Federal Office
for Information Security

Technical Guideline TR-03124-1
eID-Client –
Part 1: Specifications

Version 1.0
9. August 2013

Federal Office for Information Security
Post Box 20 03 63
D-53133 Bonn
Phone: +49 22899 9582-0
E-Mail: eid@bsi.bund.de
Internet: https://www.bsi.bund.de

Table of Contents

List of Figures

1 Introduction

This Technial Guideline specifies the eID-Client software for Online-Authentication based on Extended Access Control Version 2 (EAC2) between an eService and an eID-Card, e.g. the German eID-Card or the German electronic Residence Permit. The eID-Client implements the client side of this authentication. The server side is implemented by the eID-Server, see [TR-03130],part 1.

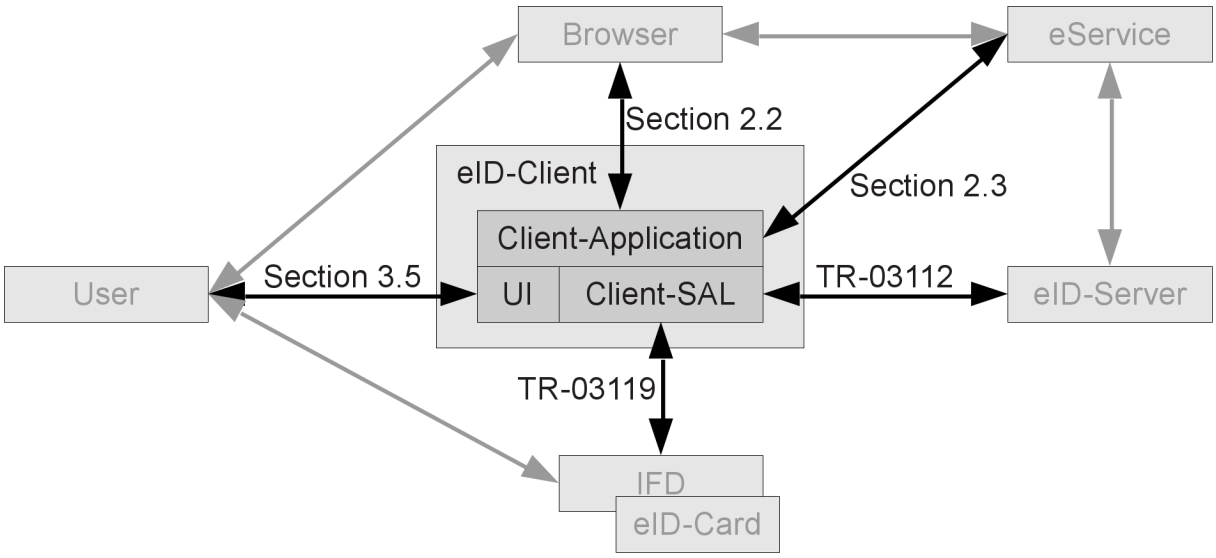
The same software can also support other functionalities and interfaces, like signature creation or support of other smart cards. This is out of scope of this document, as is the offline use of an eID-Card.

The eID-Client is based on the eCard-API-Framework ([TR-03112]) and implements at least the functions of this API necessary to support Online-Authentication, listed in section 3.2. Both the eID-Client and the eID-Server contain at least a Service Access Layer (SAL) according to part 4 of [TR-03112], called Client-SAL and Server-SAL, respectively, in the following. The eID-Client contains also the necessary functions to access a card reader according to section 3.4/[TR-03119].

1.1 Modules

The eID-Client can be implemented as a single product, or split up into different modules. This Guideline uses the following conventions:

- The term **eID-Client** denotes the complete client-software necessary to perform Online-Authentication, either implemented as a single product or as separate modules.
- The term **Client-Application** denotes the part/module(s) of the eID-Client which implements the communication with the web-browser and the eService according to section 2.
- The term **Client-SAL** denotes the part/module(s) of the eID-Client which implements the necessary functions of the eCard-API-Framework [TR-03112] according to section 3.2. The functions necessary to access a card reader can be implemented as a separate module *IFD-Interface* according to part 6 of [TR-03112] or as part of the Client-SAL. For the purpose of this Guideline, the IFD-Interface (if present) is treated as part of the Client-SAL.
- The **User Interface (UI)** contains all elements necessary for user interaction, see section 3.5.



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 word "CONDITIONAL" is to be interpreted as follows:

CONDITIONAL: 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.

2 Online-Authentication based on EAC2

This section specifies the establishment of a connection between eID-Client and eID-Server, including binding to a preexisting connection between browser and eService, for performing Online-Authentication based on Extended Access Control Version 2.

2.1 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”.

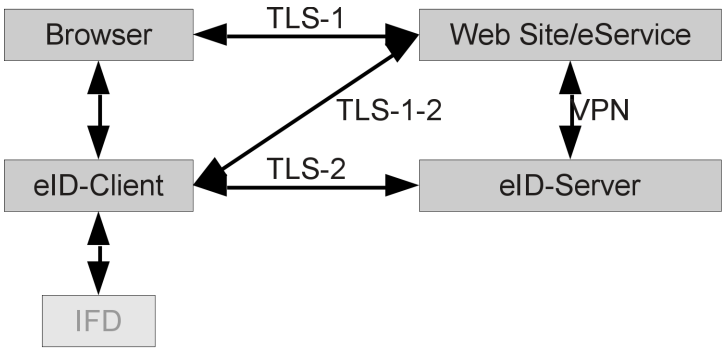


Figure 1: Generic Communication Model

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 channel, see [TR-03130], part 1.

In the special case of both channels TLS-1 and TLS-2 terminating at the same domain (“*Attached eID-Server*”), the intermediary channel TLS-1-2 is not necessary. If TLS-1-2 is not used, the same TLS certificate MUST be used for both channels TLS-1 and TLS-2.

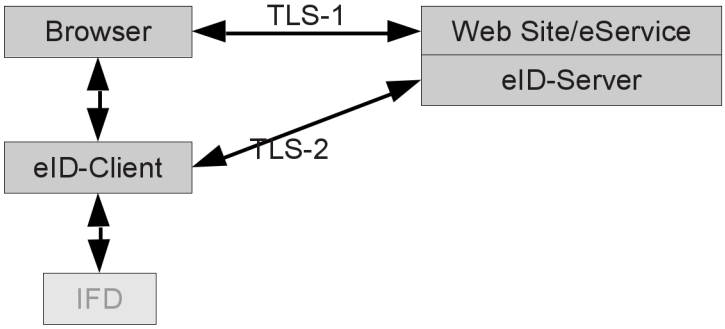


Figure 2: Attached eID-Server

Note: If browser and eID-Client are integrated into one single component, TLS-2 and TLS-1 MAY be the same channel, i.e. no separate channel TLS-2 is necessary.

2.2 Client-Interface

The eID-Client offers its services at http://127.0.0.1:24727/eID-Client. This interface MUST NOT be available to external callers, i.e. must only be open locally.

The eID-Client is activated by sending a HTTP GET to the URL http://127.0.0.1:24727/eID-Client?tcTokenURL=*URL*. Here *URL* is a (properly encoded) https-URL where the TC Token (see 2.3) for this authentication can be retrieved by the eID-Client.

The response to this GET command contains a HTTP redirect pointing back to the website of the eService.

Note: In general, URLs (including parameters) used in or intended for use in HTTP GET commands processed by the browser SHOULD NOT exceed a size of 2kB, since some browsers reject longer URLs.

To embed a call to the eID-Client into the website of an eService, two mechanisms are available:

- Embedded Link
- MIME-Type

The mechanisms only differ in the integration of the call to the eID-Client into the website, the interface of the eID-Client is the same for both. The mechanisms are described in the following.

2.2.1 Embedded Link

The website contains an embedded link to the address http://127.0.0.1:24727 as defined above.

This method has the advantage that no components installed in the browser (e.g. plugins, extensions) are necessary. All platforms which allow to install software and to open or register port 24727 locally can be supported independently of the browser.

Disadvantages are:

- Some browsers might warn the user if he activates the link that he is connecting to an insecure site (http instead of https used for TLS-1).
- If no eID-Client is available, the error message to the user is “website not found”, which does not convey the actual problem.

2.2.2 MIME-Type

Alternatively, the website can embed an `<object>` (see [HTML]) of MIME type `application/vnd.eid-client`. In this case, a handler for this MIME type must be available in the browser.

The `data`-attribute of this object MUST contain the URL of the TC Token.

The text in the body of the object (which is displayed in case no handler for this MIME type is available) SHOULD contain a meaningful text and an embedded link as described above, so that the MIME type-method falls back gracefully to the embedded link-method in this case.

The handler for this MIME type MUST perform the following:

- Call http://127.0.0.1:24727/eID-Client?tcTokenURL=*URL*, where *URL* is the (properly encoded) content of the `data` attribute of the `<object>`. Unknown parameters MUST be ignored. If no eID-Client is available, a meaningful error message SHOULD be displayed to the user.
- The HTTP Response to the above call is a redirect. The handler directs the browser to load this page.

The main advantage of this method is the possibility of the handler to display a meaningful message to the user in case no eID-Client is available. The main disadvantage is the dependency on a (browser specific) handler for this MIME type available in the browser.

An example of an embedded object is given below:

```
<object type="application/vnd.eid-client"
    data="https://service.example.de/tctoken?A876 ... 64CD">
  <a href = "http://127.0.0.1:24727/eID-Client?tcTokenURL=
    https%3A%2F%2Fservice.example.de%2Ftctoken%3FA876 ... 64CD"> Start eID
  </a>
</object>
```

2.3 TC Token

The TC Token is used to convey the necessary information for setting up a Trusted Channel between eID-Client and eID-Server from the eService to the eID-Client. Furthermore, the retrieval of the TC Token and the setting up of the Trusted Channel is part of the channel binding mechanism to bind TLS-1 and TLS-2 (see section 2.6).

The TC Token MUST be provided by the eService at the Token URL given in the embedded link or the `data` attribute of the `<object>`. The Token is an XML Fragment (without prolog, e.g. document type or namespace declarations) of `TCTokenType`:

```
<complexType name="TCTokenType">
  <sequence>
    <element name="ServerAddress" type="anyURI" />
    <element name="SessionIdentifier" type="string" />
    <element name="RefreshAddress" type="anyURI" />
    <element name="CommunicationErrorAddress" type="anyURI" minOccurs="0" />
    <element name="Binding" type="anyURI" />
    <element name="PathSecurity-Protocol" type="anyURI" minOccurs="0" />
    <element name="PathSecurity-Parameters" minOccurs="0">
      <complexType>
        <choice>
          <element name="PSK" type="hexBinary" />
        </choice>
      </complexType>
    </element>
  </sequence>
</complexType>
```

The contents of the elements are defined as follows:

- `ServerAddress [anyURI]` (REQUIRED)
  MUST contain a https-URL which shall be used by the eID-Client to connect to the eID-Server.
- `SessionIdentifier [string]` (REQUIRED)
  MUST contain a unique identifier of the current authentication session.
- `RefreshAddress [anyURI]` (REQUIRED)
  MUST be a https-URL. The eID-Client redirects the browser to this URL (or the URL retrieved by following redirects starting from this URL, see 2.4.5) after conclusion of the Online-Authentication. The `RefreshAddress` (and, if applicable, all URLs encountered as redirect URLs in the procedure described in 2.4.5) SHOULD be unpredictable, i.e. contain a randomly generated sessionID or similar, which SHOULD NOT be derivable from any information transmitted in TLS-1, including session cookies or similar.

- `CommunicationErrorAddress [anyURI, 0..1]` (OPTIONAL)
  If present, the eID-Client redirects the browser to this URL if an communication error occurred and no valid refreshURL could be determined, see section 2.4.5.3). The URL MAY contain a session ID.
- `Binding [anyURI]` (REQUIRED)
  MUST be set to `urn:liberty:paos:2006-08` to indicate PAOS v2.0 binding.
- `PathSecurity-Protocol [anyURI, 0..1]` (CONDITIONAL)
  This element MUST be present if two channels TLS-1-2 and TLS-2 are used (see section 2.1). If present, the content of this element MUST be set to `urn:ietf:rfc:4279` to indicate TLS with cipher suite using a pre-shared key. Note that also PSK cipher suites from subsequent RFCs to RFC 4279 MAY be negotiated during the TLS handshake.
- `PathSecurity-Parameters [0..1]` (CONDITIONAL)
  MUST be present if `PathSecurity-Protocol` is present. In this case, SHALL contain additional parameters for PathSecurity, depending on the protocol indicated in `PathSecurity-Protocol`. Since only TLS with pre-shared key is allowed currently, only one choice is available:
  - `PSK [hexBinary]` (REQUIRED)
    MUST contain a pre-shared key which will be used for connection establishment between client SAL and server SAL. The `PSK` SHALL be unpredictable and cryptographically strong. The `PSK` MUST NOT be derivable from any information transmitted in TLS-1, including session cookies or similar.

An example of a TC Token is given below:

```
<TCTokenType>
  <ServerAddress> https://eid-server.example.de/entrypoint </ServerAddress>
  <SessionIdentifier> 1A2B … B129 </SessionIdentifier>
  <RefreshAddress>
    https://service.example.de/loggedin?7eb3...9f62
  </RefreshAddress>
  <CommunicationErrorAddress>
    https://service.example.de/ComError?7eb3...9f62
  </CommunicationErrorAddress>
  <Binding> urn:liberty:paos:2006-08 </Binding>
  <PathSecurity-Protocol> urn:ietf:rfc:4279 </PathSecurity-Protocol>
  <PathSecurity-Parameters>
    <PSK> 4BC1 … A0B5 </PSK>
  </PathSecurity-Parameters>
</TCTokenType>
```

Note: The retrieval of the TC Token via the channel TLS-1-2 ensures that the relevant data are retrieved from a source whose authenticity can be checked by the eID-Client by comparing the server certificate of TLS-1-2 against the eService CV certificate checked by the Terminal Authentication.

2.4 Online-Authentication

The Online-Authentication is performed by the following steps (see also Figure 3):

2.4.1 Client Activation

The eID-Client is activated by sending a HTTP GET as described in 2.2 to the eID-Client.

2.4.2 Retrieval of TC Token

Starting with the TC Token-URL submitted to the eID-Client in 2.4.1, the following SHALL be performed:

1. The eID-Client connects to the URL and submits a HTTP GET. The server certificate retrieved during the TLS handshake SHALL be stored by the eID-Client to be used for the certificate check in 2.4.4. An additional chain verification of the certificate SHOULD NOT be performed, the authenticity of the certificate is checked via the mechanism described in 2.4.4.
2. If a redirect ("302 Found", "303 See Other" or "307 Temporary Redirect") is returned, the procedure is repeated from step 1 with the URL given in the "Location" of the redirect.

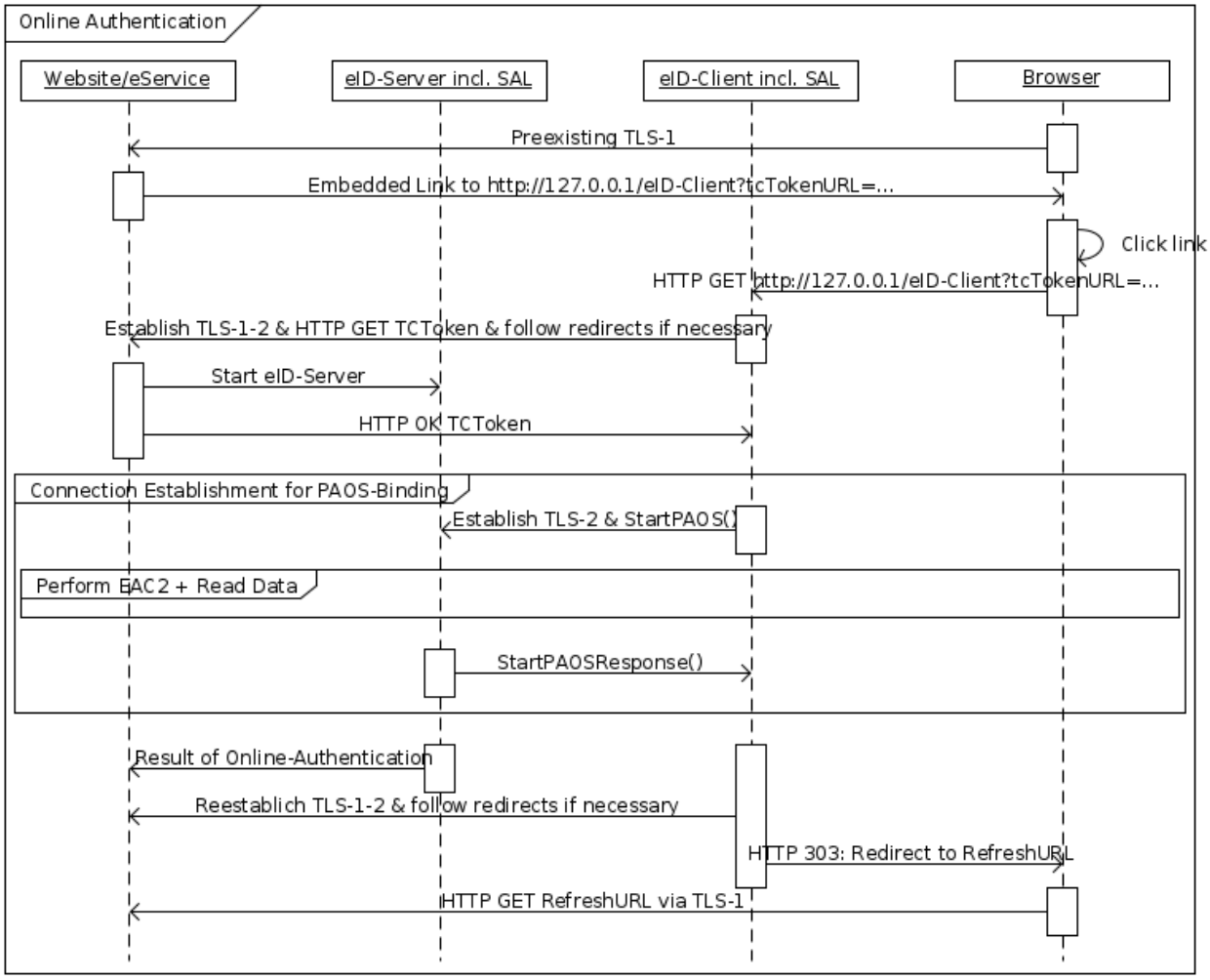


Figure 3: Online-Authentication (Simplified)

The HTTP response containing the TC Token SHALL have `Content-Type: text/xml` and `charset=utf-8`. The response SHALL NOT contain any further content besides the TC Token.

Note: To enhance interoperability, the eID-Client SHOULD not check the correct format of the HTTP response but accept malformed responses from the eService as far as possible.

If any of the URLs encountered during this procedure is not a https-URL, the eID-Client MUST NOT connect to this URL, SHALL abort the procedure, and SHALL report a communication error as described in section 2.4.5.3.

If the TC Token cannot be retrieved by this procedure, the eID-Client SHALL return a HTTP error "404 Not Found" to the browser. The eID-Client SHOULD include a meaningful human-readable error message/description into the body of the response to the browser.

If the eService encounters an error during generation of the TC Token (e.g. the eID-Server could not be reached), the eService MAY return a TC Token with empty elements except the `CommunicationErrorAddress`. In this case, the eID-Client SHALL return a communication error as described in section 2.4.5.3.

The eService MAY use the request for a TC Token by the eID-Client as a signal to activate the eID-Server as specified in [TR-03130], part 1.

Note: The redirect mechanism described above is used to facilitate indirect communication between eService and eID-Server, e.g. using SAML with redirect binding as described in section 2.5. Other communication protocols which can be mapped to redirect binding are also possible.

2.4.3 Connection Establishment

The eID-Server and the eID-Client send a `TC_API_Open` to the Server-SAL and the Client-SAL, respectively, as specified in [TR-03112], part 7, using `SessionIdentifier`, `Binding`, `PathSecurity-Protocol` and `PSK` as extracted from the retrieved TC Token as parameters. If eID-Client and Client-SAL (or eID-Server and Server-SAL) are an integrated component, no explicit call to `TC_API_Open` in the client (or in the server) is necessary.

A TLS channel between Client-SAL and Server-SAL is negotiated and a PAOS connection using `StartPAOS` established. If a `PSK` is given in the TC Token, a PSK Cipher Suite MUST be used. If no `PathSecurity-Protocol/PSK` is given in the TC Token, the same TLS channel as established to retrieve the TC Token MUST be used for the PAOS connection, i.e. a new channel MUST NOT be established.

If the `TCToken` contains an empty `ServerAddress` or any error occurs during connection establishment, the eID-Client SHALL return to the web session (see 2.4.5) indicating the error.

2.4.4 Online-Authentication

Online-Authentication is performed using the Extended Access Control protocol as specified [TR-03112], part 7, section 3.6.

As part of the Online-Authentication the eID-Client MUST perform the following checks to ensure the authenticity of the TLS certificates immediately after the eService certificate is sent to the eID-Client:

- The eID-Client MUST check that the hashes of all server certificates stored during 2.4.2 and the hash of the server certificate of the eID-Server are contained in the `CertificateDescription` extension of the eService certificate.
- The eID-Client MUST check that the TC Token-URL submitted in 2.4.1 and the `subjectURL` contained in the `CertificateDescription` extension of the eService certificate conform to the Same-origin policy according to [RFC6454].

If any of these checks fails, the eID-Client SHALL abort the Online-Authentication and return to the web session (see 2.4.5) indicating the error.

Note: The eID-Client SHOULD employ suitable mechanisms to avoid the browser to time out during Online-Authentication. Examples of possible mechanisms are

- *Sending a HTTP Response Header after the GET*
- *Sending a HTTP "303 See Other" pointing back to the eID-Client in regular intervals*
- *Sending a HTTP "102 Processing" after the GET*[1].

2.4.5 Return to the web-session

Returning to the web-session comprises the following steps:

- If a TC Token including a non-empty `RefreshAddress` was successfully retrieved, the eID-Client SHALL determine the refreshURL as specified in section 2.4.5.1;
- If a valid refreshURL was determined, the eID-Client SHALL redirect the browser to the eService (section 2.4.5.2);
- If no valid refreshURL could be determined, the eID-Client SHALL report a communication error (section 2.4.5.3).

Note: The mechanism described in this section ensures that the user/browser is redirected to a website, which is confirmed to belong to the holder of the eService certificate, i.e. the channel TLS-1 is bound to the (successful) Online-Authentication.

The channel TLS-1 before and after the Online-Authentication might not be the same network connection if the TLS-1 connection is dropped by either the browser or the web server during Online-Authentication. The channel binding mechanism connects the Online-Authentication only to TLS-1 after the authentication.

The eService is responsible to provide continuity (if necessary) of the channel TLS-1 before and after the authentication and therefore SHOULD employ suitable mechanisms, e.g. cryptographically strong session cookies.

2.4.5.1 Determination of the refreshURL

The following procedure SHALL be performed to determine the refresh URL:

- If the `RefreshAddress` given in the TC Token and the `subjectURL` contained in the `CertificateDescription` extension of the eService certificate conform to the Same-origin policy according to [RFC6454], the `RefreshAddress` is used as refresh URL.
- Otherwise, starting with the URL given as `RefreshAddress` in the TC Token, the following algorithm SHALL be used to determine the refresh URL:

1 Example: The eID-Client sends the following as response after GET:

```
HTTP/1.1 102
Content-Length: 0
Connection: close
Server: eID-Client
```

After finishing the Online-Authentication the client sends the following as Content of the first answer:

```
HTTP/1.1 303 See Other
Content-Length: 0
Connection: close
Location: https://service.example.de/loggedin?7eb3...9f62
```

1. The eID-Client establishes a connection to this URL and performs a HTTP GET to the URL. If the response is not a redirect (“302 Found”, “303 See Other” or “307 Temporary Redirect”), the procedure is aborted and the eID-Client SHALL report a communication error as described in section 2.4.5.3.
2. If the URL and the `subjectURL` contained in the `CertificateDescription` extension of the eService certificate conform to the Same-origin policy according to [RFC6454], the procedure is finished with the URL given in the “Location” as refresh URL, otherwise repeat from step 1 with the “Location” given in the redirect as URL.

The eID-Client SHALL establish a connection (i.e. perform a TLS handshake without HTTP interaction) to the refresh URL to retrieve the server certificate of the refresh URL.

If any of the URLs (including the refresh URL) encountered during this procedure

- is not a https-URL, or
- the hash of the retrieved server certificate is not contained in the `CertificateDescription` extension of the eService certificate,

the eID-Client SHALL abort the procedure, and SHALL report a communication error as described in section 2.4.5.3. An additional chain verification of the certificates SHOULD NOT be performed (see also section 4.4).

If the `subjectURL` is not known, e.g. because the Online-Authentication was aborted before the eService certificate is known to the eID-Client, the TCToken-URL as delivered to the eID-Client by the calling application (see 2.4.1) SHALL be used in the procedure above instead of the `subjectURL` for the checks according to the Same-origin policy, and the check against the `CertificateDescription` extension of the eService certificate SHALL be skipped.

Note: The redirect mechanism described above is used to facilitate indirect communication between eService and eID-Server, e.g. using SAML with redirect binding as described in section 2.5. Other communication protocols which can be mapped to a redirect binding are also possible.

2.4.5.2 Redirecting the Browser to the eService

If a refresh URL was successfully determined, the eID-Client SHALL answer the HTTP GET, which initiated the Online-Authentication (see 2.4.1), by returning “303 See Other” with the refresh URL determined by the above procedure as target, with added URL-parameter

- `ResultMajor=ok`, if no error has occurred, or
- `ResultMajor=error&ResultMinor=`*res_min*, if an error occurred; it is RECOMMENDED to include an additional parameter `ResultMessage=`*tls_alert_description* parameter using a value defined in [TLS-Alerts], if the error is due to a TLS-related problem.[2]

2.4.5.3 Communication error without refreshURL

If no refresh URL could be determined, the eID-Client SHALL report a communication error:

2 Example 1: An unknown `psk_identity` will yield the following error: `ResultMajor=error&ResultMinor=trustedChannelEstablishmentFailed&ResultMessage=unknown_psk_identity`.
Example 2: If the eCard-API-Framework has not been initialized, there will be the following error: `ResultMajor=error&ResultMinor=notInitialized`.

- if a `CommunicationErrorAddress` from the TC Token is available, the eID-Client SHALL redirect the browser to the `CommunicationErrorAddress` with added URL-Parameter `ResultMajor=error&ResultMinor=communicationError`;
- if no `CommunicationErrorAddress` is available, the eID-Client SHALL convey a HTTP error "400 Bad Request" to the browser. The eID-Client SHOULD include a meaningful human-readable error message/description into the body of the response to the browser.

In both cases, the failure to determine a valid refreshURL MUST be signaled clearly and unambiguously to the user by the eID-Client.

2.5 SAML

This section describes the embedding of the Online-Authentication into a SAML (see [SAML-Core]) based authentication framework as an example of the redirect-mechanism described in 2.4.2 and 2.4.5.

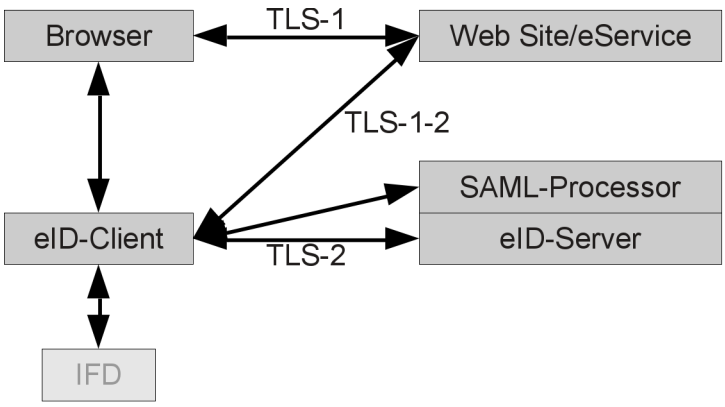


Figure 4: Communication Model for SAML based Authentication

The process flow is as follows:

- The Online-Authentication is invoked as described in 2.2.
- The eID-Client tries to retrieve the TC Token from the eService. The eService responds with a redirect to the SAML-Processor containing a SAML Authentication Request in HTTP Redirect Binding (see [SAML-Binding]). The eID-Client connects to the SAML-Processor with the URL provided by the eService and retrieves the TC Token. See section 2.4.2.
- Online-Authentication is performed as described in sections 2.4.3 and 2.4.4.
- The `RefreshAddress` in the TCToken points to the SAML-Processor. Since this URL and the `subjectURL` from the eService certificate do not share the same origin, the eID-Client connects to the `RefreshAddress`. The SAML-Processor returns a redirect to the eService, containing the SAML Authentication Response in HTTP Redirect Binding (see [SAML-Binding]). The eID-Client connects to the eService using this URL and thereby transmitting the result of the authentication to the eService. The eService returns a redirect, which is sent by the eID-Client to the web browser. See section 2.4.5.

The communication between eService and SAML-Processor (i.e. the SAML-Request and the SAML-Response) MUST be end-to-end encrypted and authenticated. It is RECOMMENDED to carefully consider the well known security aspects of SAML-based systems (see [SAML-Sec]).

The eService MUST ensure that the SAML-Response is received through the same TLS-channel as the SAML-Request was sent. Depending on the details of the eService implementation, this can in particular involve appropriate configuration of the TLS session cache parameters.

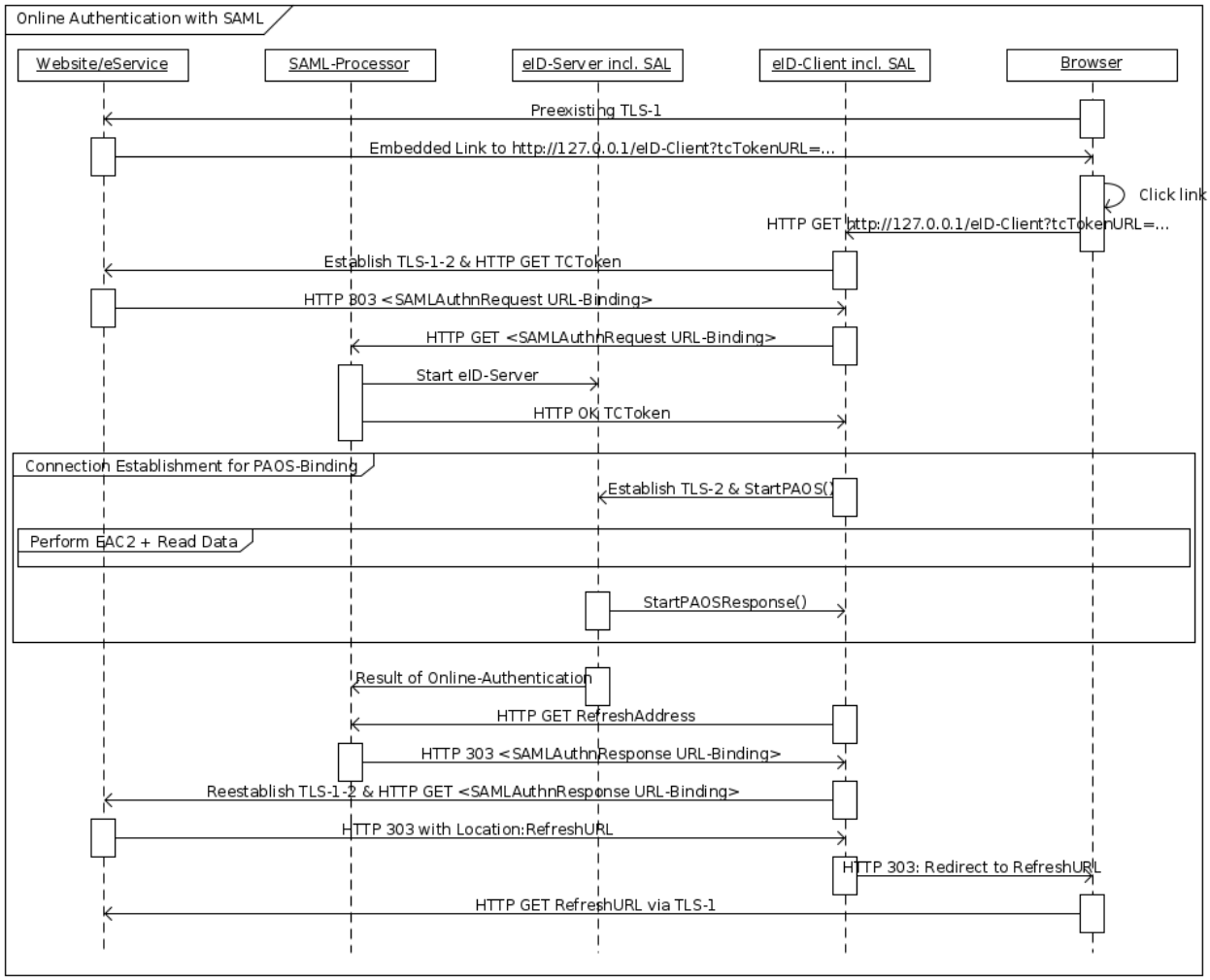


Figure 5: Online-Authentication with SAML (Simplified)

For further details on using SAML, including Request and Response format, see [TR-03130], part 1.

2.6 Channel Binding

This section discusses the security of the binding of TLS-1 to the EAC2-based Online-Authentication.

In the generic communication model, a pre-shared key (PSK) is transmitted in TLS-1-2 which is used for connection establishment of TLS-2. Provided TLS-1-2 is not attacked by a man-in-the-middle (which would be detected later on by the check of the server-certificate of TLS-1-2 against the CV certificate of the eService), the PSK is only known to the eID-Client, the eService and the eID-Server (the latter two communicate trustworthy). Therefore TLS-1-2 and TLS-2 are strongly bound and it suffices to consider the Attached eID-Server model, where TLS-1 and TLS-2 are terminating at the same domain.

In the Attached eID-Server model, the following ingredients bind TLS-1 and TLS-2:

- The eID-Client checks the URL of TLS-2 and the refresh URL determined according to section 2.4.5 against the `subjectURL` from the eService CV certificate
- The eID-Client checks the TLS certificate of TLS-2 against the eService CV certificate
- The eID-Client redirects the Browser to the refresh URL after Online-Authentication.

Since the refresh URL is specific for a single session, this not only binds the endpoints of TLS-1 and TLS-2, but also the specific sessions.

In principle, the following attack remains possible: An attacker could try to intercept the redirect to the refresh URL either locally (as a man-in-the-browser) or remotely (by DNS-Spoofing combined with a faked certificate for the refresh URL). The local attack can only be thwarted by using a non-compromised browser. The remote attack requires a very high attack potential and access to the DNS resolver of the user[3] as well as a rogue TLS-CA, which is accepted by the used browser. This potential attack is subverted if DNSSEC is used.

3 Functional Requirements for an eID-Client

An eID-Client compliant to this Technical Guideline MUST support

- Online-Authentication using an eID-Card based on EAC2
- PIN-Management of eID-Cards based on EAC2, i.e.
  - setting a PIN using the transport PIN delivered to the citizen by letter,
  - setting a new PIN using the current PIN (this includes setting an operational PIN using the transport PIN),
  - resuming a suspended PIN using the CAN, and
  - unblocking a blocked PIN using the PUK.

In order to implement these functionalities, the requirements detailed in the following subsections MUST be fulfilled.

3.1 Connection Establishment for Online-Authentication

The eID-Client SHALL support the method for Connection Establishment for Online-Authentication as specified in section 2.

3.2 eCard-API-Profile

The eID-Client SHALL implement at least the following commands of the eCard-API [TR-03112] to support Online-Authentication:

- As caller:
  - `StartPAOS` ([TR-03112], Part 7) (REQUIRED)
- As callee:
  - `InitializeFramework` ([TR-03112], Part 3) (REQUIRED)
  - `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)

3 The attacker needs to answer the query of the eID-Client for the IP address of the refresh URL with the correct answer, while providing the spoofed answer to the same query from the browser. Due to the caching of queries, e.g. in the network stack of the computer, access to a random DNS server is not sufficient.

If Client-Application and Client-SAL are implemented as separate modules, the following commands SHALL be implemented by the Client-Application as caller and the Client-SAL as callee:

- `Initialize`/`Terminate` ([TR-03112], Part 4) (CONDITIONAL)
- `CardApplicationPath`/`CardApplicationConnect`/`CardApplicationDisconnect` ([TR-03112], Part 4) (CONDITIONAL)
- `TC_API_Open`/`TC_API_Close` ([TR-03112], Part 7) (CONDITIONAL)
- `DIDUpdate` ([TR-03112], Part 4) with support for `DIDMarker` of type `PACEMarkerType` ([TR-03112], Part 7) (CONDITIONAL)

3.3 HTTP Communication

All HTTP communication (interfaces to browser, eService and eID-Server) SHALL follow [RFC2616] and [RFC2818], where applicable. This includes the ability to communicate via a CONNECT-proxy to the eService and the eID-Server. Appropriate configuration options SHOULD be available to configure the proxy settings.

3.4 Card Reader Interface

The eID-Client SHALL support at least all card readers compliant to [TR-03119] intended for use with the host platforms the eID-Client supports. The Technical Guideline [TR-03119] specifies communication with the card reader via PC/SC and via CCID. The eID-Client SHOULD support both interfaces.

The eID-Client SHALL support the following calls as caller if different card readers can be used:

- GetReaderPACECapabilities (REQUIRED)
- EstablishPACEChannel/DestroyPACEChannel (REQUIRED)
- ModifyPIN (REQUIRED)

In case of devices with embedded card readers, the communication of the eID-Client with the reader is out of scope of this Technical Guideline.

3.5 User Interface

The eID-Client SHALL provide a suitable user interface to support Online-Authentication and PIN-Management, as detailed below. The user interface MUST be clearly distinguishable from the web browser by the user.

The eID-Client SHALL provide the following user interface as part of Phase 1 of the Extended Access Control protocol (see [TR-03112], Part 7, Section 3.6) before the user enters his PIN:

- The eID-Client SHALL offer an interface to the user to display and to restrict the access rights requested by the eService (“CHAT restriction”).
  - The eID-Client SHALL use the texts from [TR-03119], Section A.5.1, to denote the access rights.
  - The eID-Client SHOULD preselect all access rights given in the `RequiredCHAT`- or `OptionalCHAT`-elements in `EAC1InputType` (if present). If these elements are not present, the eID-Client SHOULD preselect all access rights contained in the CHAT of the eService certificate.
  - The eID-Client SHALL allow the user to deselect the access rights not contained in `RequiredCHAT` (if present), or of all access rights if no `RequiredCHAT` is present.

- The eID-Client SHOULD NOT allow selection of rights which are not granted in the CHAT of the eService certificate.

The restricted CHAT SHALL be transmitted to the card as part of PACE and to the eID-Server in the `CertificateHolderAuthorizationTemplate` element of `EAC1OutputType`.

- The eID-Client SHALL display the following information extracted from the data transmitted from the eID-Server:
  - the Holder of the certificate as contained in the `CertificateDescription` (`subjectName` and `subjectURL`);
  - if Age Verification is requested, the intended comparison data from `AuthenticatedAuxiliaryData`. The eID-Client SHALL ensure that the `AuthenticatedAuxiliaryData` transmitted during Terminal Authentication match the comparison data presented to the user.

  Additionally, the following information extracted from the data transmitted from the Server-SAL MAY be displayed and MUST be displayed upon request by the user:
  - further information on the eService, the purpose of data transmission and information regarding the responsible office for data protection as contained in the `termsOfUsage` in the `CertificateDescription`;
  - information regarding the issuing Document Verifier (DV) of the certificate (`issuerName` and `issuerURL` (if present) in the `CertificateDescription`).

  The eID-Client SHALL ensure that the eService certificate used during Terminal Authentication is identical to the certificate presented to the user.

The eID-Client SHALL provide a clear interface which

- informs the user about the current status of the Online-Authentication,
- informs the user about communication errors, e.g. failure to determine a valid refreshURL, and
- informs the user if connection establishment (including the specified checks) as described in section 2 or the authentication (especially Terminal Authentication) fails.

As part of Online-Authentication and PIN Management, the eID-Client SHALL

- inform the user about the current value of the retry counter of the PIN if PIN verification fails;
- enable the user to retry PIN verification, and
- enable the user to resume the PIN by entering the CAN if the PIN is suspended.

If a card terminal with secure PIN entry (PIN Pad) is available, this MUST be used to enter PIN or PUK. Otherwise, the keyboard of the host of the eID-Client can be used.

For security reasons, the eID-Client SHALL advise the user to remove the card from the reader as soon as a process (Online-Authentication, PIN Management) has finished.

3.6 Test Mode

It is RECOMMENDED to implement a Test Mode as follows:

- If any of the security checks specified in section 2 fails, the eID-Client SHOULD, instead of aborting the procedure, continue if possible and inform the user about the exact reason of the failure.
- If the eID-Client implements the optional pre-verification of the eService certificate (see [TR-03112], Part 7, Section 3.6.5), the pre-verification SHOULD be based on a test-PKI.

- The Test Mode SHOULD provide enhanced user accessible log data. Log data MAY also be provided outside of the Test Mode.

The Test Mode MAY be implemented as a separate executable or as a configuration option. In both cases the Test Mode MUST be clearly signaled to the user.

4 Security Requirements for an eID-Client

An attacker SHOULD be considered to be able to send arbitrary commands/data to publicly accessible interfaces. In this context, interfaces protected by access control are to be considered public until access control has been performed properly.

An implementor can consider the underlying operating system to be not compromised. Nevertheless, security mechanisms and heuristics SHOULD be implemented which mitigate the effects if the operating system or other third party components are compromised.

4.1 Interfaces

Calls to the eCard-API web service interface (if present/see [TR-03112], Part 1) and to the Client-Interface of the eID-Client (see Section 2.2) from origins other than `localhost` MUST NOT be accepted.

The eID-Client SHOULD restrict the available calls to the minimum required for the desired functionality. This includes restrictive access conditions depending on the current state of the eID-Client.

4.2 Active Web Content

The eID-Client MUST NOT rely on active web content for required functional or security features. In this context active web content is defined as code downloaded as part of a website which is executed on the host computer without prior installation, e.g. java-applets. JavaScript MAY be used for non security critical operations in web browsers supporting sand-boxing, a fall-back without JavaScript SHOULD be available.

4.3 Functionalities

The eID-Client implements the following security relevant functionalities:

- PIN-Pad for card readers without secure PIN entry (i.e. a “basic reader” according to [TR-03119])
- Display of information from the eService certificate/Restriction of access rights (see Section 3.5)
- Channel establishment and channel binding for Online-Authentication (see Section 2)
- (OPTIONAL) Pre-verification of the eService certificate prior to display (see [TR-03112], Part 7, Section 3.6.5).

These functionalities MUST be suitably protected against manipulation.

4.4 Cryptography

For support of card readers without secure PIN entry, PACE with the cryptographic algorithms given in [TR-03116], Part 2, MUST be implemented.

For all TLS channels, the requirements from [TR-03116], Part 4, MUST be fulfilled. This includes the TLS version to be supported and the selection of supported Cipher Suites. In deviation from [TR-03116], Part 4,

the cipher suite `TLS_RSA_PSK_WITH_AES_256_CBC_SHA` MUST be supported for TLS-2 in the generic communication model. Additional PSK cipher suites according to [TR-03116], Part 4, MAY be supported.

Since the authenticity of the TLS certificate is checked via the mechanism described in 2.4.4, an additional chain verification of the TLS certificate is not necessary and SHOULD NOT be performed. The eID-Client MUST perform name matching according to [RFC2818]. If a TLS certificate contains revocation information, a revocation check MUST be performed.

4.5 Updates

The vendor of the eID-Client SHALL supply necessary security updates. The update mechanism MUST be cryptographically protected against manipulation. If the host operating system offers a protected update mechanism, this mechanism SHOULD be used.

5 Certification

5.1 Conformity Certification

The eID-Client can be certified for conformity according to Part 2 of this Technical Guideline.

5.2 Security Certification

The eID-Client can be certified according to the Common Criteria Protection Profile [PP-Client-eID].

Reference Documentation

PP-Client-eID	BSI: Common Criteria Protection Profile "eID-Client based on eCard-API", BSI-PP-0066
TR-03112	BSI: Technische Richtlinie TR-03112, eCard-API-Framework
TR-03116	BSI: Technische Richtlinie TR-03116, Technische Richtlinie für die eCard-Projekte der Bundesregierung
TR-03119	BSI: TR-03119: Requirements for Smart Card Readers supporting eID and eSign based on Extended Access Control
TR-03130	BSI: TR-03130: eID-Server
TLS-Alerts	IANA: TLS Alert Registry
RFC2119	IETF: RFC 2119: S. Bradner: Key words for use in RFCs to Indicate Requirement Levels
RFC2616	IETF: RFC 2616: R. Fielding, J. Gettys, J. Mogul, H. Frystykc, L. Masinter, P. Leach, T. Berners-Lee: Hypertext Transfer Protocol -- HTTP/1.1
RFC2818	IETF: RFC 2818: E. Rescorla: HTTP Over TLS
RFC6454	IETF: RFC 6454: A. Barth: The Web Origin Concept
SAML-Core	OASIS: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0
SAML-Binding	OASIS: Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0
SAML-Sec	OASIS: F. Hirsch, R. Philpott, E. Maler: Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0
HTML	W3C: HTML 4.01 Specification