FedICT eID Applet Developer's Guide

Integrating the eID Applet within your web applications.

Frank Cornelis

draft 26 Jan 2009

Copyright © 2008-2009 FedICT

Abstract

This developer's guide serves as an entry point for integrating the eID Applet in your web applications.

1. Introduction 1
2. eID Applet 2
3. eID Applet Service 3
    3.1. eID Identification 4
    3.2. eID Authentication 7
    3.3. eID Signatures 9
    3.4. eID Administration 10
    3.5. Generic eID Applet Service settings 11
4. eID Applet Protocol 13
    4.1. eID Applet Protocol Messages 13
A. eID Applet Developer's Guide License 19
B. eID Applet License 20
C. Revision history 20

1. Introduction

The FedICT eID Applet is a browser component that exposes the functionality of an eID card to your web applications. The main features of the eID Applet are:

- Easy to integrate within an existing web application.
- Security and privacy of the citizen is protected.
- Interactive eID card handling.

Mac OS X

Because Apple does not fully support the Java 6 runtime on their Mac OS X operating systems, the identification functionality will not work.

The strategy is to no longer support operating systems, but to support a specific Java platform. For the eID Applet this is the Java 6 platform (for eID identification that is. eID authentication and eID signature creation can also run using a Java 1.5 JRE). We can only give advice on how to configure Java 6 on your operating system.

Linux Fedora 9

The Fedora operating system comes by default with the IcedTea JRE which is based on the OpenJDK. If the Firefox browser uses this JRE the eID Applet refuses to run. Please download the official Sun Java 6 JRE and also enable it in the Firefox browser. The Firefox plugins can be configured via symbolic links under: `/usr/lib/mozilla/plugins` . Remove the IcedTea JRE link via: `rm /usr/lib/mozilla/plugins/libjavaplugin.so` . Afterwards add a symbolic link to the Sun JRE plugin, which can be found under: `$JAVA_HOME/jre/plugin/i386/ns7/libjavaplugin_oji.so` . Check the installed plugins in Firefox by navigating to: `about:plugins` .

2. eID Applet

The eID Applet can be used within a web page as shown in the following example:

```
<script src="https://www.java.com/js/deployJava.js"></script>
<script>
   var attributes = {
      code :'be.fedict.eid.applet.Applet.class',
      archive :'eid-applet.jar',
      width :400,
      height :300
   };
   var parameters = {
      TargetPage :'identity-result.jsp',
      AppletService : 'applet-service',
      BackgroundColor : '#ffffff'
   };
   var version = '1.6';
   deployJava.runApplet(attributes, parameters, version);
</script>
```

The web application in which the eID Applet is embedded should use SSL for securing the communication between the web browser and the web application server. The eID Applet will not proceed when it detects a non SSL browser session.

The eID Applet will also not proceed when it detects that it has insufficient privileges to do so. This implies that the eID Applet JAR has to be signed and trusted by the citizen. The eID Applet that ships with an officially released eID Applet SDK has been signed by FedICT. In case of a security breach with the eID Applet, FedICT can revoke the corresponding code signing certificate to guarantee maximal safety of the citizen.

The available eID Applet parameters are summarized in *Table 1, "eID Applet Parameters"* .

Table 1. eID Applet Parameters

Parameter	Required	Description
`TargetPage`	required	Indicates the page to which the eID Applet navigates at the end of the requested eID operation. For example: `result.jsp`
`AppletService`	required	Points to the eID Applet Service server-side component that will handle the communication between the eID Applet and the (servlet) web application container. For example: `applet-service`
`BackgroundColor`	optional	The background color that is used by the eID Applet user interface. The default background color is white. For example: `#ffffff`
`ForegroundColor`	optional	The foreground color that is used by the eID Applet user interface. The default foreground color is black. For example: `#000000`
`Language`	optional	The language that is used by the eID Applet user interface for internationalization on the status messages. If it is not provided, the eID Applet defaults to the JRE runtime locale settings. For example: `nl`

Javascript

The eID Applet cannot be accessed from Javascript for XSS security reasons.

3. eID Applet Service

The eID Applet requires a server-side service component to communicate the identity or authentication data from the web browser to the server using a secure channel. We will call this component the eID Applet Service. The eID Applet SDK comes with eID Applet Service servlet components to ease integration of the eID Applet within a servlet container Java EE based web

application. The eID Applet Service components require at least a servlet version 2.4 container and a JRE version 1.5.

eID Applet Service implementations

For the moment we only support Java EE servlet containers out of the box. Depending on the success of the eID Applet SDK, we will also provide backends for other web application frameworks like PHP or ASP.NET.

3.1. eID Identification

By default the eID Applet Service will operate the eID Applet to make it perform an eID identification. This is also known as data capture.

The eID Applet Service Servlet can be configured via your `web.xml` web deployment descriptor as shown in the following example:

```
<servlet>
   <servlet-name>AppletServiceServlet</servlet-name>
   <servlet-class>be.fedict.eid.applet.service.AppletServiceServlet</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>AppletServiceServlet</servlet-name>
   <url-pattern>/applet-service</url-pattern>
</servlet-mapping>
```

eID Applet Service dependencies

The eID Applet Service, which can be found in the `eid-applet-service-x.x.x.jar` artifact, has the following dependencies:

- `eid-applet-shared-x.x.x.jar`
- `eid-applet-service-spi-x.x.x.jar`
- `commons-codec-1.3.jar`
- `commons-io-1.4.jar`
- `commons-logging-api-1.1.jar`
- `bcprov-jdk16-140.jar`

You should place these JAR files under the `META-INF/lib` directory of your web application.

After a successful identification took place the `AppletServiceServlet` eID Applet Service will push at least the `eid.identity` attribute, which holds the parsed identity fields, to the servlet container session. The `eid.identity` session attribute is of Java type `be.fedict.eid.applet.service.Identity` . More information on the exposed attributes can be found in the Javadoc API documentation of the eID Applet Service artifact.

eID Session Attributes

To ease integration of the eID Applet Service in web frameworks like JBoss Seam we've provided a top-level `eid` session attribute and getters on all exposed session attribute types. The top-level `eid` session attribute is of Java type `be.fedict.eid.applet.service.EIdData` . This means that the identity is available via both `eid.identity` session attribute and invocation of the `getIdentity()` method on the `eid` session attribute. This way we cover as much Java web frameworks as possible.

3.1.1. eID Address

During an eID identification operation the address on the eID card can be retrieved by setting the following `init-param` on the `AppletServiceServlet` :

```
<init-param>
   <param-name>IncludeAddress</param-name>
   <param-value>true</param-value>
</init-param>
```

The eID address will be available via the `eid.address` session attribute within the servlet container session context. The `eid.address` session attribute is of Java type `be.fedict.eid.applet.service.Address` .

3.1.2. eID Photo

During an eID identification operation the photo on the eID card can be retrieved by setting the following `init-param` on the `AppletServiceServlet` :

```
<init-param>
   <param-name>IncludePhoto</param-name>
   <param-value>true</param-value>
```

```
</init-param>
```

The photo will be available via the `eid.photo` session attribute within the servlet container session context. The eID photo is of Java type `byte[]` and in JPEG image format.

We provide a `PhotoServlet` to ease visualization of the eID photo within your web application. Configure the `PhotoServlet` as follows:

```
<servlet>
  <servlet-name>PhotoServlet</servlet-name>
  <servlet-class>be.fedict.eid.applet.service.PhotoServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>PhotoServlet</servlet-name>
  <url-pattern>/photo.jpg</url-pattern>
</servlet-mapping>
```

Then you can display the eID photo simply by putting next HTML tag in your web page:

```
<img src="photo.jpg" />
```

3.1.3. Identity Data Integrity

During an eID identification operation the eID Applet Service can perform integrity verification on the retrieved eID identity data. This feature prevents malicious parties to alter critical identity data.

To enable this functionality as part of an eID identification operation, you need to implement the `IdentityIntegrityService` interface. This interface can be found in the `eid-applet-service-spi` artifact. The corresponding service component (EJB3) session bean should be registered somewhere in JNDI. The JNDI location of this service component needs to be communicated to the eID Applet Service via the following `init-param` on the `AppletServiceServlet` :

```
<init-param>
  <param-name>IdentityIntegrityService</param-name>
  <param-value>your/location/in/jndi/IdentityIntegrityServiceBean</param-value>
</init-param>
```

Java EE Application Classpath

In an EJB Java EE application the `eid-applet-service-spi` artifact should be moved from your web application `WEB-INF/lib` WAR artifact to the EAR and be registered in `application.xml` as a Java module to avoid classpath issues in your application server.

Java EE 6 Web Profile support

To support the coming Java EE 6 Web Profile we already foresee the usage of two types of service component lookups. The first one is JNDI based. This type of service lookup allows you to utilize EJB3 session beans as service provider interface implementation. The second type is via simple Java class name references. This type of service lookup is meant for lightweight servlet container environment. For example the `SignatureService` interface implementing component can be referred to via both `SignatureService init-param` and via `SignatureServiceClass init-param` . The `SignatureService init-param` will trigger a JNDI lookup of the signature service. The `SignatureServiceClass init-param` will trigger a class instantiation using the default constructor of the given class.

Authenticated Identification

If the eID identification is preceded with an eID authentication then the eID Applet Service will be able to link the national registration number of the eID authentication with the one found in the eID identity file during identity integrity verification. This makes for a bullet-proof eID identification.

3.2. eID Authentication

The eID Applet can be used to authenticate an end user via the eID card. eID based authentication is much safer than a simple password based authentication scheme since the eID card makes a two-factor authentication possible.

To perform an eID authentication, you need to implement the `AuthenticationService` interface. This interface can be found in the `eid-applet-service-spi` artifact. This service component (EJB3) session bean should be registered somewhere in JNDI. The JNDI location of this service component needs to be set via the following `init-param` on the `AppletServiceServlet` :

```
<init-param>
  <param-name>AuthenticationService</param-name>
```

```
    <param-value>your/location/in/jndi/AuthenticationServiceBean</param-value>
</init-param>
```

After a successful authentication the `eid.identifier` session attribute will contain a unique identifier (the national registration number) for the user. The `eid.identifier` session attribute is of Java type `java.lang.String` . To respect the citizen's privacy, the national registration number should not be abuse for linking identity data. Profiling based on eID data linking is forbidden by law.

Mac OS X

Because Mac OS X does not support the Java 6 runtime we made the eID Applet to also run on a Java 5 runtime for the basic eID authentication (and eID signature) functionality.

Sun JRE

Because the eID Applet is using the `SunPKCS11` security provider we need the Sun JRE as browser applet runtime for eID authentication (and eID signature) via the eID Applet.

eID Middleware

The eID Applet is using the PKCS#11 library for eID authentication (and eID signatures). This requires that the eID Middleware has been installed on the client system.

If no PKCS#11 library has been found and the applet runtime is Java 6, then the eID Applet will fallback to the Java 6 Smart Card I/O API to generate the authentication (or in case of eID signature the non-repudiation) digital signature via the direct PC/SC smart card interface.

By default the eID Applet will sign a challenge using the authentication certificate of the citizen's eID card. This challenge is send over SSL by the eID Applet Service. To prevent a certain type of man-in-the-middle attack we can make the eID Applet to sign the concatenation of the site hostname and the server-side challenge. This feature can be enabled by setting the following `init-param` on the `AppletServiceServlet` :

```
<init-param>
    <param-name>Hostname</param-name>
    <param-value>www.PutYourSiteHostnameHere.be</param-value>
```

```
</init-param>
```

Hostname verification

It is strongly advised to enable this hostname verification feature to maximize security.

To prevent DNS attacks one can even make the eID Applet sign the IP address of the server. This feature can be enabled by setting the following `init-param` on the `AppletServiceServlet` :

```
<init-param>
  <param-name>InetAddress</param-name>
  <param-value>1.2.3.4</param-value>
</init-param>
```

If you enable both `Hostname` and `InetAddress` features at the same time, the eID Applet will be signing `hostname||IP address||challenge` . The hostname and IP address are the same as seen by the web browser.

3.3. eID Signatures

The eID Applet can also be used to create digital signatures using the non-repudiation signing eID certificate. The supported signature algorithms are `SHA1-RSA-PKCS1` , `SHA224-RSA-PKCS1` , `SHA256-RSA-PKCS1` , `SHA384-RSA-PKCS1` , `SHA512-RSA-PKCS1` , `RIPEMD128-RSA-PKCS1` , `RIPEMD160-RSA-PKCS1` , and `RIPEMD256-RSA-PKCS1` .

Legally Binding eID Digital Signatures

Please be aware that the eID digital signatures are legally binding by law. Don't make the citizen sign electronic documents unless it is absolutely necessary from a legal point of view for the correct functioning of your business work flow.

To use this functionality you need to implement the `SignatureService` interface. This interface can be found in the `eid-applet-service-spi` artifact. This service component (EJB3) session bean should be registered somewhere in JNDI. The JNDI location of this service component needs to be set via the following `init-param` on the `AppletServiceServlet` :

```
<init-param>
  <param-name>SignatureService</param-name>
  <param-value>your/location/in/jndi/SignatureServiceBean</param-value>
```

```
</init-param>
```

The eID Applet Service can be configured to perform two basic types of digital signatures:

- The digest value to be signed originates solely from the `DigitalSignature` implementing service component.
- The eID Applet first send over a set of digest values calculated from local files. These files are selected by the citizen via an eID Applet file user interface. Out of this set of digest values the `DigitalSignature` implementing service component then calculates a super digest value. This digest value is signed using the eID Applet.

  The supported file digest algorithms are `SHA-1` , `SHA-256` , `SHA-384` , and `SHA-512` .

  This type of digital signature operation can be used to construct for example XML Signatures, XAdES Signatures or PDF Signatures.

PKI Validation

The eID Applet Service does not perform any PKI validation. So the signature service component, authentication service component and the identity integrity component need to implement PKI validation of the citizen certificates itself. PKI validation is out of scope of the provided eID Applet Service.

3.4. eID Administration

The eID Applet allows for some administrative eID tasks like changing the PIN and unblocking the PIN. This feature has been implemented to break the hard dependency on the eID Middleware.

The eID PIN change administrative task can be activated by setting the following `init-param` on the `AppletServiceServlet` :

```
<init-param>
  <param-name>ChangePin</param-name>
  <param-value>true</param-value>
</init-param>
```

The eID unblock PIN administrative task can be activated by setting the following `init-param` on the `AppletServiceServlet` :

```
<init-param>
  <param-name>UnblockPin</param-name>
  <param-value>true</param-value>
```

```
</init-param>
```

3.5. Generic eID Applet Service settings

The settings listed in this section apply to eID identification operations, eID authentication operations, eID signature operations, and eID administration operations.

3.5.1. Secure Client Environment

The eID Applet offers functionality to check whether the client environment is secure enough given the application requirements. In case the eID Applet Service detects an insecure client environment the eID Applet can:

- show an error message and abort the requested eID operation.
- show a warning message and ask the citizen whether he/she wants to continue or not.

To activate this functionality you need to implement the `SecureClientEnviromentService` interface. This interface can be found in the `eid-applet-service-spi` artifact. This service component (EJB3) session bean should be registered somewhere in JNDI. The JNDI location of this service component needs to be set via the following `init-param` on the `AppletServiceServlet` :

```
<init-param>
  <param-name>SecureClientEnvironmentService</param-name>
  <param-value>your/location/in/jndi/SecureClientEnvironmentServiceBean</param-value>
</init-param>
```

Additional client environment information can be pushed to the eID Applet Service by adding the following eID Applet parameters within your web page eID Applet configuration:

```
NavigatorUserAgent : navigator.userAgent,
NavigatorAppName : navigator.appName,
NavigatorAppVersion : navigator.appVersion
```

Java 6

This eID Applet feature requires a Java 6 browser runtime or an installed eID Middleware PKCS#11 library.

3.5.2. eID Card Removal

The eID Applet can ask the citizen for eID card removal after performing the selected eID operation (either eID identification or eID authentication). This option can be used to limit the window of opportunity for malware to abuse the eID card.

The eID card removal can be activated by setting the following `init-param` on the `AppletServiceServlet` :

```
<init-param>
  <param-name>RemoveCard</param-name>
  <param-value>true</param-value>
</init-param>
```

3.5.3. eID Card Logoff

After an eID authentication or eID signature the eID card will re-use the PIN authorization for future eID authentication operations. This feature was originally implemented on the eID JavaCard Applet (which is located inside the eID chip) to allow for mutual authenticated SSL without the need to re-enter the PIN on each SSL session renewal. Although this makes sense in the context of SSL, it actually makes for a serious eID security weakness: SSO should be handled at the IdP level, not at the card level. Only an IdP can have notion of trust domains between different web applications. Luckily the eID card foresees in an eID card logoff. This eID logoff feature can be enabled during both eID authentication or eID signature operations by setting the following `init-param` on the `AppletServiceServlet` :

```
<init-param>
  <param-name>Logoff</param-name>
  <param-value>true</param-value>
</init-param>
```

This feature requires a Java 6 browser runtime as it is using the Smart Card I/O API. If no Java 6 runtime is available this feature will default to an eID card removal as this yields the same result.

Enable eID card logoff

It is strongly advised to enable the eID card logoff feature to prevent abuse of the authentication functionality of the eID card.

4. eID Applet Protocol

In this section we will elaborate on the eID Applet protocol used in the communication between the eID Applet and the eID Applet Service. If you use the eID Applet Service that comes with the eID Applet SDK you actually don't need to know the details of the eID Applet protocol. However, this information can be useful for application developers that use other web frameworks than a Java EE servlet container based framework.

The eID Applet Protocol is based on the HTTP protocol using the POST method. Parameters are passed as HTTP headers and for binary data the HTTP body is used. The messages should be transported over a secure SSL connection.

4.1. eID Applet Protocol Messages

The following documentation has been generated automatically.

4.1.1. HelloMessage

This message starts a communication session between eID Applet and eID Applet Service. It sets the protocol state to: INIT

Table 2. HelloMessage HTTP headers

Header name	Required	Value
`X-AppletProtocol-Type`	true	`HelloMessage`
`X-AppletProtocol-Version`	true	`1`

Allowed eID Applet Service response messages are: *Section 4.1.8, "IdentificationRequestMessage" Section 4.1.9, "CheckClientMessage" Section 4.1.11, "AuthenticationRequestMessage" Section 4.1.12, "AdministrationMessage" Section 4.1.13, "SignRequestMessage" Section 4.1.14, "FilesDigestRequestMessage"*

4.1.2. ClientEnvironmentMessage

This message is only accepted if the eID Applet Service protocol state is: ENV_CHECK

Table 3. ClientEnvironmentMessage HTTP headers

Header name	Required	Value
`X-AppletProtocol-Type`	true	`ClientEnvironmentMessage`
`X-AppletProtocol-JavaVersion`	true	Some String value.
`X-AppletProtocol-JavaVendor`	true	Some String value.
`X-AppletProtocol-OSName`	true	Some String value.
`X-AppletProtocol-OSArch`	true	Some String value.

Header name	Required	Value
`X-AppletProtocol-OSVersion`	true	Some String value.
`X-AppletProtocol-NavigatorUserAgent`	false	Some String value.
`X-AppletProtocol-NavigatorAppName`	false	Some String value.
`X-AppletProtocol-NavigatorAppVersion`	false	Some String value.
`X-AppletProtocol-Version`	true	`1`

HTTP body should contain the data.

Body content: A list of strings containing the names of the smart card readers.

Allowed eID Applet Service response messages are: *Section 4.1.8, "IdentificationRequestMessage" Section 4.1.10, "InsecureClientMessage" Section 4.1.11, "AuthenticationRequestMessage" Section 4.1.12, "AdministrationMessage" Section 4.1.13, "SignRequestMessage" Section 4.1.14, "FilesDigestRequestMessage"*

4.1.3. AuthenticationDataMessage

This message is only accepted if the eID Applet Service protocol state is: AUTHENTICATE

Table 4. AuthenticationDataMessage HTTP headers

Header name	Required	Value
`X-AppletProtocol-Type`	true	`AuthenticationDataMessage`
`X-AppletProtocol-SignatureValueSize`	true	Some Integer value.
`X-AppletProtocol-Version`	true	`1`

HTTP body should contain the data.

Body content: Contains concatenation of signature value and authn cert chain.

Allowed eID Applet Service response messages are: *Section 4.1.15, "FinishedMessage"*

4.1.4. SignatureDataMessage

This message is only accepted if the eID Applet Service protocol state is: SIGN

Table 5. SignatureDataMessage HTTP headers

Header name	Required	Value
`X-AppletProtocol-Type`	true	`SignatureDataMessage`

Header name	Required	Value
`X-AppletProtocol-SignatureValueSize`	true	Some Integer value.
`X-AppletProtocol-Version`	true	`1`

HTTP body should contain the data.

Body content: Contains concatenation of signature value and sign cert chain.

Allowed eID Applet Service response messages are: *Section 4.1.15, "FinishedMessage"*

4.1.5. FileDigestsDataMessage

This message is only accepted if the eID Applet Service protocol state is: DIGEST

Table 6. FileDigestsDataMessage HTTP headers

Header name	Required	Value
`X-AppletProtocol-Type`	true	`FileDigestsDataMessage`
`X-AppletProtocol-Version`	true	`1`

HTTP body should contain the data.

Body content: List of string triples containing (digest algo, digest value in hex, description)

Allowed eID Applet Service response messages are: *Section 4.1.13, "SignRequestMessage"*

4.1.6. ContinueInsecureMessage

This message is only accepted if the eID Applet Service protocol state is: INSECURE

Table 7. ContinueInsecureMessage HTTP headers

Header name	Required	Value
`X-AppletProtocol-Type`	true	`ContinueInsecureMessage`
`X-AppletProtocol-Version`	true	`1`

Allowed eID Applet Service response messages are: *Section 4.1.8, "IdentificationRequestMessage" Section 4.1.11, "AuthenticationRequestMessage" Section 4.1.12, "AdministrationMessage" Section 4.1.13, "SignRequestMessage" Section 4.1.14, "FilesDigestRequestMessage"*

4.1.7. IdentityDataMessage

This message is only accepted if the eID Applet Service protocol state is: IDENTIFY

Table 8. IdentityDataMessage HTTP headers

Header name	Required	Value
`X-AppletProtocol-Type`	true	`IdentityDataMessage`
`X-AppletProtocol-IdentityFileSize`	true	Some Integer value.
`X-AppletProtocol-AddressFileSize`	false	Some Integer value.
`X-AppletProtocol-PhotoFileSize`	false	Some Integer value.
`X-AppletProtocol-IdentitySignatureFileSize`	false	Some Integer value.
`X-AppletProtocol-AddressSignatureFileSize`	false	Some Integer value.
`X-AppletProtocol-RrnCertFileSize`	false	Some Integer value.
`X-AppletProtocol-Version`	true	`1`

HTTP body should contain the data.

Body content: Concatenation of identity file, optional address file, optional photo file, optional identity signature file, optional address signature file, and optional national registry certificate.

Allowed eID Applet Service response messages are: *Section 4.1.15, "FinishedMessage"*

4.1.8. IdentificationRequestMessage

Table 9. IdentificationRequestMessage HTTP headers

Header name	Required	Value
`X-AppletProtocol-Type`	true	`IdentificationRequestMessage`
`X-AppletProtocol-IncludeAddress`	false	Some boolean value.
`X-AppletProtocol-IncludePhoto`	false	Some boolean value.
`X-AppletProtocol-IncludeIntegrityData`	false	Some boolean value.
`X-AppletProtocol-RemoveCard`	false	Some boolean value.
`X-AppletProtocol-Version`	true	`1`

This message will perform an eID Applet protocol state transition to: IDENTIFY

4.1.9. CheckClientMessage

Table 10. CheckClientMessage HTTP headers

Header name	Required	Value
`X-AppletProtocol-Type`	true	`CheckClientMessage`
`X-AppletProtocol-Version`	true	`1`

This message will perform an eID Applet protocol state transition to: ENV_CHECK

4.1.10. InsecureClientMessage

Table 11. InsecureClientMessage HTTP headers

Header name	Required	Value
`X-AppletProtocol-Type`	true	`InsecureClientMessage`
`X-AppletProtocol-WarnOnly`	false	Some boolean value.
`X-AppletProtocol-Version`	true	`1`

This message will perform an eID Applet protocol state transition to: INSECURE

4.1.11. AuthenticationRequestMessage

Table 12. AuthenticationRequestMessage HTTP headers

Header name	Required	Value
`X-AppletProtocol-Type`	true	`AuthenticationRequestMessage`
`X-AppletProtocol-RemoveCard`	false	Some boolean value.
`X-AppletProtocol-IncludeHostname`	false	Some boolean value.
`X-AppletProtocol-IncludeInetAddress`	false	Some boolean value.
`X-AppletProtocol-Logoff`	false	Some boolean value.
`X-AppletProtocol-Version`	true	`1`

HTTP body should contain the data.

Body content: The challenge to be signed using the authentication certificate. If IncludeHostname is set, then prefix the challenge with the server hostname before signing.

This message will perform an eID Applet protocol state transition to: AUTHENTICATE

4.1.12. AdministrationMessage

This message stops a communication session between eID Applet and the eID Applet Service.

Table 13. AdministrationMessage HTTP headers

Header name	Required	Value
`X-AppletProtocol-Type`	true	`AdministrationMessage`
`X-AppletProtocol-ChangePin`	false	Some boolean value.
`X-AppletProtocol-UnblockPin`	false	Some boolean value.
`X-AppletProtocol-RemoveCard`	false	Some boolean value.
`X-AppletProtocol-Version`	true	`1`

4.1.13. SignRequestMessage

Table 14. SignRequestMessage HTTP headers

Header name	Required	Value
`X-AppletProtocol-Type`	true	`SignRequestMessage`
`X-AppletProtocol-DigestAlgo`	true	Some String value.
`X-AppletProtocol-Description`	false	Some String value.
`X-AppletProtocol-RemoveCard`	false	Some boolean value.
`X-AppletProtocol-Logoff`	false	Some boolean value.
`X-AppletProtocol-Version`	true	`1`

HTTP body should contain the data.

Body content: The digest value to be signed using the non-repudiation certificate

This message will perform an eID Applet protocol state transition to: SIGN

4.1.14. FilesDigestRequestMessage

Table 15. FilesDigestRequestMessage HTTP headers

Header name	Required	Value
`X-AppletProtocol-Type`	true	`FilesDigestRequestMessage`
`X-AppletProtocol-DigestAlgo`	true	Some String value.
`X-AppletProtocol-Version`	true	`1`

This message will perform an eID Applet protocol state transition to: DIGEST

4.1.15. FinishedMessage

This message stops a communication session between eID Applet and the eID Applet Service.

Table 16. FinishedMessage HTTP headers

Header name	Required	Value
`X-AppletProtocol-Type`	true	`FinishedMessage`
`X-AppletProtocol-Version`	true	`1`

A. eID Applet Developer's Guide License

This document has been released under the Creative Commons license.

You are free to Share — to copy, distribute and transmit the work.

You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

You may not use this work for commercial purposes.

You may not alter, transform, or build upon this work.

B. eID Applet License

The eID Applet source code has been released under the GNU LGPL version 3.0.

C. Revision history

Table C.1. Revision history

Date	Author	Description
26 Jan 2009	Frank Cornelis	Initial version.