Technical Guideline TR-03130
eID-Server

Part 1: Functional Specification

Version 2.2.0
16.12.2019
## Document history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0 Draft</td>
<td>July 20.07.2012</td>
<td>English translation, complete restructuring and deletion of redundant parts, eID-Activation as specified in <a href="#">TR-03112</a> starting from Version 1.1.2</td>
</tr>
<tr>
<td>2.0 RC1</td>
<td>September 21.09.2012</td>
<td>Correction of signature in SAML context, Introduction of NotOnChip flag for attributes, Correction and standardization of various examples and lists, Correction of several references, Optimization of message flow description</td>
</tr>
<tr>
<td>2.0</td>
<td>October 24.10.2012</td>
<td>Minor corrections, Code examples labeled and listed</td>
</tr>
<tr>
<td>2.0.1 RC</td>
<td>December 2.12.2013</td>
<td>Conversion to new CD, No more Unions in XML-Structure, Default-Values for Attributes in Operations...Types, Minor corrections, Updated References</td>
</tr>
<tr>
<td>2.0.1</td>
<td>January 15.01.2014</td>
<td>Boolean values corrected to lower-case, Minor corrections</td>
</tr>
<tr>
<td>2.0.2</td>
<td>November 16.11.2016</td>
<td>Improved Description for Error Handling, Incorporation of feedback from specification of conformity tests (cf. Part 4 of this Technical Guideline)</td>
</tr>
<tr>
<td>2.1.1 Draft</td>
<td>26.07.2017</td>
<td>Added changes for use in the context of eIDAS, Added new types of service providers, General corrections and revisions</td>
</tr>
<tr>
<td>2.2.0</td>
<td>16.12.2019</td>
<td>Added changes regarding the eID-Card for Union Citizens. Updated and improved section about infrastructure.</td>
</tr>
</tbody>
</table>
Content

Document history...................................................................................................................................................... 2
1 Introduction.................................................................................................................................................................. 9
  1.1 Out of Scope.................................................................................................................................................... 9
  1.2 Key Words.................................................................................................................................................... 10
2 Infrastructure and general regulations.................................................................................................................. 11
  2.1 Description.................................................................................................................................................... 11
     2.1.1 eService ................................................................................................................................................ 11
     2.1.2 eID-Server ............................................................................................................................................ 11
  2.2 Communication Model.................................................................................................................................... 12
  2.3 Interfaces....................................................................................................................................................... 14
     2.3.1 eService Communication .................................................................................................................... 14
     2.3.2 Communication to eID-Client and eID-Document ............................................................................. 15
     2.3.3 Public Key Infrastructure ................................................................................................................... 15
     2.3.4 eIDAS Connector .................................................................................................................................. 16
  2.4 Document Validity Verification....................................................................................................................... 16
  2.4.1 CSCA Certificate Trust Store .................................................................................................................. 16
  2.4.2 Certificate Revocation Lists .................................................................................................................. 17
  2.4.3 Validation of Signed Objects ................................................................................................................. 17
  2.4.4 Black Lists .............................................................................................................................................. 17
  2.4.5 Defect Lists ............................................................................................................................................. 18
  2.4.6 Document Validation Procedure .......................................................................................................... 18
3 eID-Interface (SOAP) ........................................................................................................................................... 19
  3.1 General Message Flow .................................................................................................................................... 19
     3.1.1 Initiation ............................................................................................................................................... 19
     3.1.2 Interaction .......................................................................................................................................... 20
     3.1.3 Completion ......................................................................................................................................... 21
  3.2 Functions....................................................................................................................................................... 21
     3.2.1 useID.................................................................................................................................................. 22
     3.2.2 getResult......................................................................................................................................... 24
     3.2.3 getServerInfo................................................................................................................................. 26
  3.3 Data types.................................................................................................................................................... 27
     3.3.1 SessionType....................................................................................................................................... 27
     3.3.2 RestrictedIDType............................................................................................................................... 28
     3.3.3 PersonalDataType............................................................................................................................... 29
     3.3.4 GeneralPlaceType............................................................................................................................. 30
     3.3.5 PlaceType.......................................................................................................................................... 30
     3.3.6 OperationsSelectorType.................................................................................................................... 31
     3.3.7 OperationsRequestorType .................................................................................................................. 32
     3.3.8 OperationsResponderType .................................................................................................................. 33
     3.3.9 AgeVerificationRequestType ............................................................................................................ 34
     3.3.10 PlaceVerificationRequestType ........................................................................................................ 34
     3.3.11 VersionType..................................................................................................................................... 35
     3.3.12 VerificationReturnType...................................................................................................................... 35
     3.3.13 PreSharedKeyType............................................................................................................................ 36
     3.3.14 GeneralDateTime.............................................................................................................................. 36
     3.3.15 AttributeSelectionType..................................................................................................................... 36
     3.3.16 AttributeRequestType....................................................................................................................... 37
5  eIDAS-Extension and eSAML Profile........................................................................................................... 73
  5.1 Infrastructure Adaptation to eIDAS context.............................................................................................. 73
  5.2 eIDAS Message Flow..................................................................................................................................... 74
        5.2.1 Initiation.............................................................................................................................................. 74
        5.2.2 eIDAS authentication process........................................................................................................... 75
        5.2.3 Completion......................................................................................................................................... 76
  5.3 Additional Attributes in eSAML .................................................................................................................. 77
        5.3.1 Request Attributes............................................................................................................................. 78
        5.3.2 Response Attributes........................................................................................................................... 78
  5.4 Additional Data Types in eSAML.................................................................................................................. 78
        5.4.1 EidasExtensionType........................................................................................................................... 79
        5.4.2 LevelOfAssuranceType...................................................................................................................... 79
        5.4.3 SendingMemberStateType................................................................................................................ 80
        5.4.4 ICAOSex............................................................................................................................................... 80
  5.5 Attribute Mapping......................................................................................................................................... 80
        5.5.1 eIDAS Minimum Data Set................................................................................................................ 80
        5.5.2 Further Attributes............................................................................................................................ 81
        5.5.3 Non-Latin Characters........................................................................................................................ 81
        5.5.4 RestrictedID........................................................................................................................................ 81
        5.5.5 BirthName.......................................................................................................................................... 81
        5.5.6 PlaceOfResidence.............................................................................................................................. 82
  5.6 Error Handling.............................................................................................................................................. 83

References.............................................................................................................................................................. 84

Figures
Figure 1: Interfaces and surrounding Components.......................................................................................... 11
Figure 2: Generic Communication Model (eID-Interface)................................................................................... 12
Figure 3: Communication Model for SAML based Authentication................................................................. 13
Figure 4: Communication with Attached eID-Server ....................................................................................... 13
Figure 5: Integrated eID-Client in the Generic Communication Model and with Attached eID-Server............ 13
Figure 6: Overview of the relevant interfaces................................................................................................. 14
Figure 7: General message flow during initiation (SOAP).............................................................................. 19
Figure 8: General message flow during interaction (SOAP).......................................................................... 20
Figure 9: General message flow during completion (SOAP)............................................................................ 21
Figure 10: Webservice overview.................................................................................................................... 22
Figure 11: Function useID Request.................................................................................................................. 22
Figure 12: Function useID Response............................................................................................................... 23
Figure 13: Function getResult Request.......................................................................................................... 24
Figure 14: Function getResult Response........................................................................................................ 25
Figure 15: Function getServerInfo Response................................................................................................. 27
Figure 16: Data type SessionType.................................................................................................................. 27
Figure 17: Data type RestrictedIDType............................................................................................................ 28
Figure 18: Data type PersonalDataType.......................................................................................................... 29
Figure 19: Data type GeneralPlaceType.......................................................................................................... 30
Figure 20: Data type PlaceType....................................................................................................................... 30
Figure 21: Data type OperationsSelectorType................................................................................................. 31
Figure 22: Data type OperationsRequestorType.............................................................................................. 32
Figure 23: Data type OperationsResponderType.............................................................................................. 33
Figure 24: Data type AgeVerificationRequestType.......................................................................................... 34
Figure 25: Data type PlaceVerificationRequestType........................................................................................ 34
Tables

Table 1: Function useID Parameters.......................................................................................... 23
Table 2: Function useID Return Values.................................................................................... 24
Table 3: Function getResult Parameters.................................................................................. 25
Table 4: Function getResult Return Values.............................................................................. 26
Table 5: Function getServerInfo Return Values..................................................................... 27
Table 6: List of Error Codes...................................................................................................... 39
Table 7: Mapping of eID-Infrastructure and SAML Specification roles.................................... 46
Table 8: SAML Request Attributes.......................................................................................... 50
Table 9: SAML Response Attributes........................................................................................ 51
Table 10: Attributes and Elements of the Data type DocumentValidityResultType.................. 53
Table 11: Attribute and Elements of the Data type AuthnRequestExtensionType...................... 55
Table 12: Elements and Attributes of the AuthnRequest............................................................. 57
Table 13: Elements and Attributes of the AuthnRequestExtension........................................... 58
Table 14: Elements and Attributes of the Response................................................................. 59
Table 15: Elements and Attributes of the Assertion................................................................. 62
Table 16: Supplementing SAML Request attributes for eIDAS usage.................................... 78
Table 17: Supplementing SAML response attributes for eIDAS usage................................... 78
Table 18: Elements of the Data Type EidasExtensionType and their usage in Request and Response 79
Table 19: Attribute Mapping of the Minimum Data Set............................................................. 80
Table 20: Handling of Attributes not in the eIDAS Minimum Data Set.................................... 81
Table 21: Examples for handling of the name mapping............................................................. 82
Table 22: Mapping of address elements.................................................................................... 82

Examples

Example 1: InitiatorToken........................................................................................................ 40
1 Introduction

In the course of the digitization of business and governmental processes, secure electronic identification is of crucial importance in order to enable trust in electronic services.

This Technical Guideline specifies the eID-Server for Online-Authentication based on Extended Access Control Version 2 (EAC2) between an eService and an eIDAS token, e.g. the German National Identity Card, the German electronic Residence Permit or the eID-Card for Union Citizens (in the following subsumed under the term “eID document”). The eID-Server implements the server side of this authentication. The client side is specified by [TR-03124].

The eID-Server serves to encapsulate the complexity of Online-Authentication into a dedicated component and provides interfaces for eServices and eID-Clients and uses the interface to the eID Public Key Infrastructure (PKI). If the eID-Server should be able to provide cross-border authentication in the context of the eIDAS regulation, the eID-Server will communicate to an eIDAS Service over an eIDAS Connector. The eID-Client and eID-Server are based on the eCard-API-Framework and support a subset of functions specified by this framework.

The eID-Server is operated by the Service Provider itself or by a dedicated eID-Service. The term Service Provider in the context of this Technical Guideline includes also the Identification Service Provider (compare [TR-03128]) unless further mentioned. The eID-Server may also be used by a “Service Provider for On-Site Reading among Attendees” as described in [TR-03128].

This Technical Guideline specifies the external interfaces the eID-Server may provide to the Service Provider, i.e. eID-Interface (SOAP) and the SAML-Profile. When used in the eIDAS context an extended SAML-Profile (eSAML) is provided to the Service Provider and the eID-Server communicates with the eIDAS Connector.

While Part 1 of this guideline contains the Functional Specification, Part 2 contains a Security Framework for the secure operation of eID-Servers. Part 2 is especially intended to assist the Service Provider in creating a security concept based on IT-Grundschutz\(^1\). Part 3 describes the eID-Server based eIDAS-Middleware for providing authentication of German eID tokens in the context of the eIDAS regulation. Part 4 specifies conformance tests according to the requirements of Part 1 and Part 3 of the Technical Guideline.

1.1 Out of Scope

This technical guideline describes no organizational process flows in the eID-Server’s operation and no requirements for general administration. Change management of the eID-Document is not considered in this guideline either. This means there is only read access to the eID-Function’s data groups and functions. Requirements for integrating the eID-Server into an identity management are only described in the context of SAML.

Furthermore some components of the eID-Server are described in separate guidelines. This concerns especially:

- The eID-Server’s communication with the eID-Client MUST be implemented according to the eCard-API-Framework which is specified in [TR-03112].
- The eID-Client's functionalities are described in [TR-03124].
- The communication of the eIDAS Connector with eIDAS Services MUST be implemented according to [eIDAS-Interop].

\(^1\) https://www.bsi.bund.de/ContentBSI/EN/Topics/ITGrundschutz/itgrundschutz.html
• While the technical communication with the components of the Public Key Infrastructure (PKI) is specified in [TR-03129] the general organization of the PKI used for the eID-Function and the requirements for every participant of the PKI are specified in [CP-eID].

1.2 Key Words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. The key words "CONDITIONAL" and "IF" are to be interpreted as follows:

CONDITIONAL/IF: The usage of an item is dependent on the usage of other items. It is therefore further qualified under which conditions the item is REQUIRED or RECOMMENDED.

In some cases these key words may occur without directly referring to the eID-Server in this document and are then written in lower case only.
2 Infrastructure and general regulations

In this section the services offered by the eID-Server and the requirements placed on its operational environment are described. In addition this section describes all further components relevant in the eID-Server's context and the interfaces they share.

2.1 Description

![Figure 1: Interfaces and surrounding Components](image)

2.1.1 eService

The eService implements the application logic of the web application presented in the user's Browser. The application itself offered by the eService is out of scope of this guideline. In this context, it is only important that the eService uses the eID-Server to offer mutual Online-Authentication for its users. This especially means that the eService must implement the protocols specified in [TR-03124] Part 1, Section 2: Online-Authentication based on EAC2.

2.1.2 eID-Server

As a hard- and software component, the eID-Server implements the server-side eCard-API-Framework as specified in [TR-03112] and establishes communication to the client-side eCard-API-Framework implementation (eID-Client). See Section 2.3.2 for the profile of the eCard-API-Framework to be implemented. It also takes over communication for calling terminal authorization certificates, revocation lists and CSCA certificates (see Figure 1: Interfaces and surrounding Components) from the Public Key Infrastructure (PKI).
For these purposes the application logic controlling the process flow and all relevant security algorithms and features SHALL be implemented according to this specification. The eID-Server must have access to all counterparts of the interfaces listed in Section 2.3: Interfaces.

The eID-Server saves and manages authentication certificates on the Service Provider's behalf and MUST protect the Service Provider's terminal authorization certificates from misuse.

The eID-Server MAY be implemented as a logically autonomous server so that several eServices (clients) are able to use it and it MAY also be operated by a third party. The data exchanged between eService and eID-Server is always signed and SHALL be encrypted if it is sent through an open network to secure the processed data's confidentiality, authenticity and integrity.

This technical guideline specifies the eID-Server's external communication with the eService. The eID-Interface's main objective is mutual authentication and preparation of data from the user's eID-Document for the eService.

### 2.2 Communication Model

The Online-Authentication starts from an existing TLS channel named “TLS-1” between the browser of the user and the website of the eService. The Online-Authentication is performed between the eID-Client and the eID-Server of the eService using a second TLS channel “TLS-2”.

In general, TLS-1 and TLS-2 terminate at different domains. An intermediary TLS channel “TLS-1-2” between eID-Client and eService is necessary to allow the binding of TLS-1 and TLS-2. The eService and the eID-Server MUST communicate using an encrypted and integrity-protected mutually authenticated connection, which is described in this guideline (see Section 2.3.1).

The communication can use a direct connection between eService and eID-Server e.g. over a VPN as shown in Figure 2: Generic Communication Model (eID-Interface). This interface is described in Section 3 eID-Interface (SOAP).

Alternatively or additionally a SAML communication relayed by the eID-Client can be used, as shown in Figure 3: Communication Model for SAML based Authentication and described in Section 4 SAML-Profile.
The eService and eID-Server MAY be operated as a single component ("Attached eID-Server") as shown in Figure 4: Communication with Attached eID-Server. In this case the intermediary channel TLS-1-2 is not necessary and the requirements according to [TR-03124] Part 1, Section 2.1 Communication Model and 2.7 Session Binding MUST be fulfilled.

For Integrated eID-Clients TLS-1 and TLS-1-2 MAY be the same channel, or a single TLS channel can be used in the case of an Integrated eID-Client communicating with an Attached eID-Server as shown in Figure 5: Integrated eID-Client in the Generic Communication Model and with Attached eID-Server.
2.3 Interfaces

The eID-Server communicates with other entities via several interfaces:

- to the eService, see section 2.3.1;
- to the eID-Client and eID-Document, see section 2.3.2;
- to the Public Key Infrastructure (PKI), see section 2.3.3; and
- OPTIONALLY to the eIDAS Network, see section 2.3.4.

The eID-Server’s interfaces are based on web services and can thus generally be addressed through open networks. Additional services of the eID-Server MAY also be provided through additional interfaces. One example is an Administration Interface for the Service Provider which the eID-Server MAY offer.

2.3.1 eService Communication

The eService invokes the eID-Server, and the eID-Server in turn returns the read personal data via one of the following interfaces:

- a SOAP based eID-Interface as specified in Section 3; or
- a SAML based interface offered by a SAML processor communicating to or being directly attached to the eID-Server as specified in Section 4. This interface supports identification via other European eID schemes utilizing the eIDAS network in addition to the eID function of the German eID system, see Section 5.
- an internal interface, if an eID-Server is directly attached to an eService, i.e. the communication between eService and eID-Server is purely internal communication. This interface is out of scope of this specification.

The eService receives or generates a random Pre-Shared Key (PSK) and a unique identifier for the PSK for each Online-Authentication to bind the eCard-API-Framework and eID-Interface communication to the web-session. The PSK must be randomly generated exclusively for each Online-Authentication that the eID-Server performs for the eService. The eService and eID-Server MUST ensure the PSK’s confidentiality while transferring it to the eID-Client with the help of the TC Token.
2.3.2 Communication to eID-Client and eID-Document

The eID-Server communicates to the eID-Client (see [TR-03124]) via the eCard API as specified in [TR-03112] in order to read the data from the eID Document, that is presented by the user. This communication comprises SOAP/PAOS based command exchange encapsulated in TLS.

The eID-Client responds to the requests it receives via the user's browser and subsequently connects itself to the eID-Server. To do this, the client establishes a connection to the eID-Server's eCard-API-Framework interface which is then used to read out data and perform operations.

Communication is initiated by the eID-Client via negotiating a TLS channel to the eID-Server. For this channel, the eID-Server SHALL support the cipher suite TLS_RSA_PSK_WITH_AES_256_CBC_SHA (compare section 4.5 of [TR-03124]). Additional TLS_RSA_PSK_* cipher suites according to [TR-03116], Part 4, MAY be supported. Other cipher suites MUST NOT be supported. The eID-Server SHALL support session resumption.

The eID-Server SHALL implement at least the following commands of the eCard-API [TR-03112] to support Online-Authentication (compare section 3.5 of [TR-03124]):

- As callee:
  - StartPAOS ([TR-03112], Part 7) (REQUIRED)
- As caller:
  - DIDAuthenticate ([TR-03112], Part 4) with support for AuthenticationProtocolData of type EAC1InputType,EAC2InputType and EACAdditionalInputType ([TR-03112], Part 7) (REQUIRED)
  - Transmit ([TR-03112], Part 6) (REQUIRED)

2.3.3 Public Key Infrastructure

The German eID system utilizes two public key infrastructures (PKIs), one to ensure the authenticity of eID Documents (and the data contained therein), the other to control access to eID Documents.

For document authenticity, the German eID system uses the ICAO PKI for eMRTDs as specified in [ICAO 9303], Part 12. The root of this PKI (Country Signing Certificate Authority – CSCA) is operated by the BSI.

For access control, the German eID system includes a PKI (Country Verifying Certificate Authority – CVCA) whose root is also operated by the BSI. Certification of eService terminal authorization is handled via a subordinate CA (Berechtigungs-CA or BerCA). The eID-Server retrieves the necessary terminal authorization certificates on behalf of the eService from this subordinate CA and protects the private keys of these certificates according to [CP-eID].

To simplify communication, the BerCA offers all necessary PKI related communication via a common web service based interface specified in [TR-03129].

The eID-Server SHALL implement the following web service methods from Part 1 of [TR-03129], including the corresponding call backs, where applicable:

- RequestCertificate
2.3.4 eIDAS Connector

In order to accept identification via other European eID schemes utilizing the eIDAS network as a relying party, the eID-Server MAY implement an eIDAS Connector as described in the eIDAS Interoperability Framework [eIDAS-Interop]. Over the eIDAS Connector the user’s authentication session invoked by the eService via the Extended SAML Profile as described in Section 5 is then forwarded towards the eIDAS Service.

The eIDAS Connector interface SHALL implement SAML communication according to [eIDAS-SAML]. Trust and responsibility within the eIDAS network is ensured by the exchange of signed SAML metadata between eIDAS Connectors and eIDAS Services. The metadata for each Connector is signed with a Metadata-Signer certificate (see [CP-eID]), which is obtained from the BerCA. The signed metadata MUST be made publicly available under a HTTPS URL.

2.4 Document Validity Verification

An eID-Document is valid if it is checked to be genuine (a real document), authentic and not expired/not revoked. The eID-Server MUST check the validity of the eID-Document by

- performing Chip Authentication,
- performing Passive Authentication,
- checking Black List(s),
- checking validity date,

and additionally

- the eID-Server MAY use defect-lists provided.

See also Section 4.7 of [TR-03127].

2.4.1 CSCA Certificate Trust Store

In order to perform validity verification of cryptographically secured objects, the eID-Server SHALL maintain a Trust Store of trusted CSCA Root certificates.

Upon deployment of the eID-Server software, the Trust Store contains the trusted CSCA Root certificates available at that time. To update the Trust Store after deployment, several mechanisms are available:

- Update via CSCA Link certificates. The CSCA issues Link certificates linking a new CSCA Root certificate to the previous one, i.e. if the previous CSCA Root certificate is trusted, the eID-Server can establish trust in the new CSCA Root certificate via validating the CSCA Link certificate. Retrieving CSCA Link certificates is out of scope of this specification.
- Update via Master Lists. The CSCA provides (via the BerCA) a Master List (see [TR-03129]) containing all valid CSCA Root certificates and CSCA Link certificates issued by that CSCA. The eID-Server...
downloads the Master List on a regular basis, validates the Master List according to section 2.4.3 and imports any CSCA Root certificates not yet known into the Trust Store.

- Manual configuration. The Trust Store can also be updated via careful manual configuration by the operator.

Compliant eID-Server SHALL support Trust Store update via Master Lists and MAY support Trust Store update via CSCA Link certificates and manual configuration.

The eID-Server MAY remove expired CSCA certificates from the Trust Store.

2.4.2 Certificate Revocation Lists

To check the revocation state of CSCA certificates and Signer certificates, the eID-Server SHALL use the Certificate Revocation List provided by the CSCA and check the revocation state in compliance with section 6.2 of [ICAO 9303], Part 12.

It is not necessary to retrieve the CRL during each certificate validation. The CRL MAY be cached by the eID-Server for a defined time, e.g. 24h.

Note: Revoked CSCA certificates are not automatically removed from the Master List, i.e. explicit revocation check of CSCA certificates via the Certificate Revocation List is necessary.

2.4.3 Validation of Signed Objects

Document Security Objects, Master Lists and Defect Lists share a common structure, i.e. they are all signed CMS containers. Therefore, the validation procedure for these Signed Objects is the same, as detailed below. To verify the validity of a Signed Object, the following steps MUST be performed:

1. The eID-Server SHALL extract the Signer Certificate (i.e. Document Signer certificate, Master List Signer certificate, or Defect List signer certificate) from the CMS object.
2. The eID-Server SHALL check if the CSCA certificate used to sign the Signer certificate is present in the Trust Store. If not, the validation procedure fails.
3. The eID-Server SHALL perform a path validation of the certificate path CSCA Root certificate → Signer certificate according to Section 6.2 of [ICAO 9303], Part 12. A mapping to the path validation procedure of the Web PKI according to Section 6.1 of [RFC 5280] is given in Appendix D of [ICAO 9303], Part 12.

As part of the path validation the following checks are performed:
1. check if the CSCA certificate is valid, i.e. not expired and not revoked;
2. verify the signature of the Signer certificate;
3. check if the Signer certificate is valid, i.e. not expired and not revoked;

If any of these checks fails, the path validation procedure fails.

4. The eID-Server SHALL verify the signature of the Signed Object using the public key from the validated Signer certificate.

2.4.4 Black Lists

The revocation state of eID-Documents is communicated via Black Lists according to Appendix B of [TR-03129]. The eID-Server SHALL retrieve a fresh Black List on a regular basis.
2.4.5 Defect Lists

The CSCA provides Defect Lists containing information about defect or non-compliant document series, see Appendix A of [TR-03129]. The eID-Server MAY use Defect Lists for further indications of correct processing of eID-Documents.

2.4.6 Document Validation Procedure

Validation of an eID-Document is part of the General Authentication Procedure as specified in section 2.3.5 of [TR-03110], Part 2, and comprises Passive Authentication and Chip Authentication. To validate an eID-Document, the eID-Server SHALL perform the following steps as part of the General Authentication Procedure:

1. Perform Passive Authentication, i.e. validate the Document Security Object retrieved from the eID-Document as described in section 2.4.3;
2. Perform Chip Authentication;
3. Check if the eID Document is expired;
4. Check the revocation state of the eID Document via the Black List (see section 2.4.4).

Depending on the interface being used by the eService to communicate with the eID-Server the eID-Server MUST send an error code according to Section 3.4.1: Error Codes or show the documents validity status in the element described in Section 4.4.3: DocumentValidityResultType.
3 eID-Interface (SOAP)

The eID-Interface is a web service formally described in Web Services Description Language (WSDL). The corresponding WSDL-File is attached to this technical guideline as file TR-03130eID-Server.wsdl. The data types, attributes and elements used by the eID-Interface are described separately as an XML Schema Definition (XSD) in the XSD-File TR-03130eID-Server.xsd. Both files are formally attached to this technical guideline.

The general message flow, functions and data types, as well as the error handling and security measurements are described in the following sections.

3.1 General Message Flow

The following sections demonstrate the general message flow as it is intended by the author of this technical guideline. However participants of the eID-Infrastructure may differ from this sequence if (missing) technical dependencies allow them to do so.

3.1.1 Initiation

The following Figure 7: General message flow during initiation (SOAP) shows the steps of the message flow as it is described in [TR-03124] Part 1, Section 2.1: Communication Model.

![General message flow during initiation (SOAP)](image)

The above steps have to be completed for the initiation of the Online-Authentication and are part of the Service Provider’s scope of responsibilities. Each step is described in the following.

1. **Generate link to tcTokenURL**

   The eService SHALL generate a link for the user’s eID-Client containing the URL where the TC Token may be received according to [TR-03124] Part 1, Section 2.2: Client-Interface.

2. **Forward tcTokenURL to eID-Client**

   The user actively decides to use the eID-Function for Online-Authentication by clicking on the formerly generated link and thus forwarding the tcTokenURL to his local instance of the eID-Client.

3. **Call tcTokenURL**

   The eID-Client calls the tcTokenURL at the eService.
3.1.2 Interaction

The following Figure 8: General message flow during interaction (SOAP) shows the steps of the message flow at the eID-Interface that SHALL be implemented by the eService and the eID-Server and the context of the steps described in [TR-03124] Part 1, Section 2.5: Online-Authentication. In this description “Step 5. Call RefreshAddress” is used to trigger the call of the getResult function at the eID-Interface. The eService MAY as well use other appropriate events as a trigger.

![Diagram showing the general message flow during interaction (SOAP)](image)

The above steps have to be completed for the Online-Authentication and are part of the Service Provider's and the eID-Service's scope of responsibilities. Each step is described in the following.

1. **eID-Interface: useID Request**
   
   The eService calls the function `useID` (see Section 3.2.1.1: Request) at the eID-Interface of the eID-Server to request a valid session.

2. **eID-Interface: useID Response**
   
   The eID-Server responds to the call of the function `useID` (see Section 3.2.1.2: Response) and opens a new session if the request was legitimate.

3. **Transmit TC Token**
   
   The eService answers the former request of the eID-Client by sending the TC Token including the necessary connection parameters for the eID-Client in accordance to [TR-03124] Part 1, Section 2.5.1: Retrieval of TC Token.

4. **Establish secure authentication channel**
   
   The eID-Client establishes a secure channel to the eID-Server as described in [TR-03124] Part 1, Section 2.5.2: Connection Establishment.

5. **Call RefreshAddress**
   
   After the eCard-API-Framework specific communication between the eID-Client and the eID-Server has been finished the eID-Client re-connects to the RefreshAddress.

6. **eID-Interface: getResult Request**
   
   The eService calls the function `getResult` (see Section 3.2.2.1: Request) at the eID-Interface of the eID-Server to request the Online-Authentication's result.

7. **eID-Interface: getResult Response**
The eID-Server responds to the call of the function `getResult` (see Section 3.2.2.2: Response) by sending the results of the Online-Authentication to the eService.

### 3.1.3 Completion

The following Figure 9: General message flow during completion (SOAP) shows the steps of the message flow at the eID-Interface’s functions and the context of the steps described in [TR-03124] Part 1, Section 2.5.4 Return to the caller.

![Diagram of General message flow during completion (SOAP)](image)

The above steps have to be completed for the completion of the Online-Authentication and are part of the Service Provider’s scope of responsibilities. Each step is described in the following.

1. **Response from eService**
   
The eService answers the re-connect of the eID-Client.

2. **Forward Browser to RefreshAddress**
   
The eID-Client forwards the verified `RefreshAddress` to the Browser.

3. **Surf to RefreshAddress**
   
The Browser calls the `RefreshAddress` at the eService.

### 3.2 Functions

In the following section all functions of the eID-Interface the eID-Server SHALL offer are described. Technically this web service is described in the corresponding WSDL-File.
3 eID-Interface (SOAP)

3.2.1 useID

The function `useID` MUST be used by the eService to initialize a new Online-Authentication.

3.2.1.1 Request

The eService MUST use the call of the function `useID` to select operations the eID-Server SHALL perform with the user’s eID-Document. The operations consist of reading out selected data groups and performing age or place verification. If age or place verification is selected the eService MUST transmit additional parameters. The `useID` function SHALL only be called once during each session. Independently from the selected operations the eID-Server MUST perform a validity check of the eID-Document. The parameters of the function are shown in Figure 11: Function `useID` Request and Table 1: Function `useID` Parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>UseOperations</code></td>
<td>This parameter is used by the eService to define the data groups and functions the eID-Server SHALL try to read from the eID-Document.</td>
</tr>
<tr>
<td><code>AgeVerificationRequest</code></td>
<td>When an age verification request is executed, this parameter gives the year of life (Age) the eID-Document's owner is supposed to have completed.</td>
</tr>
</tbody>
</table>

Figure 10: Webservice overview

Figure 11: Function useID Request

Federal Office for Information Security
### 3.2.1.2 Response

The response to the function `useID` shows the eService if the eID-Server was able to open a session for the requested Online-Authentication. The return values of the function are shown in Figure 12: Function `useID` Response and Table 2: Function `useID` Return Values.

The eID-Interface allows a maximum number of simultaneously active sessions for each eService. It is advisable to set an appropriate value for that maximum so that heavy loading of individual eServices does not lead to restricted availability of other eServices. If the maximum number of simultaneous sessions for one eService is exceeded, the eID-Server MUST send an error message in the Result element with the `ResultMinor URI .../useID#tooManyOpenSessions` (see Section 3.4.1: Error Codes).

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session</td>
<td>The Session ID connects the <code>useID</code> function call with the <code>getResult</code> function call and MUST be present in this message. It MUST be uniquely</td>
</tr>
</tbody>
</table>
### 3.2.2 getResult

The function `getResult` MUST be used by the eService to request a result for an earlier call of the function `useID`.

#### 3.2.2.1 Request

![getResultRequest Diagram]

`getResultRequest` is the initial key for the encrypted channel between the eID-Client and the eID-Server as shown in Figure 10: Webservice overview. The eService MUST use exactly this PSK for the TC Token. If a PSK has already been transmitted to the eID-Server during the eService’s call to the function `useID`, the same PSK MUST be part of the eID-Server’s response. If the PSK sent in this message differs from the PSK sent in the request the eService SHALL abort the Online-Authentication.

A request’s result is retrieved via the `getResult` function. The parameters of the function are shown in Figure 13: Function `getResult` Request and Table 3: Function `getResult` Parameters.
The `getResult` function MAY potentially be called multiple times within one Online-Authentication. The eService MUST call the function within a Online-Authentication until it no longer gets the `.../getResult#noResultYet` error. If the request of the function `getResult` is answered with a message not containing `.../getResult#noResultYet` error the session SHALL no longer be used by the eService. The session SHOULD also expire if the time between the call of the function `useID` and the call of the function `getResult` exceeds a time limit defined by the eID-Server depending on its operational requirements.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session</td>
<td>MUST provide the Session ID of the Online-Authentication for this function call.</td>
</tr>
<tr>
<td>RequestCounter</td>
<td>A counter that MUST make function calls within an Online-Authentication uniquely identifiable. This value MUST be incremented by 1 for each <code>getResult</code> function call within an Online-Authentication. This procedure prevents replay attacks on the <code>getResult</code> function.</td>
</tr>
</tbody>
</table>

Table 3: Function `getResult` Parameters

3.2.2.2 Response

The response of the function `getResult` gives the eService the result of the requested Online-Authentication, if the result is present at the eID-Server. After the function has been executed without error, the session becomes invalid and the server MUST delete the queried data. If the call of the function `getResult` is answered by the eID-Server with a message other than containing the `noResultYet` error the session (ID) MUST be invalidated and SHALL no longer be accepted by the eID-Server. The session SHOULD also expire if the time between the call of the function `useID` and the call of the function `getResult` exceeds a time limit defined by the eID-Server depending on the operational requirements.

The return values of the function are shown in Figure 14: Function `getResult` Response and Table 4: Function `getResult` Return Values.

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PersonalData</td>
<td>MUST contain the data groups read from the eID-Document by the eID-Server, if the Online-Authentication was successfully processed.</td>
</tr>
<tr>
<td>Return Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FulfilsAgeVerification</td>
<td>If age verification was performed the eID-Server MUST include the element FulfilsAge Verification in the response of the function getResult. The value of the element SHALL be true if the eID-Document's owner has attained the requested age and SHALL be false if the eID-Document's owner has not yet attained the requested age.</td>
</tr>
<tr>
<td>FulfilsPlaceVerification</td>
<td>The place verification's result MUST be contained in this return value, if place verification was activated in the UseOperations parameter and the function call was successful.</td>
</tr>
<tr>
<td>OperationsAllowedByUser</td>
<td>Defines the data groups and functions that could effectively be read from the eID-Document after the terminal authorization certificate and the user's selection have been applied. The element MUST be present, if the Online-Authentication was successfully processed by the eID-Server.</td>
</tr>
<tr>
<td>Result</td>
<td>MUST show whether it was possible to process the request or an error has occurred. In particular, the error .../getResult#noResultYet may occur here (see Section 3.4.1: Error Codes).</td>
</tr>
</tbody>
</table>

Table 4: Function getResult Return Values

### 3.2.3 getServerInfo

The function getServerInfo MUST offer information about the eID-Server to the eService. The eService MAY check the eID-Server's configuration with the help of this

#### 3.2.3.1 Request

The function's parameter is an element of type nullType. This type contains no information at all and serves only to enable the function to be called correctly. It may be called by the eService at anytime and is therefore not included in the overview of the general message flow in Section 3.1: General Message Flow.

#### 3.2.3.2 Response
The return values of the function are shown in Figure 15: Function getServerInfo Response and Table 5: Function getServerInfo Return Values.

<table>
<thead>
<tr>
<th>Return Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerVersion</td>
<td>MUST show the version of the eID-Interface the eID-Server currently implements.</td>
</tr>
<tr>
<td>DocumentVerificationRights</td>
<td>MUST show the operations the eService MAY use with the currently configured terminal authorization certificate.</td>
</tr>
</tbody>
</table>

Table 5: Function getServerInfo Return Values

3.3 Data types

The data types used at the eID-Interface are described in this section. The data types involve mostly complex data types defined in the XML Schema Definition (XSD). The corresponding XSD-File TR-03130eID-Server.xsd is attached to this guideline.
3.3.1 SessionType

The type `SessionType` SHALL be used to uniquely identify a request to the eID-Interface. The eID-Server MUST generate values for the `ID` element to identify different requests for Online-Authentication. The value of the element `ID` MUST be random and at least 32 characters long in hexadecimal representation so that it’s not easy to guess. The software component generating the random numbers used as IDs SHALL fulfill the requirements described in [TR-03116-4] Section 6.2: Zufallszahlen and the eID-Service MUST ensure that no active sessions using identical IDs exist in the same context.

The eService SHALL only use valid session IDs for calls of the function `getResult`.

3.3.2 RestrictedIDType

The data type `RestrictedIDType` MUST contain a (from the Service Provider’s perspective) card-specific attribute as specified in [TR-03110] Part 2 Section 3.6: Restricted Identification. The `ID` is unique for each eService’s user and MAY thus be used by the eService to recognize an existing user. If the eService’s terminal authorization certificate contains a second terminal sector, a second `ID` is generated on the eID-Document’s chip and SHALL be contained in the element `ID2`.

![Figure 16: Data type SessionType](image)

![Figure 17: Data type RestrictedIDType](image)
3.3.3 PersonalDataType

The data type **PersonalDataType** describes the data that can be read from the eID-Document using the eID-Function. The data type's elements correspond to the eID-Function's data groups specified in [TR-03110] Part 4 Section 2.2.3: Data Groups and Generic Attributes. The data type **DocumentType** and **ICAOCountry** are derived from the simple type string and restrict the allowed characters as specified in [TR-03110] Part 4. The data types **GeneralDateType** and **GeneralPlaceType** are also basically XML translations of the data groups **Date** and **GeneralPlace** specified in [TR-03110] Part 4.

3.3.4 GeneralPlaceType

![Data type PersonalDataType diagram]

*Figure 18: Data type PersonalDataType*

The data type **PersonalDataType** describes the data that can be read from the eID-Document using the eID-Function. The data type's elements correspond to the eID-Function's data groups specified in [TR-03110] Part 4 Section 2.2.3: Data Groups and Generic Attributes. The data type **DocumentType** and **ICAOCountry** are derived from the simple type string and restrict the allowed characters as specified in [TR-03110] Part 4. The data types **GeneralDateType** and **GeneralPlaceType** are also basically XML translations of the data groups **Date** and **GeneralPlace** specified in [TR-03110] Part 4.
The data type `GeneralPlaceType` corresponds to the eID Application's GeneralPlace data group specified in [TR-03110] Part 4 Section 2.2.3: Data Groups and Generic Attributes. This data type describes various formats in which the user's addresses MAY exist. Especially the element `StructuredPlace` of the data type `PlaceType` (see Section 3.3.5: PlaceType) is common in this context.

### 3.3.5 PlaceType

The data type `PlaceType` corresponds to the eID Application's Place data group specified in [TR-03110] Part 4 Section 2.2.3: Data Groups and Generic Attributes. This data type is a structured representation of a person's residence.
3.3.6 OperationsSelectorType

The data type `OperationsSelectorType` represents the possible operations offered by the eID-Function taking into account the rights encoded in the configured terminal authorization certificate. Therefore it uses several elements of the data type `AttributeSelectionType` (see Section 3.3.15: `AttributeSelectionType`). The default value for all elements in this data type is PROHIBITED so only differing values have to be set explicitly. The names of the elements correspond to the eID Application's data groups specified in [TR-03110] Part 4 Section 2.2.3: Data Groups and Generic Attributes.

Figure 21: Data type OperationsSelectorType
3.3.7 OperationsRequestorType

The data type OperationsRequestorType allows the selection of operations offered by the eID-Function. Therefore it uses several elements of the data type AttributeRequestType (see Section 3.3.16: AttributeRequestType). The default value for all elements in this data type is PROHIBITED so only differing values have to be set explicitly. The names of the elements correspond to the eID Application's data groups specified in [TR-03110] Part 4 Section 2.2.3: Data Groups and Generic Attributes.

Figure 22: Data type OperationsRequestorType
3.3.8 OperationsResponderType

The data type OperationsResponderType represents the selected operations performed by the eID-Function based on the configured terminal authorization certificate and the potential further restriction by the user. Therefore it uses several elements of the data type AttributeResponseType (see Section 3.3.17: AttributeResponseType). The default value for all elements in this data type is PROHIBITED so only differing values have to be set explicitly. The names of the elements correspond to the eID Application's data groups specified in [TR-03110] Part 4 Section 2.2.3: Data Groups and Generic Attributes.

3.3.9 AgeVerificationRequestType
The data type \texttt{AgeVerificationRequestType} represents the parameter for age verification, which \textit{MAY} be requested using the function \texttt{useID}. The \texttt{Age} element \textit{MUST} contain the year of life that the eID-Document's owner is supposed to have attained.

### 3.3.10 PlaceVerificationRequestType

The data type \texttt{PlaceVerificationRequestType} contains the parameter for place verification, which \textit{MAY} be requested using the function \texttt{useID}. The \texttt{CommunityID} corresponds to the Wohnort-ID described in [TR-03127] and is compared with the CommunityID stored on the eID-Document's chip.
3.3.11 VersionType

![VersionType Diagram]

Figure 26: Data type VersionType

The data type VersionType describes the version of the eID-Interface. It contains a readable textual description in the VersionString element. The format of the text string is not predefined, but it SHOULD correspond to the content of the elements Major, Minor and Bugfix as well as it SHOULD contain a release date (see Section 3.6.3: Call of Function getResponse for an example). The Major element MUST identify the main version and the Minor element SHALL be used for the sub-versions. An optional Bugfix identifier MAY also be transmitted here. This identifies specification corrections that have already been implemented on the eID-Server. Changes of the Major and Minor version are usually not backwards compatible.

3.3.12 VerificationResultType

![VerificationResultType Diagram]

Figure 27: Data type VerificationResultType

The data type VerificationResultType represents the result of a verification performed on the eID-Document. The element FulfilsRequest indicates whether or not the request ended with a positive result.

3.3.13 PreSharedKeyType
The data type **PreSharedKeyType** contains the randomly generated Pre-Shared Key (PSK) itself in the element **Key** and a unique identifier in the element **ID**. The PSK is required to initialize the communication as specified in [TR-03112] Part 7, Section 2.4.1.2: TLS with pre-shared keys. The identifier SHALL be used as **psk_identity** for the communication mentioned above and MUST therefore be encoded to the element **SessionIdentifier** in the TC Token.

### 3.3.14 GeneralDateType

The data type **GeneralDateType** MUST contain the date representation read directly from the chip in the **DateString** element. An 8-place string SHALL be used here that MUST contain spaces as well as numerical values (0-9). It's transmitted in the yyyyymmdd format (4 places for the year, 2 places for the month and 2 places for the day) as specified in [TR-03110] Part 2 Annex A: ASN.1 Specifications. The optional element **DateValue** of the simple type **date** MUST only be present if the eID-Server was able to read a complete date from the eID Application.

### 3.3.15 AttributeSelectionType

The data type **AttributeSelectionType** is derived from the simple data type **string**. It allows the following values:

- ALLOWED
- PROHIBITED

The data type is used to represent the selection of individual attributes with the aid of the data type **OperationsSelectorType** (see Section 3.3.6: **OperationsSelectorType**).
3.3.16 AttributeRequestType

The data type **AttributeRequestType** is derived from the simple data type **string**. It allows the following values:

- ALLOWED
- PROHIBITED
- REQUIRED

The data type is used to represent the selection of individual attributes with the aid of the data type **OperationsRequestorType** (see Section 3.3.7: OperationsRequestorType). Requested attributes marked with the value **REQUIRED** MUST be used for the RequiredCHAT according to [TR-03112] Part 7. Requested attributes marked with the value **ALLOWED** MUST be used for the OptionalCHAT according to [TR-03112] Part 7. Requested attributes marked with the value **PROHIBITED** MUST NOT be requested.

3.3.17 AttributeResponseType

The data type **AttributeResponseType** is derived from the simple data type **string**. It allows the following values:

- ALLOWED
- PROHIBITED
- NOTONCHIP

The data type is used to represent the availability of individual attributes with the aid of the data type **OperationsResponderType** (see Section 3.3.8: OperationsResponderType).

The value **PROHIBITED** MUST be used for attributes that could not be read because of missing rights in the terminal authorization certificate and/or the user has restricted the document reading rights. The value **NOTONCHIP** MUST be used in the response to identify requested attributes that could not be read from the eID-Document, because the eID-Document does not support this attribute.

3.4 Error Handling

![Figure 30: Element Result](image)

Error conditions that arise while the eID-Server processes a request MUST be reported to the eService. Therefore the element **Result** as specified in [DSS] and referenced in [TR-03112] Part 1, Section 4.1.2 SHALL be used.
In addition to the error codes described in [TR-03112] Part 1, Section 4.2 the eID-Server MUST use the URI
http://www.bsi.bund.de/eid/server/2.0/resultminor/
in the element ResultMinor to indicate errors that arise at the eID-Server.
The element ResultMajor MUST NOT contain other values than specified in [TR-03112] Part 1, Section 4.1.2.
The optional element ResultMessage MAY be used to deliver a textual description of the error.

3.4.1 Error Codes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description / Conditions</th>
</tr>
</thead>
</table>
| .../common#internalError    | **Internal error**
This error code MUST be used, IF an error occurs, that can not be mapped with any of the following more detailed error codes.                                                                                       |
| .../common#schemaViolation  | **Schema violation**
This error code MUST be used, IF the eService’s request does not validate to the schema used by the eID-Server.                                                                                                    |
| .../useID#invalidPSK       | **Initial PSK is invalid**
This error code MUST be used, IF the PSK transmitted from the eService to the eID-Server is invalid and can not be processed.                                                                                      |
| .../useID#tooManyOpenSessions | **Maximum number of sessions is reached**
This error code MUST be used, IF the request can not be processed due to excessive eID-Server loading.                                                                                                           |
| .../useID#missingArgument  | **Parameters are missing**
This error code MUST be used, IF the AgeVerification or PlaceVerification function was selected but the corresponding Request elements are missing.                                                            |
| .../useID#missingTerminalRights | **Necessary permissions are missing**
This error code MUST be used, IF the permissions necessary to process the request are missing in the terminal authorization certificate.                                                                               |
| .../getResult#noResultYet | **Request still has not been completed**
This error code MUST be used, IF the eID-Server does not have the result of the request yet. The function call may succeed at a later time.                                                                       |
| .../getResult#invalidSession | **Used session ID is invalid**
This error code MUST be used, IF the session has expired or was replied to and has thus been deleted.                                                                                                           |
| .../getResult#invalidCounter | **RequestCounter incremented incorrectly**
This error code MUST be used, IF the request’s requestCounter value is invalid (e.g. less or equal to that of a previous request) and will not be processed.                                |
### Error Codes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description / Conditions</th>
</tr>
</thead>
</table>
| .../getResult#invalidDocument  | **Used eID-Document is invalid**  
This error code MUST be used, IF passive authentication of the eID-Document or the revocation list check has determined that the eID-Document used is invalid.  |

Table 6: List of Error Codes

### 3.5 Security Measures

![Communication channel overview (SOAP)](image)

This section describes the Security Measures the eService and the eID-Server MUST take to ensure secure communication at the eID-Interface. The eID-Interface is represented by the arrow in *Figure 31: Communication channel overview (SOAP)* connecting the eService and the eID-Server. Encryption at the transport level and signature of the XML-Messages are used to guarantee the confidentiality, integrity and authenticity of the transmitted data and function calls.

#### 3.5.1 Encryption

Because the eService and the eID-Server are directly connected with each other and no third entity is involved in the communication, encryption at the transport level is sufficient. IF the eService and the eID-Server communicate through an otherwise open network (e.g. Internet), they MUST communicate using TLS-secured connections in compliance to [TR-03116-4] Section 2: Vorgaben für SSL/TLS. Due to the trust relationship between the eService and the eID-Server, Client- and Server-Authentication must be implemented. The connection MUST be refused by the eID-Server, IF the Client-Authentication fails.

#### 3.5.2 Signature

The eService and the eID-Server MUST apply XML-Signatures to all XML-Messages that are sent using the eID-Interface. Therefore the *InitiatorToken* and a *RecipientToken* as specified by the web service described in the file TR-03130eID-Server.wsdl MUST be used. The algorithm suites used for XML-Signatures MUST comply to the requirements of [TR-03116-4] Section 4.3: XML-Signature. Currently this means one of the following suites MUST be used:
3 eID-Interface (SOAP)

- Basic256Sha256
- Basic192Sha256
- Basic128Sha256

XML-Messages received by the eID-Server with invalid XML-Signatures MUST be answered with the error code .../common#internalError only.

3.5.2.1 InitiatorToken

Example 1: InitiatorToken

The **InitiatorToken** is a X.509 certificate that MUST be used by the eService to sign each request to the eID-Interface of the eID-Server. The eID-Server SHALL check the signature's validity and separate different clients with the aid of their X.509 certificates. The eID-Server MUST NOT answer to requests signed with an invalid X.509 certificate or not signed at all.

3.5.2.2 RecipientToken

Example 2: RecipientToken

The **RecipientToken** is the X.509 certificate that MUST be used by the eID-Server to sign each response send to the eService using the eID-Interface. The eService SHALL check the signature's validity and SHALL not process data that is signed with an invalid X.509 certificate or not signed at all.

3.5.3 Session Binding

The eService MUST use the **PSK** negotiated at the eID-Interface (see Section 3.2.1.2: Response) for the creation of the TC Token as specified in [TR-03124] Part 1, Section 2.4: TC Token. Further measures which SHALL be implemented for channel binding, but do not directly affect the eID-Interface, are described in
Part 1, Section 2.7: Session Binding. Both are necessary to reach the final goal of binding the channel TLS-1 between the Browser and the eService to the channel TLS-2 between the eID-Client and the eID-Server represented in Figure 31: Communication channel overview (SOAP).

The eService SHALL only accept and process valid responses from the eID-Server that are bound to an existing web-session. The eID-Server SHALL only accept and process valid requests from authorized eServices and MUST ensure only valid requests at the eCard-API Interface are accepted. For this purpose the eID-Server MUST verify that each SessionIdentifier of a PSK used by an eID-Client to initialize eCard-API-Framework communication matches a PSK ID previously negotiated at the eID-Interface for one specific session and a secure connection can be established according to [TR-03112] Part7, Section 2.3.1: Setting up a Trusted Channel.

3.6 Examples

The example in the following sections illustrates the message flow as specified in Section 3.1: General Message Flow. Because the eService waited for the eID-Client to re-connect the first call of the function getResult is already successful.

3.6.1 Call of Functions useID

The following sample code shows a useID function call. In the example, the eService mandatorily asks for all operations except reading out the ArtisticName and AcademicTitle, which are marked as optional, by calling the function useID. The ResidencePermitI is not requested.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:eid="http://bsi.bund.de/eID/">
<soapenv:Header />
<soapenv:Body>
<eid:useIDRequest>
<eid:useIDRequest>
<eid:UseOperations>
<eid:DocumentType>REQUIRED</eid:DocumentType>
<eid:IssuingState>REQUIRED</eid:IssuingState>
<eid:DateOfExppiry>REQUIRED</eid:DateOfExppiry>
<eid:GivenNames>REQUIRED</eid:GivenNames>
<eid:FamilyNames>REQUIRED</eid:FamilyNames>
<eid:ArtisticName>ALLOWED</eid:ArtisticName>
<eid:AcademicTitle>ALLOWED</eid:AcademicTitle>
<eid:DateOfBirth>REQUIRED</eid:DateOfBirth>
<eid:PlaceOfBirth>REQUIRED</eid:PlaceOfBirth>
<eid:Nationality>REQUIRED</eid:Nationality>
<eid:BirthName>REQUIRED</eid:BirthName>
<eid:PlaceOfResidence>REQUIRED</eid:PlaceOfResidence>
<eid:ResidencePermitI />
<eid:RestrictedID>REQUIRED</eid:RestrictedID>
<eid:AgeVerification>REQUIRED</eid:AgeVerification>
<eid:PlaceVerification>REQUIRED</eid:PlaceVerification>
</eid:UseOperations>
<eid:AgeVerificationRequest>
<eid:Age>18</eid:Age>
</eid:AgeVerificationRequest>
<eid:PlaceVerificationRequest>
<eid:CommunityID>027605</eid:CommunityID>
</eid:PlaceVerificationRequest>
</eid:useIDRequest>
</soapenv:Body>
</soapenv:Envelope>
```
Example 3: useIDRequest

The response of the eID-Server contains the Session and PSK elements. The Result element contains no error message in this case.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:eid="http://bsi.bund.de/eID/"
xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema">
  <soapenv:Header />
  <soapenv:Body>
    <eid:useIDResponse>
      <eid:Session>
        <eid:ID>1234567890abcdef1234567890abcdef</eid:ID>
      </eid:Session>
      <eid:PSK>
        <eid:ID>0987654321abcdef1234567890abcdef</eid:ID>
        <eid:Key>fedcba0987654321fedcba0987654321</eid:Key>
      </eid:PSK>
      <dss:Result>
        <ResultMajor>
          http://www.bsi.bund.de/ecard/api/1.1/resultmajor#ok
        </ResultMajor>
      </dss:Result>
    </eid:useIDResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 4: useIDResponse

3.6.2 Call of Function getResult

The following code example shows the getResult call by the eService.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:eid="http://bsi.bund.de/eID/"
xmlns:eid="/">
  <soapenv:Header />
  <soapenv:Body>
    <eid:getResultRequest>
      <eid:Session>
        <eid:ID>1234567890abcdef1234567890abcdef</eid:ID>
      </eid:Session>
      <eid:RequestCounter>1</eid:RequestCounter>
    </eid:getResultRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 5: getResultRequest

Because the results are available at the eID-Server, this call delivers the matching return values. Transmission of the artistic name was deselected by the user (see OperationsAllowedByUser element in the example). Additionally the eID-Server was not able to read the BirthName because this data group was not available on the eID-Document's chip. How the eService deals with the user's and document's restrictions is up to the Service Provider.
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:eid="http://bsi.bund.de/eID/
xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema" >
  <soapenv:Header />
  <soapenv:Body>
    <eid:getResultResponse>
      <eid:PersonalData>
        <eid:DocumentType>ID</eid:DocumentType>
        <eid:IssuingState>D</eid:IssuingState>
        <eid:DateOfExpiry>2020-10-31</eid:DateOfExpiry>
        <eid:GivenNames>ERIKA</eid:GivenNames>
        <eid:FamilyNames>MUSTERMANN</eid:FamilyNames>
        <eid:AcademicTitle />
        <eid:DateOfBirth>
          <eid:DateString>19640812</eid:DateString>
          <eid:DateValue>1964-08-12</eid:DateValue>
        </eid:DateOfBirth>
        <eid:PlaceOfBirth>
          <eid:FreetextPlace>BERLIN</eid:FreetextPlace>
        </eid:PlaceOfBirth>
        <eid:Nationality>D</eid:Nationality>
        <eid:PlaceOfResidence>
          <eid:StructuredPlace>
            <eid:Street>HEIDESTRASSE 17</eid:Street>
            <eid:City>KÖLN</eid:City>
            <eid:Country>D</eid:Country>
          </eid:StructuredPlace>
        </eid:PlaceOfResidence>
        <eid:RestrictedID>
          <eid:ID>01A4FB509CEBC6595151A4FB5F9C75C6FE01A4FB59CB655A4FB5F9C75C6FEE</eid:ID>
          <eid:ID2>5C6FE01A4FB59CB655A4FB5F9C75C6FE01A4FB509CEBC6595151A4FB5F9C7</eid:ID2>
        </eid:RestrictedID>
      </eid:PersonalData>
      <eid:FulfilsAgeVerification>
        <eid:FulfilsRequest>true</eid:FulfilsRequest>
      </eid:FulfilsAgeVerification>
      <eid:FulfilsPlaceVerification>
        <eid:FulfilsRequest>true</eid:FulfilsRequest>
      </eid:FulfilsPlaceVerification>
      <eid:OperationsAllowedByUser>
        <eid:DocumentType>ALLOWED</eid:DocumentType>
        <eid:IssuingState>ALLOWED</eid:IssuingState>
        <eid:DateOfExpiry>ALLOWED</eid:DateOfExpiry>
        <eid:GivenNames>ALLOWED</eid:GivenNames>
        <eid:FamilyNames>ALLOWED</eid:FamilyNames>
        <eid:AcademicTitle />
        <eid:DateOfBirth>ALLOWED</eid:DateOfBirth>
        <eid:PlaceOfBirth>ALLOWED</eid:PlaceOfBirth>
        <eid:Nationality>ALLOWED</eid:Nationality>
        <eid:BirthName>NOTONCHIP</eid:BirthName>
        <eid:PlaceOfResidence>ALLOWED</eid:PlaceOfResidence>
      </eid:OperationsAllowedByUser>
    </eid:getResultResponse>
  </soapenv:Body>
</soapenv:Envelope>
3 eID-Interface (SOAP)

```xml
<eid:ResidencePermitI />
<eid:RestrictedID>ALLOWED</eid:RestrictedID>
<eid:AgeVerification>ALLOWED</eid:AgeVerification>
<eid:PlaceVerification>ALLOWED</eid:PlaceVerification>
</eid:OperationsAllowedByUser>
<dss:Result>
<ResultMajor>
http://www.bsi.bund.de/ecard/api/1.1/resultmajor#ok
</ResultMajor>
</dss:Result>
</eid:getResultResponse>
</soapenv:Body>
</soapenv:Envelope>

Example 6: getResultResponse

3.6.3 Call of Function getResponse

The following code example shows the getServerInfo call by the eService.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
xmlns:eid="http://bsi.bund.de/eID/">
<soapenv:Header />
<soapenv:Body>
<getServerInfoRequest />
</soapenv:Body>
</soapenv:Envelope>
```

Example 7: getServerInfoRequest

In addition to the version of the eID-Interface which the eID-Server currently supports (see element ServerVersion), the rights of the terminal authorization certificate currently stored on the eID-Server (see element DocumentVerificationRights) are included in the response.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
xmlns:eid="http://bsi.bund.de/eID/">
<soapenv:Header />
<soapenv:Body>
<eid:getServerInfoResponse>
<eid:ServerVersion>
<eid:VersionString>
Version 2.2.0 16.12.2019
</eid:VersionString>
<eid:Major>2</eid:Major>
<eid:Minor>2</eid:Minor>
<eid:Bugfix>0</eid:Bugfix>
</eid:ServerVersion>
<eid:DocumentVerificationRights>
<eid:DocumentType>ALLOWED</eid:DocumentType>
<eid:IssuingState>ALLOWED</eid:IssuingState>
<eid:DateOfExpiry>ALLOWED</eid:DateOfExpiry>
<eid:GivenNames>ALLOWED</eid:GivenNames>
<eid:FamilyNames>ALLOWED</eid:FamilyNames>
<eid:ArtisticName>ALLOWED</eid:ArtisticName>
<eid:AcademicTitle>ALLOWED</eid:AcademicTitle>
</soapenv:Body>
</soapenv:Envelope>
```
<eid:DateOfBirth>ALLOWED</eid:DateOfBirth>
<eid:PlaceOfBirth>ALLOWED</eid:PlaceOfBirth>
<eid:Nationality>ALLOWED</eid:Nationality>
<eid:BirthName>ALLOWED</eid:BirthName>
<eid:PlaceOfResidence>ALLOWED</eid:PlaceOfResidence>
<eid:RestrictedID>ALLOWED</eid:RestrictedID>
<eid:AgeVerification>ALLOWED</eid:AgeVerification>
<eid:PlaceVerification>ALLOWED</eid:PlaceVerification>
</eid:DocumentVerificationRights>
</soapenv:Body>
</soapenv:Enveloped

Example 8: getServerInfoResponse
4 SAML-Profile

The Security Assertion Markup Language defined in [SAML] is a recognized standard in an identity provider's and in the international context. However, SAML cannot be applied to the eID-Interface directly, because it contradicts two fundamental principles of the eID-Function.

The authentication protocols implemented by the eID-Document assume exactly two entities communicating with each other. These entities represent on the one side the citizen (with his eID-Document) and on the other side the Service Provider (with his terminal authorization certificate). An additional authentication protocol principle is that the authentication's security results from the eID-Document and the protocols it implements, not from the surrounding infrastructure.

By default SAML protocols assume a three-entity model and are based on the trust relationship between two of them (Identity Provider and Service Provider).

However, it is possible to use SAML so that the principles of the authentication protocol remain intact and a secure, legally compliant authentication can be achieved. The basis for this is that the Service Provider MAY delegate Online-Authentication to be performed by a third party.

This section defines a specific SAML-Profile which considers the design decisions made for the eID-Infrastructure. This SAML-Profile SHOULD be implemented by the eID-Service if SAML is offered as an alternative to the eID-Interface specified in Section 3: eID-Interface (SOAP). In this case, this alternative SHALL be provided by the "SAML Processor" as described in the following section.

4.1 Basic Commitments

The basis for the SAML-Profile implemented by the eID-Service MUST be the Web Browser Single Sign-On (Web Browser SSO) profile specified in [SAML] Profiles, Section 4.1: Web Browser SSO Profile. Because the eService and the eID-Service MUST use the transport mechanisms specified in [TR-03124] Part 1, Section 2.6: SAML the HTTP-Redirect Binding SHALL be implemented. The SAML messages MUST be transferred URL-Encoded in the Location header field of the redirect (see Section 4.9.2.1: URL-Encoding for an example).

For usage with this profile the Authentication Request Protocol specified in [SAML] Assertions and Protocols, Section 3.4: Authentication Request Protocol MUST be used as described in this section. The following table shows the mapping of the roles defined in [SAML] to the roles of the eID-Infrastructure that appear in this context.

<table>
<thead>
<tr>
<th>eID-Infrastructure</th>
<th>SAML Specification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>User</td>
<td>The user uses the components Browser and eID-Client for contacting and authenticating at the Service Provider as well as verifying the Service Provider's identity.</td>
</tr>
<tr>
<td>eID-Client</td>
<td>User Agent</td>
<td>The eID-Client offers the functionality of the User Agent and guides the user through the process of Online-Authentication.</td>
</tr>
<tr>
<td>Service Provider</td>
<td>Service Provider (SP)</td>
<td>The Service Provider is also called Relying Party and uses the eID-Service for verifying the user's identity and authenticating himself.</td>
</tr>
<tr>
<td>eID-Service</td>
<td>Identity Provider (IdP)</td>
<td>The eID-Service performs Online-Authentication on behalf of the Service Provider.</td>
</tr>
</tbody>
</table>

Table 7: Mapping of eID-Infrastructure and SAML Specification roles
4.2 General Message Flow

The SAML-Profile for the eID-Server uses exactly the communication mechanisms described in [TR-03124] Part 1, Section 2.6: SAML and MUST be implemented complying. The general communication flow is described in the following sections.

4.2.1 Initiation

The following Figure 32: General message flow during initiation (SAML) shows the steps of the message flow as it is described in [TR-03124] Part 1, Section 2.6: SAML.

The above steps have to be completed for the initiation of the Online-Authentication. While step 1 to 3 are part of the Service Provider's scope of responsibilities the steps 4 and 5 also rely on the secure communication channel between the eService and the SAML Processor (see Section 4.9: Security Measures). Each step is described in the following.

1. **Generate link to presumed tcTokenURL**

   The eService MUST generate a link for the user's eID-Client containing the presumed URL where the TC Token SHALL be received according to [TR-03124] Part 1, Section 2.2: Client-Interface.

2. **Forward presumed tcTokenURL to eID-Client**

   The user decides to use the eID-Function for Online-Authentication by actively clicking on the formerly generated link and thus forwarding the presumed tcTokenURL to his locally running instance of the eID-Client.

3. **Call tcTokenURL at eService**

   The eID-Client calls the presumed tcTokenURL at the eService.
4. Redirect to SAML Processor incl. AuthnRequest

In this context the eService MUST answer the call of the eID-Client with a redirect to the SAML Processor according to [TR-03124] Part 1, Section 2.5.1: Retrieval of the TC Token. The AuthnRequest (see Section 4.7.1: AuthnRequest) SHALL be included in the Location field of the redirect (see Section 4.9.2.1: URL-Encoding for an example).

5. Call of SAML Processor incl. AuthnRequest

Following the redirect from the eService to the SAML Processor the eID-Client calls the SAML Processor according to [TR-03124] Part 1, Section 2.6: SAML with the URL-Encoded (see Section 4.9.2.1: URL-Encoding for an example) AuthnRequest (see Section 4.7.1: AuthnRequest).

4.2.2 Interaction

The following Figure 33: General message flow during interaction (SAML) shows the steps of the message flow of the eID-Interface's functions in the context of an existing SAML Processor as an example of how the SAML Processor MAY communicate with the eID-Client and the eID-Server. The SAML Processor MAY also be operated attached to the eID-Server thus using just one communication channel to the eID-Client.

The above steps MUST be completed for the Online-Authentication and are part of the eID-Service's scope of responsibilities. The steps itself are not described in this section. They are equivalent to the steps described in Section 3.1.2: Interaction with the SAML Processor fulfilling the part of the eService in this context and MUST be performed accordingly. Implementation of the eID-Interface according to Section 3: eID-Interface (SOAP) of this technical guideline is not required in this scenario.
4.2.3 Completion

The following Figure 34: General message flow during completion (SAML) shows the steps of the message flow as described in [TR-03124] Part 1, Section 2.5.4: Return to the caller.

The above steps MUST be performed for the completion of the Online-Authentication. While step 1 and 2 rely on the secure communication channel between the eService and the SAML Processor (see Section 4.9: Security Measures) the steps 3 to 5 are part of the eService’s scope of responsibilities. Each step is described in the following.

1. Redirect from SAML Processor incl. Response
The SAML Processor answers the call of the eID-Client with a redirect to the eService. This redirect SHALL include the URL-Encoded (see Section 4.9.2.1: URL-Encoding for an example) Response (see Section 4.7.3: Response) in the Location field.

2. Call of eService incl. Response
The eID-Client calls the eService with the URL-Encoded (see Section 4.9.2.1: URL-Encoding for an example) Response (see Section 4.7.3: Response).

3. Response from eService
The eService answers the call from the eID-Client.

4. Forward Browser to eService
The eID-Client forwards the Browser to the verified URL of the eService.

5. Calling the eService
The Browser calls the URL of the eService and returns to the web-session.
4.3 Attributes

In the following section the attributes that SHALL be used in the context of eID-Documents are listed. The SAML-Profile has a more modular design compared to the eID-Interface. Therefore the attribute’s names are needed to uniquely identify the relevant attributes. Additional attributes MAY be used if specified by the Service Provider and the eID-Service.

The tables also contain information about the data types that MUST be used to transmit values upon requests and responses. The data type is either a built-in primitive type, identified below by the ‘xs:’ prefix, or a data type defined in the XSD-File TR-03130eID-Server.xsd attached to this technical guideline represented with the ‘eid:’ prefix.

4.3.1 Request Attributes

The following table contains all attributes the Service Provider may request from the eID-Service. Attributes marked with "-" do not require input values. For these attributes input values must not be transmitted in the request. The eID-Server MUST try to perform the operations requested by the eService IF the terminal authorization certificate of the eService allows the eID-Server to do so. The eID-Document’s validity MUST be checked by the eID-Service during every Online-Authentication.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Data type used for value in request</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocumentValidity</td>
<td>-</td>
</tr>
<tr>
<td>DocumentType</td>
<td>-</td>
</tr>
<tr>
<td>IssuingState</td>
<td>-</td>
</tr>
<tr>
<td>DateOfExpiry</td>
<td>-</td>
</tr>
<tr>
<td>GivenNames</td>
<td>-</td>
</tr>
<tr>
<td>FamilyNames</td>
<td>-</td>
</tr>
<tr>
<td>ArtisticName</td>
<td>-</td>
</tr>
<tr>
<td>AcademicTitle</td>
<td>-</td>
</tr>
<tr>
<td>DateOfBirth</td>
<td>-</td>
</tr>
<tr>
<td>PlaceOfBirth</td>
<td>-</td>
</tr>
<tr>
<td>Nationality</td>
<td>-</td>
</tr>
<tr>
<td>BirthName</td>
<td>-</td>
</tr>
<tr>
<td>PlaceOfResidence</td>
<td>-</td>
</tr>
<tr>
<td>ResidencePermitI</td>
<td>-</td>
</tr>
<tr>
<td>RestrictedID</td>
<td>-</td>
</tr>
<tr>
<td>AgeVerification</td>
<td>eid:AgeVerificationRequestType</td>
</tr>
<tr>
<td>PlaceVerification</td>
<td>eid:PlaceVerificationRequestType</td>
</tr>
</tbody>
</table>

Table 8: SAML Request Attributes

4.3.2 Response Attributes

The following table contains all attributes the eID-Service MAY transmit to the Service Provider depending on the request from the Service Provider. The same names that have been used in the request SHALL be
used here. The eID-Document's validity MUST be checked by the eID-Service during every Online-Authentication. The result therefore MUST be included in every response.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Data type used for value in response</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocumentValidity</td>
<td>eid:DocumentValidityResultType</td>
</tr>
<tr>
<td>DocumentType</td>
<td>eid:DocumentType</td>
</tr>
<tr>
<td>IssuingState</td>
<td>eid:ICAOCountry</td>
</tr>
<tr>
<td>DateOfExpiry</td>
<td>xs:date</td>
</tr>
<tr>
<td>GivenNames</td>
<td>xs:string</td>
</tr>
<tr>
<td>FamilyNames</td>
<td>xs:string</td>
</tr>
<tr>
<td>ArtisticName</td>
<td>xs:string</td>
</tr>
<tr>
<td>AcademicTitle</td>
<td>xs:string</td>
</tr>
<tr>
<td>DateOfBirth</td>
<td>eid:GeneralDateType</td>
</tr>
<tr>
<td>PlaceOfBirth</td>
<td>eid:GeneralPlaceType</td>
</tr>
<tr>
<td>Nationality</td>
<td>eid:ICAOCountry</td>
</tr>
<tr>
<td>BirthName</td>
<td>xs:string</td>
</tr>
<tr>
<td>PlaceOfBirth</td>
<td>eid:GeneralPlaceType</td>
</tr>
<tr>
<td>ResidencePermitI</td>
<td>xs:string</td>
</tr>
<tr>
<td>RestrictedID</td>
<td>eid:RestrictedIDType</td>
</tr>
<tr>
<td>AgeVerification</td>
<td>eid:AgeVerificationResultType</td>
</tr>
<tr>
<td>PlaceVerification</td>
<td>eid:PlaceVerificationResultType</td>
</tr>
</tbody>
</table>

Table 9: SAML Response Attributes

4.4 Data types

In addition to the general data types described in Section 3.3: Data types several SAML-specific data types are specified in the XML-Schema TR-03130eID-Server.xsd and SHALL be used as described in the following section.
4.4.1 PlaceVerificationResultType

This data type SHALL be used for transmitting the result of a place verification from the SAML Processor to the eService in an assertion. The Request element contains the requested comparison value (CommunityID) and the Result element the verification’s result (FulfilsRequest).

4.4.2 AgeVerificationResultType

This data type SHALL be used for transmitting the result of an age verification from the SAML Processor to the eService in an assertion. The Request element contains the requested comparison value (Age) and the Result element the verification’s result (FulfilsRequest).
4.4.3 DocumentValidityResultType

This data type SHALL be used for transmitting information about the version of the SAML Processors interface and the validity of the eID-Document used for the specific authentication.

<table>
<thead>
<tr>
<th>Attribute/Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Indicates the version of the data representation the eID-Service implements.</td>
</tr>
<tr>
<td></td>
<td>Version '1' SHOULD typically be used here.</td>
</tr>
<tr>
<td></td>
<td>Note that this version relates only to the content and semantics of the authentication response's data. The version of the schema itself SHALL be managed through other mechanisms such as schema namespace.</td>
</tr>
<tr>
<td>ReferenceDate</td>
<td>The exact date the eID-Document's validity (expiry date and revocation status) was determined. In principle this should be the date of the request, but this value may be needed under certain conditions (e.g. day change during Online-Authentication).</td>
</tr>
<tr>
<td>Status</td>
<td>MUST contain the result of the document validation test using the following values: <code>valid</code>: Document is valid. <code>failed</code>: Test has failed. Additionally, the following values MAY be implemented: <code>expired</code>: Document has expired. <code>revoked</code>: Document is revoked. <code>notAuthentic</code>: Document is not authentic. If the document isn’t valid (the status is other than <code>valid</code>), none of the attributes to be read from the eID-Document MAY be transmitted.</td>
</tr>
</tbody>
</table>

Table 10: Attributes and Elements of the Data type DocumentValidityResultType
4.4.4 RequestedAttributesType

This data type contains the requested attributes and the request values for verification functions. It MAY contain several attributes from the data type AttributeType specified in [SAML]. The identifiers specified in Section 4.3.1: Request Attributes SHALL be used for the attribute Name. Additional attributes with other identifiers MAY be used if specified by the Service Provider and the eID-Service.

Each attribute element contained in the AuthnRequestExtension MAY contain the attribute RequiredAttribute to identify mandatory fields. This attribute is also defined by the schema (see Section 4.6.1: RequiredAttribute). If the attribute RequiredAttribute is not present, the eID-Service SHALL assume the requested attribute is required for the specific use case.

Each attribute element contained in the Assertion MAY contain the attribute AttributeNotOnChip to identify data groups not supported by the eID-Document’s chip. This attribute is also defined by the schema (see Section 4.6.2: AttributeNotOnChip). If the attribute AttributeNotOnChip is not present, the eService SHALL assume the requested attribute is present on the eID-Document’s chip.

4.4.5 AuthnRequestExtensionType

This data type is the basis for the authentication request. In addition to the version of the interface it contains a list of the attributes requested by the eService.

<table>
<thead>
<tr>
<th>Attribute/Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Indicates the version of the data representation the Service Provider implements.</td>
</tr>
<tr>
<td>Attribute/Element</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Version ‘2’</td>
<td>SHOULD typically be used here. Note that this version relates only to the content and semantics of the authentication request’s data. The version of the schema itself SHALL be managed through other mechanisms such as using the schema namespace.</td>
</tr>
</tbody>
</table>

RequestedAttributes

The list of attributes that are requested containing additional input if needed for verification functions.

Table 11: Attribute and Elements of the Data type AuthnRequestExtensionType

4.5 Additional Elements

The following elements are used for the SAML Messages (see Section 4.7: SAML Messages) and have been defined in the XSD-File TR-03130eID-Server.xsd attached to this technical guideline.

4.5.1 AuthnRequestExtension

![Element AuthnRequestExtension](image)

Figure 40: Element AuthnRequestExtension

The AuthnRequestExtension element is of the AuthnRequestExtensionType data type defined in this schema (see Section 4.4.5: AuthnRequestExtensionType) and includes the transport information for the AuthnRequest.
4.5.2 EncryptedAuthnRequestExtension

The `EncryptedAuthnRequestExtension` is of the `EncryptedElementType` data type defined in [SAML]. It MUST contain the encrypted `AuthnRequestExtension` (see Section 4.4.5: `AuthnRequestExtensionType`) in the `EncryptedData` element as specified in [XML-Enc].

4.6 Additional Attributes

Some SAML components allow the definition of specific attributes for a use case using the `anyAttribute` placeholder. The attributes in this section have been specified in the XSD-File TR-03130eID-Server.xsd attached to this technical guideline for such cases.

4.6.1 RequiredAttribute

The attribute `RequiredAttribute` uses the simple data type `boolean` without further restrictions. It MAY be used in the request to identify attributes as mandatory (`true`) or optional (`false`). The default value for this attribute is mandatory (`true`). Requested attributes marked with this attribute MUST be used for the RequiredCHAT according to [TR-03112] Part 7. Requested attributes that are not marked with this attribute MUST be used for the OptionalCHAT according to [TR-03112] Part 7.

4.6.2 AttributeNotOnChip

The attribute `AttributeNotOnChip` uses the simple data type `boolean` without further restrictions. It MUST be used in the response to identify attributes that could not be read from the eID-Document, because the eID-Document does not support this attribute (`true`). It MUST NOT be used for attributes that could not be read because of missing rights in the terminal authorization certificate and/or the user has restricted the document reading rights. Generally it is assumed that the requested attribute is supported by the eID-Document. Therefore the default and expected (e.g. if the attribute is not present) value for this attribute is `false`.

4.7 SAML Messages

The following sections define individual SAML Messages that MUST be used to meet this SAML-Profile. The scenario described in Section 4.1: Basic Commitments as well as the overview described in Section 4.2: General Message Flow and the data types and elements defined in Section 4.4: Data types form the basis for this messages.
4.7.1 AuthnRequest

The AuthnRequest must contain the following XML elements (<element>) and attributes (attribute). Unless otherwise indicated, all data presented here are mandatory fields.

<table>
<thead>
<tr>
<th>Element/Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;AuthnRequest&gt;</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>Specifies the SAML-Version being used and MUST have the value '2.0'.</td>
</tr>
<tr>
<td>ID</td>
<td>Uniquely identifies the request and MUST fulfill the same requirements as the session ID described in Section 3.3.1: SessionType.</td>
</tr>
<tr>
<td>IssueInstant</td>
<td>MUST contain the time of issue of the request in compliance to [SAML]</td>
</tr>
<tr>
<td>Destination</td>
<td>MUST contain the URL of the eID-Service which SHALL perform the Online-Authentication for the eService.</td>
</tr>
<tr>
<td>ProtocolBinding</td>
<td>Since the binding being used is already defined in the technical guideline this attribute is optional, but MUST have the value 'urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect' if present.</td>
</tr>
<tr>
<td>AssertionConsumerServiceURL</td>
<td>Contains the exact URL of the eService where the Response SHALL be sent. The attribute is optional IF the eService has a default URL configured at the eID-Service. The eService MAY use this URL to differ from the default URL.</td>
</tr>
<tr>
<td>ProviderName</td>
<td>This element MAY be used by the eService to submit a textual identifier of the eService to the eID-Service.</td>
</tr>
<tr>
<td>&lt;Issuer&gt;</td>
<td>MUST contain the eServices URL and SHALL be used by the eID-Service to identify the eService.</td>
</tr>
<tr>
<td>&lt;Extensions&gt;</td>
<td>This element MUST be present to encapsulate all extensions defined or referenced in this specific profile.</td>
</tr>
<tr>
<td>&lt;EncryptedAuthnRequestExtension&gt;</td>
<td>This element MUST be present to encapsulate the extension containing the encrypted AuthnRequestExtension. The element is defined in Section 4.5.2: EncryptedAuthnRequestExtension.</td>
</tr>
<tr>
<td>&lt;EncryptedData&gt;</td>
<td>Using XML-Encryption as referenced in [SAML] this element MUST contain the encrypted AuthnRequestExtension (see Section 4.7.2: AuthnRequestExtension).</td>
</tr>
</tbody>
</table>

Table 12: Elements and Attributes of the AuthnRequest
4.7.2 AuthnRequestExtension

The AuthnRequestExtension is of the data type AuthnRequestExtensionType and MUST contain the following XML elements (<element>) and attributes (attribute). Unless otherwise indicated, all data presented here are mandatory fields.

<table>
<thead>
<tr>
<th>Element/Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;AuthnRequestExtension&gt;</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>Specifies the version of the extension structure being used and SHOULD have the value ‘2’.</td>
</tr>
<tr>
<td>&lt;RequestedAttributes&gt;</td>
<td>This element MUST encapsulate at least one Attribute element in compliance with [SAML] and as defined in Section 4.4.4: RequestedAttributesType.</td>
</tr>
<tr>
<td>&lt;Attribute&gt;</td>
<td>For each attribute requested by the eService this element MUST be present and used in compliance to [SAML].</td>
</tr>
<tr>
<td>Name</td>
<td>Identifier of the requested attribute in compliance with [SAML]. Usage of the attribute names defined under Section 4.3.1: Request Attributes MUST be supported. Additional elements MAY be allowed if specified by the Service Provider and eID-Service.</td>
</tr>
<tr>
<td>RequiredAttribute</td>
<td>Contains optional information about whether the attribute is required for the eService's use case or not. If no value is presented the default value 'true' SHALL be assumed by the SAML Processor.</td>
</tr>
<tr>
<td>&lt;AttributeValue&gt;</td>
<td>If a requested attribute requires input the input SHALL be contained in this element using the data type as specified in Section 4.3.1: Request Attributes.</td>
</tr>
</tbody>
</table>

Table 13: Elements and Attributes of the AuthnRequestExtension

4.7.3 Response

The Response must contain the following XML elements (<element>) and attributes (attribute). Unless otherwise indicated, all data presented here are mandatory fields.

<table>
<thead>
<tr>
<th>Element/Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Response&gt;</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>Specifies the SAML-Version being used and MUST have the value '2.0'.</td>
</tr>
<tr>
<td>ID</td>
<td>Uniquely identifies the response and MUST fulfill the same requirements as the session ID described in Section 3.3.1: SessionType.</td>
</tr>
<tr>
<td>InResponseTo</td>
<td>MUST contain the ID submitted in the initial request to which the eID-Service is responding.</td>
</tr>
<tr>
<td>IssueInstant</td>
<td>MUST contain the time of issue of the response in compliance to [SAML].</td>
</tr>
<tr>
<td>Element/Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Destination</strong></td>
<td>URL of the eService to which the response is being sent. Especially the value of the AssertionConsumerServiceURL attribute from the response SHOULD be used here. Alternatively a default URL MAY be used.</td>
</tr>
<tr>
<td>&lt;Issuer&gt;</td>
<td>MUST contain the eID-Services URL and SHALL be used by the eService to identify the eID-Service.</td>
</tr>
<tr>
<td>&lt;Status&gt;</td>
<td>MUST contain general information about the status of the Online-Authentication.</td>
</tr>
<tr>
<td>&lt;StatusCode&gt;</td>
<td>This element MUST contain the primary StatusCode of the response. If the request was processed successfully the value 'urn:oasis:names:tc:SAML:2.0:status:Success' MUST be used. All other values identify a case of an error during processing. Under this StatusCode MAY be a Subordinate StatusCode in compliance with [SAML]. This element is optional and MAY only be used in case of an error. The values transmitted in case of an error are application specific and SHOULD enable the eService to generate a textual message to the user.</td>
</tr>
<tr>
<td>&lt;StatusMessage&gt;</td>
<td>This element is optional and MAY be used only in case of an error. It contains a textual description for the eService and is thus unsuitable for display to the user.</td>
</tr>
<tr>
<td>&lt;EncryptedAssertion&gt;</td>
<td>This element contains the encrypted assertion as defined in [SAML].</td>
</tr>
<tr>
<td>&lt;EncryptedData&gt;</td>
<td>Using XML-Encryption as referenced in [SAML] this element MUST contain the encrypted Assertion (see Section 4.7.4: Assertion).</td>
</tr>
</tbody>
</table>

Table 14: Elements and Attributes of the Response

4.7.4 Assertion

The encrypted Assertion inside of the response must contain the following XML elements (<element>) and attributes (attribute). Unless otherwise indicated, all data presented here are mandatory fields.

<table>
<thead>
<tr>
<th>Element/Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Assertion&gt;</td>
<td></td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>Specifies the SAML-Version being used and MUST have the value '2.0'.</td>
</tr>
<tr>
<td><strong>ID</strong></td>
<td>Uniquely identifies the assertion and MUST fulfill the same requirements as the session ID described in Section 3.3.1: SessionType.</td>
</tr>
<tr>
<td>Element/Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IssueInstant</td>
<td>MUST contain the time of issue of the assertion in compliance to [SAML].</td>
</tr>
<tr>
<td>&lt;Issuer&gt;</td>
<td>MUST contain the eID-Services URL and SHALL be used by the eService to identify the eID-Service.</td>
</tr>
<tr>
<td>&lt;Subject&gt;</td>
<td>This element MUST be present so the eService is able to verify the assertion’s validity.</td>
</tr>
<tr>
<td>&lt;NameID&gt;</td>
<td>This element contains a random ID in compliance with [SAML].</td>
</tr>
<tr>
<td>Format</td>
<td>The format 'urn:oasis:names:tc:SAML:2.0:nameid-format:transient' SHALL be used in this context.</td>
</tr>
<tr>
<td>&lt;SubjectConfirmation&gt;</td>
<td>Encapsulates the information about the subject of the assertion.</td>
</tr>
<tr>
<td>Method</td>
<td>The method 'urn:oasis:names:tc:SAML:2.0:cm:bearer' SHALL be used for confirmation in this context.</td>
</tr>
<tr>
<td>&lt;SubjectConfirmationData&gt;</td>
<td>MUST contain information to confirm the subject of the assertion.</td>
</tr>
<tr>
<td>Address</td>
<td>MUST contain the IP address of the user client the SAML processor is communicating with. To this connection special security requirements as described in Section 4.9.3: Channel Binding MUST be fulfilled.</td>
</tr>
<tr>
<td>InResponseTo</td>
<td>MUST contain the ID submitted in the initial request to which the eID-Service is responding.</td>
</tr>
<tr>
<td>NotOnOrAfter</td>
<td>Since the Online-Authentication is only valid for a specific moment the time the assertion becomes invalid MUST NOT be longer then 5 minutes after the creation of the assertion.</td>
</tr>
<tr>
<td>Recipient</td>
<td>URL of the eService to which the response is being sent. Especially the value of the Assertion ConsumerServiceURL attribute from the response SHOULD be used here. Alternatively a default URL or a specific endpoint address at the eService MAY be used.</td>
</tr>
<tr>
<td>&lt;Conditions&gt;</td>
<td>This element describes the conditions that apply for the usage of the assertion.</td>
</tr>
<tr>
<td>&lt;AudienceRestriction&gt;</td>
<td>Describes the audience to which the assertion is addressed.</td>
</tr>
<tr>
<td>&lt;Audience&gt;</td>
<td>URL that identifies the eService to whom the assertion SHALL be sent. Especially the value of the Issuer element from the response SHOULD be used here.</td>
</tr>
<tr>
<td>&lt;OneTimeUse&gt;</td>
<td>This element MUST be present, because Online-Authentication with the eID-Function is valid only</td>
</tr>
<tr>
<td>Element/Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>&lt;AuthnStatement&gt;</td>
<td>Describes basic conditions under which the assertion was created by the eID-Service.</td>
</tr>
<tr>
<td>AuthnInstant</td>
<td>MUST contain the time the Online-Authentication took place in compliance to [SAML].</td>
</tr>
<tr>
<td>&lt;AuthnContext&gt;</td>
<td>Encapsulates the context in which the Online-Authentication took place.</td>
</tr>
<tr>
<td>&lt;AuthnContextDeclRef&gt;</td>
<td>Since eID-Documents are basically Smartcards with a PKI the fixed value 'urn:oasis:names:tc:SAML:2.0:ac:classes:SmartcardPKI' SHALL be used.</td>
</tr>
<tr>
<td>&lt;AttributeStatement&gt;</td>
<td>The element contains a set of Attribute elements with AttributeValue sub-elements containing the user's personal information. Use of the identifiers defined in Section 4.3.2: Response Attributes is mandatory for the eID function's attributes here too.</td>
</tr>
<tr>
<td>&lt;Attribute&gt;</td>
<td>For each successfully read data field an Attribute element MUST be transmitted.</td>
</tr>
<tr>
<td>Name</td>
<td>The identifiers listed in Section 4.3.2: Response Attributes MUST be used for here. Additional names MAY be allowed if specified by the eService and eID-Service.</td>
</tr>
<tr>
<td>AttributeNotOnChip</td>
<td>Contains optional information if reading of the attribute was not supported by the eID-Document's chip. If no value is present the default value 'false' SHALL be assumed by the eService.</td>
</tr>
<tr>
<td>&lt;AttributeValue&gt;</td>
<td>Contains the Value of the requested data field.</td>
</tr>
<tr>
<td>xsi:type</td>
<td>The types listed in Section 4.3.2: Response Attributes MUST be used here. Additional types for further attributes MAY be allowed if specified by the eService and eID-Service.</td>
</tr>
<tr>
<td>Name</td>
<td>An attribute 'DocumentValidity' MUST be present if the Service Provider's request was generally processed.</td>
</tr>
<tr>
<td>&lt;AttributeValue&gt;</td>
<td>This attribute MUST be of the data type 'eid:DocumentValidityResultType'.</td>
</tr>
<tr>
<td>Version</td>
<td>Specifies the version of the assertion structure being used and SHOULD have the value '1'.</td>
</tr>
<tr>
<td>&lt;ReferenceDate&gt;</td>
<td>This element MUST contain the exact date the verification of the eID-Document was done.</td>
</tr>
<tr>
<td>&lt;Status&gt;</td>
<td>This element MUST contain the result of the</td>
</tr>
</tbody>
</table>
4.8 Error Handling

The SAML Processor of the eID-Service has two options when dealing with invalid SAML Requests. Both options and the conditions under which they are to be chosen are described in the following sections.

4.8.1 General and Security Related Errors

If the SAML Processor is unable to verify the validity of a SAML Request (e.g. signature verification failed, scheme validation failed) the SAML Processor MUST abort the communication with the eID-Client using the HTTP status code **400 Bad Request** according to [TR-03124] Part 1, Section 2.5.4.3: Communication error without refreshURL as described in the following Figure 42: Procedure for general and security related errors (SAML).

If a general or security related (e.g. signature verification failed, scheme validation failed) error appears the following procedure SHALL be implemented by the eID-Server.

1. **Redirect to SAML Processor incl. AuthnRequest**

As described in Section 4.2.1: Initiation Step 4 the eID-Client receives a redirect to the SAML Processor including the AuthnRequest from the eService.

2. **Call of SAML Processor incl. AuthnRequest**
As described in *Section 4.2.1: Initiation Step 5* the eID-Client calls SAML Processor with the `AuthnRequest`.

3. **Answer with HTTP Code 400 Bad Request**
   
The SAML Processor answers the invalid request with the HTTP status code **400 Bad Request**.

4. **Forward Browser to HTTP Code 400 Bad Request**
   
   As described in [TR-03124] *Section 2.5.4.3: Communication error without refreshURL* the eID-Client SHALL convey a HTTP status code **400 Bad Request** including a meaningful human-readable error message.

### 4.8.2 Sending SAML Error Messages

Error messages according to [SAML] *Assertions and Protocols, Section 3.2.2.2: Element <StatusCode>* MAY only be sent from the SAML Processor to the eService using the procedure described in this section IF

1. the SAML Processor was able to verify that the SAML Request was sent by an authorized eService AND

2. the SAML Processor was able to interpret the SAML Request according to this specification.

If both conditions apply the eID-Server may act as shown in the following *Figure 43: Procedure for sending SAML error messages* in all other cases the procedure described in *Section 4.8.1: General and Security Related Errors* SHALL be used.

The procedure for sending SAML error messages in form of status codes includes the following steps.

1. **Redirect to SAML Processor incl. AuthnRequest**
   
   As described in *Section 4.2.1: Initiation Step 4* the eID-Client receives a redirect to the SAML Processor including the `AuthnRequest` from the eService.

2. **Call of SAML Processor incl. AuthnRequest**
As described in Section 4.2.1: Initiation Step 5 the eID-Client calls SAML Processor with the AuthnRequest.

3. Redirect to eService incl. Response with error code

The SAML processor answers the call of the eID-Client with a redirect to the eService including a SAML Response with an error code according to [SAML] Assertions and Protocols, Section 3.2.2.2: Element <StatusCode>.

4. Call of eService incl. Response with error code

The eID-Client calls the eService including the Response from the SAML processor.

5. Answer with HTTP status code 400 Bad Request

The eService answers the call with the HTTP status code 400 Bad Request.

6. Forward Browser to HTTP status code 400 Bad Request

As described in [TR-03124] Section 2.5.4.3: Communication error without refreshURL the eID-Client SHALL convey a HTTP status code 400 Bad Request including a meaningful human-readable error message.

4.9 Security Measures

This section describes the security measures that the communication partners MUST take if they use the SAML Profile described in this technical guideline. These measures result in particular from the trust relationship between the eID-Service and the Service Provider as well as general security requirements to the transport layer.

All security relevant communication channels are shown in Figure 44: Communication channel overview (SAML). The communication channels TLS-1-2a and TLS-1-2b ending at the user's eID-Client are used for the transport of the SAML messages. As specified in [TR-03124] Part 1, Section 4.5: TLS all of these channels SHALL be secured using TLS connections in conformance to [TR-03116-4] Section 2: Vorgaben für SSL/TLS to secure the exchange of the SAML messages. To prevent misuse of these channels on renewal the eService...
and the SAML Processor SHOULD keep these connections alive for the transport of all messages that are part of one Online-Authentication.

The threats cited in [SAML] Security and Privacy Considerations especially Section 6.4: HTTP Redirect/POST Binding and Section 7.1.1: SSO Profiles SHALL also be considered. The necessary elements for the counter-measures to the threats are included in this SAML-Profile and MUST be used by all communication partners. Especially all relevant checks described in [SAML] Security and Privacy Considerations SHALL be performed by the Service Provider and the eID-Service.

The SAML communication itself is represented by the dark blue arrow in Figure 44: Communication channel overview (SAML) connecting the eService and the SAML Processor. The security measures for this communication channel are described in the following sections.

4.9.1 Encryption

During establishment of the trust relationship between the Service Provider and the eID-Service, both parties MUST exchange the necessary keys for encrypting SAML messages on a secure way. Each party MUST create an individual encryption-key pair, which MUST be different from the signature-key pair. Only algorithms and key sizes recommended by [TR-03116-4] Section 4.4: XML Encryption SHALL be used.

The extension containing the requested data groups is encrypted in the SAML request. The element AuthnRequestExtension MUST be encrypted as specified in [XML-Enc] and placed inside the element EncryptedData in the element EncryptedAuthnRequestExtension (see Section 4.5.2: EncryptedAuthnRequestExtension).

The assertion SHALL also be encrypted as specified in [XML-Enc] and MUST be placed inside the element EncryptedData in the element EncryptedAssertion of the response (see Section 4.7.3: Response) as described in [SAML] Assertions and Protocols, Section 6: SAML and XML Encryption Syntax and Processing.

Since SAML messages are highly structured static symmetric keys MUST NOT be used for encryption. The original encryption keys used during negotiation of the trust relationship MUST NOT be transmitted in any message. Suitably encoded transient or temporary keys SHALL be transmitted if the encryption technology, for instance hybrid encryption, requires so. Hybrid encryption methods with random keys SHOULD be used.

4.9.2 Signature

During establishment of the trust relationship between the Service Provider and the eID-Service, both parties MUST exchange the necessary keys for signing SAML messages on a secure way. Each party MUST create an individual signature-key pair, which MUST be different from the encryption-key pair. Only algorithms and key sizes recommended by [TR-03116-4] Section 4.3: XML Signature SHALL be used.

The original signature-keys used during establishment of the trust relationship MUST NOT be transmitted in any message. The respective receivers MUST check the signatures before processing a SAML message.

4.9.2.1 URL-Encoding

A signature SHALL be applied to every SAML message exchanged using this profile. Since this profile uses the HTTP Redirect Binding as specified in [SAML] Bindings, Section 3.4 HTTP Redirect Binding the signature SHALL be applied to the parameters URL-Encoded in the Location field of the redirects. The AuthnRequest and the Response as well as the URI identifying the signature algorithm (SigAlg) MUST be encoded and signed as specified in [SAML] Bindings, Section 3.4.4.1 DEFLATE Encoding. An example with fictional values is shown in the following code example.
Example 9: URL-Encoded SAML Message

Only algorithms and key sizes recommended by [TR-03116-4] Section 4.3: XML Signature SHALL be used. The assertion itself SHOULD not be signed separately as it is embedded in the signed Response.

4.9.3 Channel Binding

The channel binding of the channel TLS-1-2a between the eID-Client and the eService and the channel TLS-1-2b between the eID-Client and the SAML Processor SHALL be secured using the SAML protocol as defined in this profile. The eService SHALL only accept and process valid assertions from the eID-Service that are bound to an existing web-session. The eID-Service SHALL only accept and process valid requests from authorized eServices.

The channel binding of the channel TLS-1-2b between the SAML Processor and the eID-Client and the channel TLS-2 between the eID-Server and the eID-Client represented in Figure 44: Communication channel overview (SAML) MUST fulfill the same requirements as the channel binding of the eID-Interface described in Section 3.5.3: Session Binding with the SAML Processor fulfilling the role of the eService in this scenario.

4.10 Examples

The following code shows an example process flow of an authentication using the SAML Profile specified in this section. The code samples contain carriage returns and indentations for better readability. These usually do not occur in actual messages.

4.10.1 AuthnRequest

This code sample shows an example of an AuthnRequest in compliance with Section 4.7.1: AuthnRequest.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<samlp:AuthnRequest
AssertionConsumerServiceURL="https://www.eservice.bund.de/processResponse"
Destination="https://www.eid-service.bund.de/processRequest"
ID="1234567890abcdef1234567890abcdef"
IssueInstant="2014-01-15T12:00:00Z"
Version="2.0"
xmns:ds="http://www.w3.org/2000/09/xmldsig#"
xmns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
xmns:soap="urn:oasis:names:tc:SAML:2.0:assertion"
xmns:xenc="http://www.w3.org/2001/04/xmlenc#"
xmns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmns:eid="http://bsi.bund.de/eID/">
  <saml2:Issuer>https://www.eservice.bund.de</saml2:Issuer>
  <samlp:Extensions>
    <eid:EncryptedAuthnRequestExtension>
      <xenc:EncryptedData
        Type="http://www.w3.org/2001/04/xmlenc#Element">
        <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
      </xenc:EncryptedData>
    </eid:EncryptedAuthnRequestExtension>
  </samlp:Extensions>
</samlp:AuthnRequest>
```
Example 10: AuthnRequest

4.10.2 AuthnRequestExtension

This code sample shows the `AuthnRequestExtension`, which is encrypted and then embedded in the previously described AuthnRequest, as defined in Section 4.7.2: `AuthnRequestExtension`. The `AuthnRequestExtension` contains the data fields and operations defined in Section 4.3.1: `Request Attributes` that are requested by the eService. In this example all of the data fields and operations except the `ResidencePermitI` are requested and marked as mandatory.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<eid:AuthnRequestExtension Version="2"
xmlns:eid="http://bsi.bund.de/eID/
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <eid:RequestedAttributes>
    <saml2:Attribute Name="DocumentType" eid:RequiredAttribute="true" />
    <saml2:Attribute Name="IssuingState" eid:RequiredAttribute="true" />
    <saml2:Attribute Name="DateOfExpiry" eid:RequiredAttribute="true" />
    <saml2:Attribute Name="GivenNames" eid:RequiredAttribute="true" />
    <saml2:Attribute Name="FamilyNames" eid:RequiredAttribute="true" />
    <saml2:Attribute Name="ArtisticName" eid:RequiredAttribute="false" />
    <saml2:Attribute Name="AcademicTitle" eid:RequiredAttribute="false" />
    <saml2:Attribute Name="DateOfBirth" eid:RequiredAttribute="true" />
    <saml2:Attribute Name="PlaceOfBirth" eid:RequiredAttribute="true" />
  </eid:RequestedAttributes>
</eid:AuthnRequestExtension>
</samlp:Extensions>
</samlp:AuthnRequest>
```
Example 11: AuthnRequestExtension

4.10.3 Response

This code sample shows an example of a Response in compliance with Section 4.7.3: Response.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<samlp:Response
    Destination="https://www.eservice.bund.de/processResponse"
    ID="0987654321fedcba0987654321fedcfa"
    InResponseTo="1234567890abcdef1234567890abcdef"
    IssueInstant="2014-01-15T12:00:00"
    Version="2.0"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <saml2:Issuer>https://www.eid-service.bund.de</saml2:Issuer>
    <samlp:Status>
        <samlp:StatusCode
            Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
    </samlp:Status>
    <saml2:EncryptedAssertion>
        <xenc:EncryptedData
            Type="http://www.w3.org/2001/04/xmlenc#Element">
            <xenc:EncryptionMethod
                Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
            <ds:KeyInfo>
                <xenc:EncryptedKey
                    Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
                <xenc:CipherData>

Federal Office for Information Security
Example 12: Response

4.10.4 Assertion

This code sample shows the Assertion, which is embedded encrypted in the previously described Response. It contains the successfully completed authentication's return values as specified in Section 4.3.2: Response Attributes. The assertion was constructed according to requirements from Section 4.7.4: Assertion.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<saml2:Assertion
ID="0987654321fedcbaabcdef123467890"
IssueInstant="2014-01-15T12:00:00Z"
Version="2.0"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:eid="http://bsi.bund.de/eID/"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <saml2:Issuer>https://www.eid-service.bund.de</saml2:Issuer>
  <saml2:Subject>
    <saml2:NameID
      Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
      6b69710c2d804b48356209c9788a661f
    </saml2:NameID>
    <saml2:SubjectConfirmation
      Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
      <saml2:SubjectConfirmationData
        Address="127.0.0.1"
        InResponseTo="1234567890abcdef1234567890abcdef"
        NotOnOrAfter="2014-01-15T12:05:00Z"
        Recipient="https://www.eservice.bund.de/processResponse" />
    </saml2:SubjectConfirmation>
  </saml2:Subject>
</saml2:Assertion>
```
<saml2:Conditions>
  <saml2:AudienceRestriction>
    <saml2:Audience>
      https://www.eservice.bund.de
    </saml2:Audience>
  </saml2:AudienceRestriction>
  <saml2:OneTimeUse />
</saml2:Conditions>
<saml2:AuthnStatement AuthnInstant="2014-01-15T12:00:00Z">
  <saml2:AuthnContext>
    <saml2:AuthnContextDeclRef>
      urn:oasis:names:tc:SAML:2.0:ac:classes:SmartcardPKI
    </saml2:AuthnContextDeclRef>
  </saml2:AuthnContext>
</saml2:AuthnStatement>
<saml2:AttributeStatement>
  <saml2:Attribute Name="DocumentValidity">
    <saml2:AttributeValue xsi:type="eid:DocumentValidityResultType">
      <eid:ReferenceDate>2014-01-15</eid:ReferenceDate>
      <eid:Status>VALID</eid:Status>
    </saml2:AttributeValue>
  </saml2:Attribute>
  <saml2:Attribute Name="DocumentType">
    <saml2:AttributeValue xsi:type="xs:string">
      ID
    </saml2:AttributeValue>
  </saml2:Attribute>
  <saml2:Attribute Name="IssuingState">
    <saml2:AttributeValue xsi:type="xs:string">
      D
    </saml2:AttributeValue>
  </saml2:Attribute>
  <saml2:Attribute Name="DateOfExpiry">
    <saml2:AttributeValue xsi:type="xs:date">
      2022-08-01
    </saml2:AttributeValue>
  </saml2:Attribute>
  <saml2:Attribute Name="GivenNames">
    <saml2:AttributeValue xsi:type="xs:string">
      ERIKA
    </saml2:AttributeValue>
  </saml2:Attribute>
  <saml2:Attribute Name="FamilyNames">
    <saml2:AttributeValue xsi:type="xs:string">
      MUSTERMANN
    </saml2:AttributeValue>
  </saml2:Attribute>
  <saml2:Attribute Name="AcademicTitle" />
  <saml2:Attribute Name="DateOfBirth">
    <saml2:AttributeValue xsi:type="eid:GeneralDateType">
      <eid:DateString>19640812</eid:DateString>
    </saml2:AttributeValue>
  </saml2:Attribute>
  <saml2:Attribute Name="PlaceOfBirth">

Federal Office for Information Security

70
<saml2:AttributeValue xsi:type="eid:GeneralPlaceType">
  <eid:FreetextPlace>
    BERLIN
  </eid:FreetextPlace>
</saml2:AttributeValue>

<saml2:Attribute Name="Nationality">
  <saml2:AttributeValue xsi:type="xs:string">
    D
  </saml2:AttributeValue>
</saml2:Attribute>

<saml2:Attribute Name="BirthName" AttributeNotOnChip="true">
  <saml2:AttributeValue xsi:type="xs:string"/>
</saml2:Attribute>

<saml2:Attribute Name="PlaceOfResidence">
  <saml2:AttributeValue xsi:type="eid:GeneralPlaceType">
    <eid:StructuredPlace>
      <eid:Street>HEIDESTRASSE 17</eid:Street>
      <eid:City>KÖLN</eid:City>
      <eid:Country>D</eid:Country>
    </eid:StructuredPlace>
  </saml2:AttributeValue>
</saml2:Attribute>

<saml2:Attribute Name="RestrictedID">
  <saml2:AttributeValue xsi:type="eid:RestrictedIDType">
    <eid:ID>01A4FB509CEBC6595151A4FB5F9C75C6FE01A4FB59CB655A4FB5F9C75C6FEE</eid:ID>
    <eid:ID2>5C6FE01A4FB59CB655A4FB5F9C75C6FEE01A4FB509CEBC6595151A4FB5F9C7</eid:ID2>
  </saml2:AttributeValue>
</saml2:Attribute>

<saml2:Attribute Name="AgeVerification">
  <saml2:AttributeValue xsi:type="eid:AgeVerificationResultType">
    <eid:Request>
      <eid:Age>18</eid:Age>
    </eid:Request>
    <eid:Result>
      <eid:FulfilsRequest>true</eid:FulfilsRequest>
    </eid:Result>
  </saml2:AttributeValue>
</saml2:Attribute>

<saml2:Attribute Name="PlaceVerification">
  <saml2:AttributeValue xsi:type="eid:PlaceVerificationResultType">
    <eid:Request>
      <eid:CommunityID>027605</eid:CommunityID>
    </eid:Request>
    <eid:Result>
      <eid:FulfilsRequest/>
    </eid:Result>
  </saml2:AttributeValue>
</saml2:Attribute>
Example 13: Assertion

```xml
true
  </eid:FulfilsRequest>
  </eid:Result>
</saml2:AttributeValue>
</saml2:Attribute>
</saml2:AttributeStatement>
</saml2:Assertion>
```
5 eIDAS-Extension and eSAML Profile

In order to enhance trust and interoperability of electronic transactions in the EU, the eIDAS regulation sets the regulatory framework for the mutual recognition of electronic identification means.

According to eIDAS, European Member States may notify their eID systems to the European Commission. While the notification is optional, recognition of notified schemes is mandatory under the terms and conditions of the regulation.

The choice of the identification means depends on the level of assurance (LoA) required by the service of the relying party. For that purpose, eIDAS distinguishes between three levels of assurance (LoA), i.e. ‘low’, ‘substantial’ and ‘high’.

Interoperability is achieved via the eIDAS Interoperability Framework [eIDAS-Interop] consisting of so-called eIDAS Nodes. The framework distinguishes between eIDAS Connectors (on the side of the receiving Member State) and eIDAS Services (on the side of the sending Member State). The eIDAS Nodes forming the eIDAS Network ensure the confidentiality, the authenticity and integrity of the transmitted personal data as well as the secure identification of the endpoints.

In this section, the additional measures are described, which are needed to enable the eID-Server to forward the user’s authentication session over the eIDAS Connector towards the eIDAS Service. The main difference is that the authentication happening between eID-Client and eID-Server is replaced by the authentication conducted within the eIDAS Network and the response is provided back to the relying party.

For an eIDAS-compliant cross-border authentication the receiving Member State must request the attributes of the general Minimum Data Set (see [eIDAS-Attributes]) and the sending Member State must answer with its specific Minimum Data Set (usually a subset of the general Minimum Data Set). When requesting an eIDAS authentication, the Service Provider therefore should request all attributes from the general Minimum Data Set and has to process the responded attributes from the specific Minimum Data Set. If a legally eIDAS compliant authentication is not needed, the Service Provider MAY also only request and process a subset of the Minimum Data Set.

5.1 Infrastructure Adaption to eIDAS context

The goal is, to direct the user session into the eIDAS Network and provide the result back to the relying party.

This is achieved by adapting the national communication with the eID-Server as specified in this document and using the eIDAS Connector (interface) as specified in [eIDAS-Interop]. As the eIDAS-process relies strongly on SAML [eIDAS-SAML] and therefore the advantages of the eID-Interface (SOAP) described in Chapter 3: eID-Interface (SOAP) are mitigated, support for SOAP in the eIDAS-context is dropped and only SAML is used.

The SAML-Profile described in Chapter 4 SAML-Profile is adapted and supplemented in order to comply with the different circumstances. The extended SAML-Profile (eSAML) as described in the following sections SHALL be implemented if the eID-Server is to be used as eIDAS Node.

In contrast to Chapter 4: SAML-Profile, HTTP-Post Binding as specified in [SAML] Bindings SHALL be used in the context of eIDAS. Deviating from the roles as defined in Table 11: Attribute and Elements of the Data type AuthnRequestExtensionType, the user’s browser acts as the user agent regarding the SAML messages towards the eID-Server and the eIDAS Network. By using encrypted SAML Assertions eService and eID-Server as well as eID-Server and eIDAS-Service respectively are enabled to perform secured communication with each other (depicted in Figure 45: eID-Server infrastructure adapted to eIDAS context with circular arrows).
5.2 eIDAS Message Flow

The message flow on the national part of an eIDAS authentication is adapted from the SAML profile described in Section 4.2: General Message Flow.

In the following the eIDAS Connector is depicted as a component on its own, but MAY usually be integrated into the eID-Server, as it also needs contact to the background infrastructure for retrieving e.g. Metadata Signer certificates.

In analogy to Section 4: SAML-Profile the descriptions in Section 5.2.1 Initiation and Section 5.2.3 Completion assume, that the eSAML messages used for communication with the eService are sent over a dedicated SAML Processor. The SAML Processor MAY be the same component as used in the national protocol and MAY also be directly attached to the eID-Server. Alternatively the eSAML-Profile MAY also be implemented directly by the eIDAS Connector.

The description in Section 5.2.2 eIDAS authentication process assumes, that the eIDAS-SAML messages used for communication with the eIDAS Service are sent directly from the eIDAS Connector (component), but MAY also be sent over a dedicated SAML Processor (i.e. the same component as above).

5.2.1 Initiation

The following Figure 46: General message flow during initiation (eIDAS) shows the initiation of an eIDAS authentication process. Instead of providing a local link for the eID-Client the eService provides a POST form to the browser in order to forward the SAML request to the eID-Server.
1. **Access eService**

The user accesses the eService via the browser and chooses to use eIDAS authentication. An option for selecting the Sending Member State MAY be provided to the user.

2. **Provide POST form incl. AuthnRequest**

The resource provided by the eService MUST include a XHTML form with POST. That POST MUST be targeted to the SAML Processor and MUST contain the SAML Authentication Request (see [SAML] Bindings).

3. **Submit POST to SAML Processor incl. AuthnRequest**

The included AuthnRequest (compare *Section 4.7.1: AuthnRequest*) is submitted to the SAML Processor via POST.

4. **Request eIDAS session**

The SAML Processor connects to the eIDAS Connector to request an authentication session. The attributes from the AuthnRequest SHALL be mapped to the eIDAS request according to *Section 5.5: Attribute Mapping*. If not already provided within the AuthnRequest, the eIDAS Connector MUST present a possibility to select the sending member state.

### 5.2.2 eIDAS authentication process

The following *Figure 47: General message flow during interaction (eIDAS)* shows the authentication process within the eIDAS Network. It is shown here only for reference. The technical details are provided in [eIDAS-Interop], [eIDAS-SAML] and [eIDAS-Attributes]. Within the eIDAS Network mostly HTTP POST Binding is used, but eIDAS Services must also support HTTP Redirect Binding for eIDAS-Requests [eIDAS-Interop].
The eIDAS Connector SHOULD use HTTP Redirect Binding for the `eIDAS-AuthnRequest` if the request is short enough (depending on e.g. the browser limiting URL length).²

Regardless of the eID attributes originally requested in the `eSAML AuthnRequest`, the `eIDAS-AuthnRequest` MUST request all attributes of the general Minimum Data Set as defined in [eIDAS-SAML].

1. Redirect/ Provide POST to eIDAS Service incl. eIDAS-AuthnRequest
2. Call of/ Submit POST to eIDAS Service incl. eIDAS-AuthnRequest
3. Authentication by the chosen means
4. Provide POST to SAML-Processor incl. eIDAS-Response
5. Submit POST to SAML-Processor incl. eIDAS-Response

5.2.3 Completion

The following Figure 48: General message flow during completion (eIDAS) shows the completion of the authentication process. The result of the eIDAS authentication is provided back to the eService.

If the LoA supplied in the eIDAS response is lower than the LoA of the request, the eID-Server MUST abort the authentication and provide an error to the eService.

² This recommendation is aimed at reducing latency (redirect processing is usually faster than POST-processing) and enhancing usability in environments where java-script is not available, e.g. corporate networks.
1. **Respond with eIDAS-Result**

   The eIDAS Connector provides the result of the authentication to the SAML Processor. The attributes from the eIDAS-Response SHALL be mapped back to the Response according to *Section 5.5: Attribute Mapping*.

2. **Provide POST form incl. Response**

   The SAML Processor MUST provide the SAML response in an XHTML form with POST. That POST MUST be targeted to the eService and MUST contain the SAML Authentication Request (see [SAML] **Bindings**).

3. **Submit POST to eService incl. Response**

   The included *Response* (compare *Section 4.7.3: Response*) is submitted to the eService via POST.

4. **Proceed**

   The Browser proceeds with the websession at the eService.

### 5.3 Additional Attributes in eSAML

The SAML attributes defined in *Section 4.3: Attributes* SHALL be used for communication. They are supplemented by attributes providing the additionally needed information for using the eIDAS Network.

The tables below also contain information about the data types that MUST be used to transmit values upon requests and responses.
5.3.1 Request Attributes

The following table contains additional attributes (supplementing Table 8: SAML Request Attributes) the Service Provider MAY use, when sending a request to the eID-Service. In order to request an authentication within the eIDAS Network rather than a national eID usage, the attribute UseEidas MUST be used and set to “true”. The attribute EidasExtension is CONDITIONAL. It MUST be used, if UseEidas is set to “true” and the contained elements (see Section 5.4.1: EidasExtensionType) are used to transmit additional information. Otherwise, it SHOULD be omitted. The attribute Sex MAY be used in a request.

As noted above the eService’s request should include all attributes from the general Minimum Data Set, if an eIDAS compliant authentication is needed. Regardless of the attributes requested the eID-Server SHALL however always request the full Minimum Data Set from the eIDAS Service.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Data type used for value in request</th>
</tr>
</thead>
<tbody>
<tr>
<td>UseEidas</td>
<td>xs:boolean</td>
</tr>
<tr>
<td>EidasExtension</td>
<td>eid:EidasExtensionType</td>
</tr>
<tr>
<td>Sex</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 16: Supplementing SAML Request attributes for eIDAS usage

5.3.2 Response Attributes

The following table contains additional attributes (supplementing Table 9: SAML Response Attributes) the eID-Service SHALL transmit to the Service Provider. Both attributes are MANDATORY if the attribute UseEidas was set to “true” in the request. In this case the attribute UseEidas MUST also be set to “true” in the response and both contained elements in the attribute EidasExtension (see Section 5.4.1: EidasExtensionType) MUST be set according to the result of the eIDAS authentication.

The attribute Sex is CONDITIONAL and SHALL be used, if the gender is provided in the Minimum Data Set of the Sending Member State. Additionally, the attribute EidasExtension MAY also include the original eIDAS-Attributes (names and values) from the eIDAS-Response.

If the eService only requested a subset of the Minimum Data Set, the eID-Server’s response SHOULD only provide the attributes requested. Further Minimum Data Set attributes from the eIDAS response MAY then be discarded.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Data type used for value in response</th>
</tr>
</thead>
<tbody>
<tr>
<td>UseEidas</td>
<td>xs:boolean</td>
</tr>
<tr>
<td>EidasExtension</td>
<td>eid:EidasExtensionType</td>
</tr>
<tr>
<td>Sex</td>
<td>eid:ICAOSex</td>
</tr>
</tbody>
</table>

Table 17: Supplementing SAML response attributes for eIDAS usage

5.4 Additional Data Types in eSAML

Additional data types when using the eSAML Profile are also specified in the XML-Schema TR-03130eID-Server.xsd and SHALL be used as described in the following section.
5.4.1 EidasExtensionType

This data type SHALL be used for transmitting the additional information needed within the eIDAS Network in an assertion. It contains the elements LevelOfAssurance and SendingMemberState which SHALL be used in AuthnRequest and Response as described in Table 18: Elements of the Data Type EidasExtensionType and their usage in Request and Response. When used in a Response it MAY also contain the original eIDAS-attributes from the eIDAS-Response for reference as defined in [eIDAS-Attributes].

<table>
<thead>
<tr>
<th>Element</th>
<th>Usage in AuthnRequest</th>
<th>Usage in Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>LevelOfAssurance</td>
<td>Element is OPTIONAL and contains the minimal Level of Assurance (LoA) desired by the Service Provider. If not provided the eID-Service MUST assume the highest LoA.</td>
<td>Element is MANDATORY and contains the LoA of the actually used eID-system, as reported by the corresponding eIDAS Service.</td>
</tr>
<tr>
<td>SendingMemberState</td>
<td>Element is OPTIONAL and contains the (preselected) Member State, whose eIDAS Service should be used for authentication. If not provided, a possibility to select the sending MS has to be presented by the eIDAS Connector.</td>
<td>Element is MANDATORY and contains the sending Member State as reported by the eIDAS Service.</td>
</tr>
</tbody>
</table>

Table 18: Elements of the Data Type EidasExtensionType and their usage in Request and Response

5.4.2 LevelOfAssuranceType

The data type is used for transmitting the Level of Assurance in the element LevelOfAssurance and is a string restricted to the following allowed values as described in [eIDAS-SAML] representing the corresponding LoA.

- http://eidas.europa.eu/LoA/low
- http://eidas.europa.eu/LoA/substantial
- http://eidas.europa.eu/LoA/high
5.4.3 SendingMemberStateType

This data type is used for transmitting the Sending Member State in the element `SendingMemberState` and is a string restricted to a two-digit country code (ISO 3166-1 alpha-2) as used for eIDAS and described in [eIDAS-Attributes].

5.4.4 ICAOSex

This data type is used for transmitting the sex in the element `Sex` and is a string restricted to the following values (see [TR-03110] Part 4 Section 2.2.3: Data Groups and Generic Attributes):

- M
- F
- " " (space, without the quotation marks)

5.5 Attribute Mapping

The SAML attribute profile used in the eIDAS Network is specified in [eIDAS-Attributes] and MUST be implemented by the eIDAS Connector. Thus, the eIDAS attributes MUST be mapped to the eID attributes used in the eSAML Profile (see Section 4.3: Attributes and Section 5.3: Additional Attributes in eSAML) by the eID-Server.

5.5.1 eIDAS Minimum Data Set

The counterpart of every Minimum Data Set attribute provided in the eIDAS-Response MUST be included in the eSAML Response, IF it was included in the eSAML AuthnRequest. The mapping SHALL occur as follows:

<table>
<thead>
<tr>
<th>eID Attribute</th>
<th>eIDAS Attribute</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes mandatory according to [eIDAS IF]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GivenNames</td>
<td>FirstName</td>
<td>Latin string as received from eIDAS Network (see Section 5.5.3 Non-Latin Characters)</td>
</tr>
<tr>
<td>FamilyNames</td>
<td>FamilyName</td>
<td>Latin string as received from eIDAS Network (see Section 5.5.3 Non-Latin Characters)</td>
</tr>
<tr>
<td>DateOfBirth</td>
<td>DateOfBirth</td>
<td>eIDAS format YYYY-MM-DD SHALL be mapped to eID format YYYYMMDD.</td>
</tr>
<tr>
<td>RestrictedID</td>
<td>PersonIdentifier</td>
<td>See Section 5.5.4 RestrictedID</td>
</tr>
<tr>
<td>Attributes optional according to [eIDAS IF]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlaceOfBirth</td>
<td>PlaceOfBirth</td>
<td>String as received from eIDAS Network</td>
</tr>
<tr>
<td>BirthName</td>
<td>BirthName</td>
<td>See Section 5.5.5 BirthName</td>
</tr>
<tr>
<td>PlaceOfResidence</td>
<td>CurrentAddress</td>
<td>See Section 5.5.6 PlaceOfResidence</td>
</tr>
<tr>
<td>Sex</td>
<td>Gender</td>
<td>eIDAS (“Male”, “Female”, “Unspecified”) SHALL be mapped to eID (“M”, “F”, ” ”) respectively</td>
</tr>
</tbody>
</table>

Table 19: Attribute Mapping of the Minimum Data Set

3 Please note, that the term “mandatory” does not have a technical meaning here
5.5.2 Further Attributes

Attributes, that were included in the eSAML AuthnRequest but can not be mapped using Table 19: Attribute Mapping of the Minimum Data Set SHALL be handled as follows:

<table>
<thead>
<tr>
<th>eID Attribute</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocumentValidity</td>
<td>Status MUST be set to valid, when eIDAS authentication was successful</td>
</tr>
<tr>
<td>DocumentType</td>
<td>Unsupported, AttributeNotOnChip SHALL be set to true</td>
</tr>
<tr>
<td>IssuingState</td>
<td>Unsupported, AttributeNotOnChip SHALL be set to true</td>
</tr>
<tr>
<td>DateOfExpiry</td>
<td>Unsupported, AttributeNotOnChip SHALL be set to true</td>
</tr>
<tr>
<td>ArtisticName</td>
<td>Unsupported, AttributeNotOnChip SHALL be set to true</td>
</tr>
<tr>
<td>AcademicTitle</td>
<td>Unsupported, AttributeNotOnChip SHALL be set to true</td>
</tr>
<tr>
<td>Nationality</td>
<td>Unsupported, AttributeNotOnChip SHALL be set to true</td>
</tr>
<tr>
<td>ResidencePermitI</td>
<td>Unsupported, AttributeNotOnChip SHALL be set to true</td>
</tr>
<tr>
<td>AgeVerification</td>
<td>Age verification SHOULD be mimicked by the eID-Server by checking the</td>
</tr>
<tr>
<td></td>
<td>supposed age against the date of birth and providing a boolean answer</td>
</tr>
<tr>
<td></td>
<td>in FulfilsRequest (see Section 4.4.2 AgeVerificationResultType)</td>
</tr>
<tr>
<td>PlaceVerification</td>
<td>Unsupported, AttributeNotOnChip SHALL be set to true</td>
</tr>
</tbody>
</table>

Table 20: Handling of Attributes not in the eIDAS Minimum Data Set

Member States may include further attributes in their eIDAS response. Those attributes MAY be mapped to the according counterparts of the above attributes.

5.5.3 Non-Latin Characters

The eIDAS attributes GivenNames, FamilyNames and BirthName may contain Non-Latin script variants [eIDAS-Attributes]. In this case only the latin transliteration SHALL be mapped to the according eID attributes. If only the Non-Latin script variant is provided by the eIDAS Service the transliteration SHALL be performed by the eID-Server.

As stated in Section 5.3.2 Response Attributes the original eIDAS attributes including the Non-Latin variant MAY additionally be included within the attribute EidasExtension.

5.5.4 RestrictedID

The eIDAS attribute PersonIdentifier is encoded as UTF-8 string and SHALL be mapped to the eID attribute RestrictedID, which is encoded in hexBinary.

5.5.5 BirthName

The eIDAS attribute BirthName does also contain the first name at birth and is filled, even if the family name has not changed since birth [eIDAS-Attributes].

In order to mostly match the values expected in german eID scheme, the first name(s) or first and last names SHALL be removed from BirthName, if they match the current name(s). If the first name(s) in BirthName does not match the current name, BirthName SHALL be mapped unaltered, even if the last name at birth matches the current FamilyName. For examples see Table 21: Examples for handling of the name mapping.
In eIDAS the attribute `CurrentAddress` carries the street address in a base64 encoded XML sequence of `xs:string` elements following the Core ISA Vocabulary [eIDAS-Attributes]. These elements SHALL be mapped to elements of the data type `StructuredPlace` in the eID attribute `PlaceOfResidence`. The following table provides a guidance.

Depending on the exact usage of the elements by the Sending Memberstates the eID-Server MAY (slightly) deviate from the proposed mapping, in order to provide a better matching.

<table>
<thead>
<tr>
<th>eID address element</th>
<th>eIDAS address element(s)</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Street</td>
<td>Thoroughfare LocatorDesignator, Thoroughfare, LocatorName, PoBox</td>
<td>Choice depending on the provided elements. Order SHOULD be reversed for some countries.</td>
</tr>
<tr>
<td>City</td>
<td>PostName, CvaddressArea</td>
<td>CvaddressArea MAY be missing</td>
</tr>
<tr>
<td>State</td>
<td>AdminUnitSecondline</td>
<td>MAY be missing</td>
</tr>
<tr>
<td>Country</td>
<td>AdminUnitFirstline</td>
<td>MUST be mapped to the type ICAOCountry, if not provided as such</td>
</tr>
<tr>
<td>ZipCode</td>
<td>PostCode</td>
<td>MAY be missing</td>
</tr>
</tbody>
</table>

Table 22: Mapping of address elements

If the eIDAS attribute `CurrentAddress` includes the element `FullAddress` (providing the full address as formatted or unformatted string) its content MAY be mapped to the data type `FreetextPlace` in the eID attribute `PlaceOfResidence`.

If the mapping described in Table 22: Mapping of address elements can not be performed because of critical elements missing or other errors (e.g. `Country` could not be mapped), the data type `FreetextPlace` in the eID attribute `PlaceOfResidence` MUST be used. The content is mapped CONDITIONALLY either from the element `FullAddress`, IF present, as described above or SHALL consist of a concatenation of the provided other address elements in the following layout. Missing values SHALL be omitted:

```plaintext
Thoroughfare LocatorDesignator, LocatorName, PoBox
Postcode Postname, CvaddressArea
AdminUnitSecondline
AdminUnitFirstline
```
5.6 Error Handling

Additionally to the Error Handling described in Section 4.8: Error Handling errors occurring during the eIDAS Authentication SHALL be reported to the eService.
References


[DSS] OASIS: Digital Signature Services (DSS)


[eIDAS-Attributes] eIDAS: SAML Attribute Profile

[eIDAS-Interop] eIDAS: Interoperability Architecture

[eIDAS-SAML] eIDAS: SAML Message Format

[ICAO 9303] ICAO: Doc 9303, Machine Readable Travel Documents


[RFC2119] IETF: RFC 2119, S. Bradner: Key words for use in RFCs to Indicate Requirement Levels


[SAML] OASIS: Security Assertion Markup Language (SAML) V2.0


[TR-03116] BSI: Technische Richtlinie TR-03116, Kryptographische Vorgaben für Projekte der Bundesregierung


[TR-03124] BSI: Technische Richtlinie TR-03124, eID-Client


[TR-03128] BSI: Technische Richtlinie TR-03128, Diensteanbieter für die eID-Funktion

[TR-03129] BSI: Technische Richtlinie TR-03129, PKIs for Machine Readable Travel Documents -- Protocols for the Management of Certificates and CRLs

[TR-03129] BSI: Technical Guideline TR-03129, PKIs for Machine Readable Travel Documents

/XML-Enc W3C: XML Encryption Syntax and Processing