dss:Result(resultmajor#error), XAIPData(dss:Result(resultmajor#error, resultminor/arl/notSupported, DataLocation(DataLoc-P-NOK)), XAIPData(dss:Result(resultmajor#error, resultminor/arl/notSupported,, DataLocation(DataLoc-Q-NOK)))
2	ArchiveDataRequest (AOID_SU-12, DataLocation(DataLoc-P-01), DataLocation(DataLoc-Q-01))	ArchiveDataResponse (dss:Result(resultmajor#ok), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-P-01), Value(DataLoc-P-01)), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-Q-01), Value(DataLoc-Q-01)))
3	ArchiveDataRequest (AOID_SU-12, DataLocation(DataLoc-P-01), DataLocation(DataLoc-Q-NOK))	ArchiveDataResponse (dss:Result(resultmajor#warning, resultminor/arl/requestOnlyPartlySucessfulWarning), XAIPData(dss:Result(resultmajor#ok), DataLocation(DataLoc-P-01), Value(DataLoc-P-01)), XAIPData(dss:Result(resultmajor#error, resultminor/arl/notSupported, DataLocation(DataLoc-Q-NOK)))
Observations:		
Verdict:		

3.8.8 DA-07 – Unknown control in OptionalInputs

Identifier	DA-05	
Test Purpose	The test shall verify that there will be an error, if the request contains unknown controls in the `OptionalInputs`-element.	
Configuration	CONFIG_Implicit	
Pre-test conditions	Authenticated connection to middleware exists. Make sure that the test case SU-01 was successfully performed.	
Step	**Test sequence**	**Expected Results**
1.	ArchiveDataRequest (dss:OptionalInputs(SomethingUnknown), AOID_SU-01, DataLocation(DataLoc-P-01), DataLocation(DataLoc-Q-01))	ArchiveDataResponse (dss:Result(resultmajor#error, resultminor/arl/notSupported), XAIPData(dss:Result(resultmajor#error, resultminor/arl/notSupported, DataLocation(DataLoc-P-01)), XAIPData(dss:Result(resultmajor#error, resultminor/arl/notSupported, DataLocation(DataLoc-Q-01)))
Observations:		
Verdict:		

3.9 VerifyRequest

All test cases defined in this section are derived from the general requirement E:A2.0-2 and E:A2.0-3, MD:A6.3-1, MD:A6.3-3, F:A3.6-1 and F:A6.2-1 together with the interface specification of the VerifyRequest function in section 3.2.2 of [eCard-2] and section 3.7. and 4.1.1 of [TR-ESOR-E].

3.9.1 VE-01-*x* – Verify XAIP$_{xyz}$ with verify-xaip

Identifier	VE-01-*x*	
Test Purpose	The test shall verify that it is possible to verify different XAIP$_{xyz}$ with the TR-ESOR-specific signature policy http://www.bsi.bund.de/tr-esor/sigpolicy/verify-xaip. The test is performed for each XAIP$_{xyz}$, where *x* is specified in section 3.2 and listed below and *y* is the number of the hash algorithm, which has to be supported according to [TR-ESOR-ERS], chapter 5 and *z*=1 means CMS-based signatures and *z*=2 means XML-based signatures.	
	x	XAIP$_{xy}$
	1	XAIP_OK$_y$
	2	XAIP_NOK$_y$
	3	XAIP_NOK_EXPIRED$_y$
	4	XAIP_NOK_SUBMTIME$_y$
	5	XAIP_NOK_SIG$_y$
	6	XAIP_NOK_ER$_y$
	7	XAIP_NOK_SIG_OK_ER$_y$
	8	XAIP_OK_SIG_OK_ER$_y$
	9	DXAIP_OK$_y$
	10	DXAIP_NOK$_y$
	11	DXAIP_NOK_AOID$_y$
	12	DXAIP_NOK_EXPIRED$_y$
	13	DXAIP_NOK_SUBMTIME$_y$
	14	DXAIP_NOK_SIG$_y$
	15	DXAIP_NOK_ER$_y$
	16	DXAIP_NOK_VERSION$_y$
	17	DXAIP_NOK_ID$_y$
Configuration	CONFIG_Implicit	
Pre-test conditions	Authenticated connection to middleware exists. Requirements (A3.7.1-1); The usage of the dss:OptionalInputs “VerifyUnderSignaturePolicy” and “ReturnVerificationReport” is optional.	

Step	**Test sequence**	**Expected Results**
1.	VerifyRequest (dss:OptionalInputs(VerifyUnderSignaturePolicy(DefaultPolicy(SignaturePolicyIdentifier(http://www.bsi.bund.de/tr-esor/sigpolicy/verify-xaip	For x=1 or x=8:VerifyResponse (dss:Result(resultmajor#ok), dss:OptionalOutputs(XAIP(XAIP$_x$ + VALINFO$_x$))) For x=9:

	)))), dss:InputDocuments(dss:Document(Base64XML(XAIP$_x$))))	VerifyResponse (dss:Result(resultmajor#ok), dss:OptionalOutputs(D XAIP(D XAIP$_x$ + VALINFO$_x$))) For 9 < x < 18 : VerifyResponse (dss:Result(resultmajor#error, resultminor/arl/DXAIP$_x$), dss:OptionalOutputs(DXAIP(DXAIP$_x$ + VALINFO$_x$))) For x =7: VerifyResponse (dss:Result(resultmajor#error, resultminor/arl/XAIP$_5$), dss:OptionalOutputs(XAIP(XAIP$_x$ + VALINFO$_x$))) For 2 < x < 7 : VerifyResponse (dss:Result(resultmajor#error, resultminor/arl/XAIP$_x$), dss:OptionalOutputs(XAIP(XAIP$_x$ + VALINFO$_x$)))
2.	VerifyRequest (dss:OptionalInputs(VerifyUnderSignaturePolicy(DefaultPolicy(SignaturePolicyIdentifier(http://www.bsi.bund.de/tr-esor/sigpolicy/verify-xaip))), vr:ReturnVerificationReport(vr:IncludeVerifier(true), vr:IncludeCertificateValues(true), vr:IncludeRevocationValues(true), vr:ReportDetailLevel(urn:oasis:names:tc:dss:1.0:profiles:153verificationreport:reportdetail:allDetails))), dss:InputDocuments(dss:Document(Base64XML(XAIP$_x$))))	For x=1 or x=8:VerifyResponse (dss:Result(resultmajor#ok), dss:OptionalOutputs(XAIP(XAIP$_x$ + VALINFO$_x$), vr:VerificationReport(vr:IndividualReport(tr:Details(XAIPReport))))) For x =7: VerifyResponse (dss:Result(resultmajor#error, resultminor/arl/XAIP$_5$), dss:OptionalOutputs(XAIP(XAIP$_x$ + VALINFO$_x$), vr:VerificationReport(vr:IndividualReport(tr:Details(XAIPReport))))) For 2 < x < 7: VerifyResponse (dss:Result(resultmajor#error,

		resultminor/arl/XAIP$_x$), dss:OptionalOutputs(XAIP(XAIP$_x$ + VALINFO$_x$), vr:VerificationReport(vr:IndividualReport(tr:Details(XAIPReport)))))
Observations:		
Verdict:		

3.9.2 VE-02-*y* – Verify TST_{yz} with verify-timestamp

Identifier	VE-02-*yz*	
Test Purpose	The test shall verify that it is possible to verify different TST_y with the TR-ESOR-specific signature policy http://www.bsi.bund.de/tr-esor/sigpolicy/verify-timestamp. The test is performed for each TST_{yz} specified in section 3.2 and listed below and z is the sequence number of the hash algorithm, which has to be supported according to [TR-ESOR-ERS], chapter 5.	
	y	TST_{yz}
	1	TST_OK_z
	2	$TST\_OK\_VALINFO_z$
	3	TST_NOK_z
	4	$TST\_NOK\_SIG_z$[56]
	5	$TST\_NOK\_VALINFO_z$
	7	$TST\_BASIS\_ERS\_OK_z$[57]
	8	$TST\_BASIS\_ERS\_OK\_VALINFO_z$
	9	$TST\_BASIS\_ERS\_NOK_z$
	10	$TST\_BASIS\_ERS\_NOK\_SIG_z$[58]
	11	$TST\_BASIS\_ERS\_NOK\_VALINFO_z$
Configuration	CONFIG_Implicit	
Pre-test conditions	Authenticated connection to middleware exists. The usage of the verificationReport-element is optional. The TSTs are based on RFC3161.	

Step	**Test sequence**	**Expected Results**
1.	VerifyRequest (dss:OptionalInputs(VerifyUnderSignaturePolicy(DefaultPolicy(SignaturePolicyIdentifier(http://www.bsi.bund.de/tr-esor/sigpolicy/verify-timestamp)))), dss:SignatureObject(dss:TimeStamp(RFC3161TimeStampToken(TST_y))))	For y=1 or y=2 or y=7 or y=8: VerifyResponse (dss:Result(resultmajor#ok), dss:OptionalOutputs(UpdatedSignature(dss:SignatureObject(dss:TimeStamp(RFC3161TimeStampToken(TST_y + $VALINFO_y$)))))) For y=3 or y=4 or y=5 or y=9 or y=10 or y=11: VerifyResponse (dss:Result(resultmajor#error, resultminor/arl/TST_y), dss:OptionalOutputs(UpdatedSignature(dss:SignatureObject(

[56] If the signature validation data are not complete, the result dss:Result(resultmajor#error, resultminor/arl/TST_NOK_SIG is received.

[57] According to the BASIS_ERS_Profile as defined in [TR-ESOR-ERS]

[58] If the signature validation data are not complete, the result dss:Result(resultmajor#error, resultminor/arl/TST_NOK_SIG ist received.

		dss:TimeStamp(RFC3161TimeStampToken(TST_y + $VALINFO_y$))))))
2.	VerifyRequest (dss:OptionalInputs(VerifyUnderSignaturePolicy(DefaultPolicy(SignaturePolicyIdentifier(http://www.bsi.bund.de/tr-esor/sigpolicy/verify-timestamp))) vr:ReturnVerificationReport(vr:IncludeVerifier(true), vr:IncludeCertificateValues(true), vr:IncludeRevocationValues(true), vr:ReportDetailLevel(urn:oasis:names:tc:dss:1.0:profiles:153verificationreport:reportdetail:allDetails))), dss:SignatureObject(dss:TimeStamp(RFC3161TimeStampToken(TST_y))))	For y=1 or y=2: VerifyResponse (dss:Result(resultmajor#ok), dss:OptionalOutputs(UpdatedSignature(dss:SignatureObject(dss:TimeStamp(RFC3161TimeStampToken(TST_y + $VALINFO_y$)))))) For y=3 or y=4 or y=5 or y=9 or y=10 or y=11:: VerifyResponse (dss:Result(resultmajor#error, resultminor/arl/TST_y), dss:OptionalOutputs(UpdatedSignature(dss:SignatureObject(dss:TimeStamp(RFC3161TimeStampToken(TST_y + $VALINFO_y$)))), vr:VerificationReport(vr:IndividualReport(vr:Details(vr:IndividualTimeStampReport)))))
Observations:		
Verdict:		

3.9.3 VE-03 – Verify with unknown signature policy

Identifier	VE-03	
Test Purpose	The test shall verify that there is an appropriate error if the VerifyRequest is called with an unknown signature policy identifier.	
Configuration	CONFIG_Implicit	
Pre-test conditions	Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1.	VerifyRequest (dss:OptionalInputs(VerifyUnderSignaturePolicy(DefaultPolicy(SignaturePolicyIdentifier(http://something.unknown.org)))), dss:InputDocuments(dss:Document(Base64XML(XAIP_OK))))	VerifyResponse (dss:Result(resultmajor#error, resultminor/signature/SignaturePolicyNotSupported[59]))
Observations:		
Verdict:		

[59] **NOTE:** The error code ...SignaturePolicyNotSupported will be included in the next version of the specification.

3.9.4 VE-04 – Unknown control in `OptionalInputs`

Identifier	VE-04	
Test Purpose	The test shall verify that there will be a warning, if the request contains unknown controls in the `OptionalInputs`-element.	
Configuration	CONFIG_Implicit	
Pre-test conditions	Authenticated connection to middleware exists.	
Step	**Test sequence**	**Expected Results**
1.	VerifyRequest (dss:OptionalInputs(SomethingUnknown), dss:SignatureObject(dss:TimeStamp(RFC3161TimeStampToken(TST_OK)))))	VerifyResponse (dss:Result(resultmajor#error, resultminor/arl/notSupported))
Observations:		
Verdict:		

3.10 VE-05 – Verification of versioned XAIP-Versions

Identifier	VE-05
Test Purpose	The test shall show that a single version of an XAIP container is protected by the Evidence Record which is relevant for this version. ***Note**, that after performing the test cases SU-01 and UP-01 there are two versions of the archive data object stored in the long term storage: the first version v1 (see test case SU-01) and a second version (see test case UP-01).*
configuration	CONFIG_Implicit
Pre-test conditions	Authenticated connection to middleware exits. Make sure, that the test case SU-01[60] was successfully performed. Make sure, that the test case UP-01[61] was successfully performed.

Step	Test sequence	Expected Results
1.	ArchiveRetrievalRequest(AOID(AOID_SU-01), VersionID(v1))	ArchiveRetrievalResponse(dss:Result(resultmajor#OK), XAIP_OK(VID_SU-01)[62])
2.	ArchiveEvidenceRequest(AOID(AOID_SU_01), VersionID(v1))	ArchiveEvidenceResponse(dss:Result(resultmajor#ok), xaipERS(AOID_SU-01,VID_SU-01, ASN.1-ERS(XAIP_OK)[63])))
3.	VerifyRequest (dss:OptionalInputs (VerifyUnderSignaturePolicy (DefaultPolicy (SignaturePolicyIdentifier (http://www.bsi.bund.de/ tr-esor/sigpolicy/verify-xaip)))), dss:InputDocuments (dss:Document (Base64XML(XAIP_OK+ASN1-ERS(XAIP_OK)))))	VerifyResponse(dss:Result(resultmajor#ok,))
4.	ArchiveEvidenceRequest(AOID(AOID_SU-01)[64])	ArchiveEvidenceResponse(dss:Result(resultmajor#ok), xaipERS(AOID_SU-01,VID_UP-01, ASN.1-ERS(XAIP_OK+DXAIP_OK)[65]

[60] The test case is defined in Section 3.3.1.
[61] The test case is defined in Section 3.4.1.
[62] The original first version (v1) is returned → XAIP_OK.
[63] The Evidence Record belonging to version v1 is returned.
[64] No input of a version number → the last version will be returned.
[65] The Evidence Record belonging to version v2=VID_UP_01, created on XAIP_OK and DXAIP_OK, is returned.

		)))
5.	VerifyRequest (dss:OptionalInputs (VerifyUnderSignaturePolicy (DefaultPolicy (SignaturePolicyIdentifier (http://www.bsi.bund.de/ tr-esor/sigpolicy/verify-xaip)))), dss:InputDocuments (dss:Document (Base64XML((XAIP_OK) +ASN1-ERS(XAIP_OK+DXAIP_OK)))))	VerifyResponse(dss:Result(resultmajor#error, resultminor/sal#invalidSignature))
6.	ArchiveRetrievalRequest(AOID(AOID_SU-01))	ArchiveRetrievalResponse (dss:Result(resultmajor#OK), xaip:XAIP(XAIP_OK+DXAIP)[66])
7.	VerifyRequest (dss:OptionalInputs (VerifyUnderSignaturePolicy (DefaultPolicy (SignaturePolicyIdentifier (http://www.bsi.bund.de/ tr-esor/sigpolicy/verify-xaip)))), dss:InputDocuments (dss:Document (Base64XML((XAIP_OK+DXAIP_OK) +ASN1-ERS(XAIP_OK+DXAIP_OK)))))	VerifyResponse(dss:Result(resultmajor#ok,))
Observations:		
Verdicts:		

[66] The last version v2, created on base of XAIP_OK and DXAIP_OK, is returned.