Cl@ve v2.8.4_07 - Manual de Integración para Proveedores de Servicios (Java, PHP y ASP.NET)

Plataforma Cl@ve 2

Revisión 04

Versión 20/06/2024

CONTROL DE MODIFICACIONES

Documento nº:	Clave v2.8.4_07 - Manual Integracion SP r04
Revisión:	2.8.4_07 r04
Fecha:	20/06/2024
Descripción:	Se incorporan los apartados 6.4 eIDAS y el Identity Matching y 8.3.1.4 Atributo NationalPersonIdentifier en referencia al servicio de identificación de ciudadanos eIDAS de la Agencia Tributaria (AEAT) que permite obtener el identificador español para un ciudadano europeo. Se corrige un duplicado en la definición del atributo PersonIdentifier del apartado 8.2.1 Se modifica el formato del texto asociado al atributo RelayState en el apartado 8.3.2 Se corrige una errata en el texto referente al atributo IDPResponses en el apartado 8.3.2 Se corrige una errata en el texto referente al atributo SelectedIdP en el apartado 8.3.2

Rev.	2.4
Fecha	21/10/2020
Descripción	Documento inicial.

Rev.	2.5
Fecha	22/03/2021
Descripción	Calculo *ForceAuthn*.

Rev.	2.6
Fecha	08/09/2021
Descripción	Actualización de la librería 'es\gob\sgad\clave-j20068.eidas-saml-engine\2.7\clave-j20068.eidas-saml-engine-2.7.jar' para resolver un error de validación de certificado del usuario que se daba solo en algunas ocasiones. Se modifica el KIT de INTEGRACIÓN de CLAVE JAVA 8 para que consuma esta nueva librería de CLAVE2.

Rev.	2.7
Fecha	20/09/2021
Descripción	Documentar listado de errores que puede generar la pasarela Clave.

Rev.	2.8
Fecha	28/09/2021
Descripción	Documentar atributo *DateOfBirth* y *NationalPersonID*.

Rev.	2.9
Fecha	28/09/2021

Descripción	Añadir apartado “7.1.14.Actualización automática de los certificados de Clave en los SPs” de tecnologia Java.
Rev.	2.10
Fecha	28/09/2021
Descripción	Añadir apartado “7.2.8.Actualización automática de certificados” de tecnologia AP.NET. Cambiar icono del ministerio.
Rev.	2.11
Fecha	28/09/2021
Descripción	Añadir apartado “7.3.10.Actualización automática de certificados” de tecnologia PHP.
Rev.	2.12
Fecha	26/10/2021
Descripción	Se actualiza el apartado ‘Dependencias a importar’ del Pack de integración de JAVA.
Rev.	2.13
Fecha	26/04/2022
Descripción	En la pantalla del SP2 del Kit de Integración de JAVA, se añade el nuevo checkbox ‘IDP Móvil’. Si se activa, indica que que el IDP Móvil tiene que deshabilitarse en la pantalla de Pasarela, en caso de que este estuviera configurado para el SP que hace la petición. Se amplía el apartado 7.1.8 para indicar al integrador como añadir en la petición SAML los atributos ‘NationalPersonIdentifier’, ‘AuthnNIFLM’ y ‘CLVMOVILIdP’.
Rev.	2.14
Fecha	01/06/2022
Descripción	Se añade el checkbox 'IDP Móvil' en la pantalla del SP DEMO del Kit Java .NET Si se selecciona, se esconde el IDP Móvil de la pantalla de PASARELA, en caso de que el SP lo tuviera configurado -> "FUN-11.Implementación IDP Cl@ve Autenticador", para poder deseleccionar o seleccionar los nuevos IDP múltiples (Móvil) Se añade el apartado ‘NOTAS IMPORTANTES PARA CONECTAR EL KIT CONTRA CLAVE2 PASARELA DEL ENTORNO DE SERVICIOS ESTABLES’ para todos los Kits.
Rev.	2.15
Fecha	01/08/2022
Descripción	Se ha actualizado el Kit PHP CLAVE2 de la versión 1.9 a la versión SimpleSaml 1.19.5 - Con 10 versiones entre medias, los cambios en la librería son bastante significantes, sobre todo a nivel de estructuración y contenidos. Las clases principales siguen encontrándose en las mismas ubicaciones. Principalmente se han adaptado las nuevas clases para funcionar en concordancia con lo que hacían las de la versión anterior. La versión anterior tenía gran parte de las clases propias de la librería adaptadas al funcionamiento de la aplicación, con nuevas funciones o modificaciones de estas, y se ha tenido que llevar esas mismas modificaciones a las clases de la nueva versión. Las clases nombradas como EidasXXXXX.php que extienden a las clases de la librería se han incluido para el correcto funcionamiento.

En la pantalla del SP2 del Kit de Integración de JAVA, se añade el nuevo checkbox 'IDP Móvil'. Si se activa, indica que que el IDP Móvil tiene que deshabilitarse en la pantalla de Pasarela, en caso de que este estuviera configurado para el SP que hace la petición. Al igual que el resto de checkboxs, su configuración se encuentra en las rutas SimpleSamlSP\www\SP\selectAttributes.php y SimpleSamlSP\lib\SAML2\Constants.php.

'CLVMOVILIdP.name' => 'CLVMOVILIdP',

'CLVMOVILIdP.uri' => 'http://es.minhafp.clave/CLVMOVILIdP',

'CLVMOVILIdP.nameFormat' => 'urn:oasis:names:tc:SAML:2.0:attrname-format:uri',

'CLVMOVILIdP.value' => NULL

Rev.	2.16
Fecha	10/01/2023
Descripción	Actualización del apartado 7.2.1 para registrar la posibilidad de incluir manualmente el certificado que viene en las repuestas SAML desde Cl@ve-SE en el almacén de “Personas de Confianza” del sistema.

Rev.	2.17
Fecha	15/02/2023
Descripción	Actualización del apartado 7.1.6 para eliminar referencias a versiones anteriores de librerías de Cl@v3. Actualización del apartado 7.1.7 para indicar las dependencias correctas a importar desde el proyecto.

Rev.	2.18
Fecha	28/03/2023
Descripción	Se ha actualizado el Kit de PHP v2.5.1 de PHP v7.4.29 a PHP v8.2.0 Se ha comprobado que para conectar el Kit contra CLAVE SE, hay que utilizar el SP: 21114293V_E04975701 Los ficheros de configuración y certificados están en las siguientes carpetas: paquete_sp_php\SimpleSamlSP\cert paquete_sp_php\SimpleSamlSP\config Ahora ya se puede ejecutar dicho Kit con un Apache v8 y un PHP v8.

Rev.	2.19
Fecha	03/03/2023
Descripción	Se incorpora el apartado “3.2.1.Detalle de la cabecera enviada des del SP2 hacia CLAVE PASARELA (PROXY2) en el entorno de PRE WILDFLY”.

Rev.	2.8.4_004
Fecha	04/08/2023
Descripción	Se actualiza el apartado “6.3.1 Atributos recibidos en respuestas correctas de autenticación” para incluir el IdP PIN24H_MOVIL y añadir una nota al pie para esclarecer el valor del atributo de respuestas SelectedIdp. Se modifica la versión del presente documento a ‘v2.8.4_004’, para hacer referencia a la v2.8.4_004 desplegada en el entorno de producción.

Rev.	2.8.4_004 r01
Fecha	10/08/2023
Descripción	Se ha actualizado la estructura del contenido. Se asocia el documento a la versión de Cl@ve concreta (2.8.4_04) y se asigna numeración para las revisiones de aquél (actualmente la 01). Se actualizan las versiones de los kits de integración (apartado 4.1). Se añade al apartado 6.3.1 los atributos “IDP_PROVIDER” y “CLV_LANGUAGE”. Se elimina referencia al atributo "AuthnNIFLM" en el apartado 9.1.9. Se ha actualizado la descripción contenida en el apartado 9.1.15. Se elimina del apartado 9.2.7 referencia al atributo "AuthnNIFLM". Se lleva el glosario de términos al apartado 1 y se completa. Se llevan los enlaces de interés al apartado 2.

Rev.	2.8.4_005 r01
Fecha	22/08/2023
Descripción	Se incluye el apartado 12.1.9 con los algoritmos de firma soportados por la plataforma. Se incluye el apartado 12.1.10 con los algoritmos de resumen soportados por la plataforma. Se ha incluido en el apartado 5.2 mención a la política de seguridad "Referrer-Policy".

Rev.	2.8.4_005 r02
Fecha	29/08/2023
Descripción	Se actualiza en el apartado 8.3.1 la descripción del atributo "SelectedIdP". Se actualiza y lleva al apartado 8.2.5 la descripción del atributo "IDP_PROVIDER". Se actualiza y lleva al apartado 8.2.6 la descripción del atributo "CLV_LANGUAGE".

Rev.	2.8.4_005 r03
Fecha	16/10/2023
Descripción	Se actualiza en el apartado 8.3.1 los posibles valores de los atributos "SelectedIdP" y “RegisterType” así como una nota indicando que hay determinados IdP que pueden no devolver todos los atributos obligatorios. Se actualiza en el apartado 8.2.5 el valor asociado al atributo "IDP_PROVIDER".

Rev.	2.8.4_005 r04
Fecha	22/11/2023
Descripción	Se eliminan los apartados 8.2.5 y 8.2.6 ya que hacían referencia a 2 atributos que se envían desde Pasarela a los IdPs, no de los SPs a Pasarela.

Rev.	2.8.4_005 r05
Fecha	29/12/2023
Descripción	En el apartado 'Calidad de la Credencial (LoA) y Atributos Personales', se añade información relativa a los niveles de LOA que maneja el nuevo IdP Cl@ve MOVIL.

Rev.	2.8.4_005 r06
Fecha	24/01/2024
Descripción	Incorporación de los valores LoA devueltos por los IDPs

Rev.	2.8.4_07 r00
Fecha	04/03/2024
Descripción	Se incluye información sobre cómo obtener los datos de la organización en caso de autenticarse con certificados de representante de persona jurídica (clasificación 11 y 12)

Rev.	2.8.4_07 r01
Fecha	07/03/2024
Descripción	Se actualiza el punto “Descripción de los Ficheros de Configuración de un SP” incluyendo información de la etiqueta “signature.algorithm” para indicar el algoritmo de firma que utilizar el certificado con el que firmar los mensajes SAML para el caso de los kits de integración Java.

Rev.	2.8.4_07 r02
Fecha	11/03/2024
Descripción	Se actualiza el punto “Descripción de los Ficheros de Configuración de un SP” añadiendo información concreta para la configuración de certificados de curva elíptica.

Rev.	2.8.4_07 r03
Fecha	22/05/2024
Descripción	Se actualiza el apartado “9.1.9 Generación de un Ticket SAML2” para aclarar la necesidad de hacer uso del proveedor criptográfico BouncyCastle. Se actualiza el apartado “9.1.4 Descripción de los ficheros de configuración de un SP” para redefinir el uso de las etiquetas asociadas a firma y encriptación, así como el uso de certificados de curva elíptica. Se actualiza el apartado “9.1.6 Certificados” para ampliar la descripción de “Requisitos en el apartado de certificados” respecto a los certificados de curva elíptica. Se añaden dos nuevos anexos “12.1.10 Algoritmos de Cifrado de Datos Utilizados por Cl@ve” y “12.1.11 Algoritmos de Cifrado de Clave Utilizados por Cl@ve”.

Índice

1 Glosario de Términos

Assertion	Conjunto de datos empaquetados en un mensaje de SAML, se dividen en tres tipos de aserciones, de autenticación, de autorización o atributos declarados.
Http Redirect	Es un estado que emite el servidor (30X) para redirigir al navegador del cliente a una URL.
JKS	Java Key Store, es un tipo de fichero orientado a almacenar certificados y claves privadas para servidores de aplicaciones. Se suelen dividir en dos, keystore o "Privare Key entries" orientado claves privadas acompañadas por el certificado asociado a la clave pública, y truststore o "Trusted Certificate Entries" orientado almacenar las claves públicas de otras entidades.
Keytool	Es una utilidad de java para la gestión y manejo de certificados (normalmente se incluye junto con la JDK).
Maven	Herramienta de gestión de proyectos de software, que entre otras funcionalidades nos aporta las dependencias y gestión de librerías necesarias.
POM	Es un fichero XML que define la información acerca del proyecto de Maven, fuentes, dependencias, test.
SAML	Security Assertion Markup Language
ASP	Active Server Pages
IdP	Identity Provider
OASIS	Organization for the Advancement of Structured Information Standards
SE	Servicios Estables
PRO	Producción
eIDAS	electronic IDentification, Authentication and trust Services
PHP	Hypertext Pre-Processor
XML	Extensible Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
SSO	Single Sign-On
SP	Service Provider
SSL	Secure Sockets Layer
PRE	Pre-Producción
TLS	Transport Layer Security
LoA	Level of Assurance
NIF	Número de Identificación Fiscal
Base64	Sistema de numeración posicional que usa 64 como base
DNI	Documento Nacional de Identidad
CSV	Código Seguro de Verificación
AAPP	Administraciones Públicas

PKCS	Public-Key Cryptography Standards
IDE	Integrated Development Environment
IIS	Internet Information Services
CA	Certification Authority
SHA	Secure Hash Algorithm
MD5	Message Digest Algorithm 5
DES	Data Encryption Standard
FIPS	Federal Information Processing Standards
AES	Advanced Encryption Standard
IV	Initialization Vector
GCM	Galois/Counter Mode
PKI	Public Key Infrastructure
IETF	Internet Engineering Task Force
RFC	Request for Comments
DH	Diffie–Hellman
SMTP	Simple Mail transfer Protocol
IMAP	Internet Message Access Protocol
ECC	Elliptic Curve Cryptography

2 Enlaces de Interés

[1] SAML, [Website], https://en.wikipedia.org/wiki/Security_Assertion_Markup_Language

[2] OASIS, [Website], https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[3] XML, [Website], https://es.wikipedia.org/wiki/Extensible_Markup_Language

3 Introducción

El presente documento define e ilustra el proceso de **integración con la plataforma de autenticación federada Cl@ve** por parte de proveedores de servicios de administración electrónica, mediante el intercambio de mensajes SAML v2.0 (formato eIDAS).

La implementación de la plataforma Cl@ve se ha basado en las especificaciones técnicas eIDAS, estándar europeo que organiza y asegura la delegación de las funciones de autenticación de un proveedor de servicios, haciendo uso de auditados procesos criptográficos que incluyen firma electrónica y cifrado.

Estas especificaciones técnicas, que son un perfil basado en el estándar SAML 2.0, utilizan el concepto de identidad federada, favoreciendo que las aplicaciones no necesiten almacenar los datos de los usuarios y

soliciten los datos de autenticación y datos referentes a las identidades a una entidad que alberga y custodia los mismos, proporcionando un sistema de SSO.

Para la comunicación entre proveedores de servicios y proveedores de identidades, se incluye un intermediador central que simplifica el proceso de confianza electrónica y audita el proceso.

4 Alcance

El presente documento pretende en un primer lugar definir de una forma funcional, de manera que se comprenda de forma concisa, el nuevo modelo de Cl@ve, indicando cada uno de los componentes que forman esta arquitectura. Para posteriormente, describir de forma minuciosa las llamadas y flujos que conlleva la integración, mediante el intercambio de mensajes SAML v2.0 (formato eIDAS).

Se hace énfasis en los paquetes de integración basados en Java para poder integrar por parte del personal técnico.

4.1 Última Versión de los Integradores

Tecnología	Versión Kit
Java 7	2.1.1
Java 8	2.2.3
C# y ASP.NET – Framework 4.6.1	(Discontinued)
C# y ASP.NET – Framework 4.7.2	2.5.3
PHP 8.2.0	2.6.0

5 Modelo Funcional

Cuando un ciudadano desea acceder a un determinado servicio (por ejemplo, a la oficina electrónica de una administración que hace uso de los servicios que proporciona Clave) lo primero que hace es conectarse a la aplicación del proveedor de servicios (SP), el cual requiere que se autentique (paso 1, ver figura a continuación).

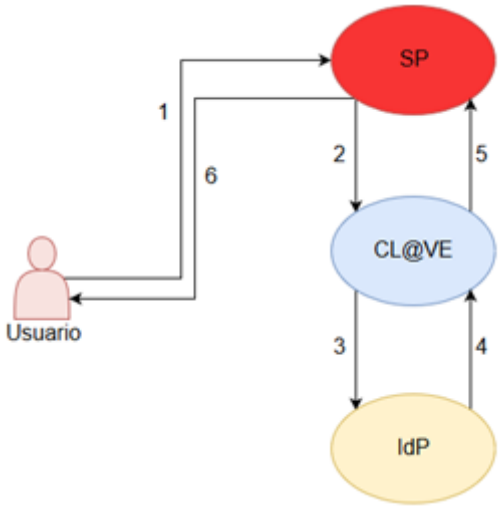


Ilustración 1: Modelo Funcional de Comunicación

La necesidad de autenticación del SP se delega en un sistema externo (nodo Cl@ve), por lo que es necesario enviar al usuario hacia dicho sistema externo, lo cual implica cumplir con dos requisitos básicos.

- Que el SP esté dado de alta en el sistema externo. Esto se consigue mediante un trámite de alta previo que realiza el SP con el órgano gestor del nodo Clave. La validación que subyace para asegurar que el SP está dado de alta consiste en una validación mutua a través de tickets SAML que se intercambian entre cada una de las partes. Estos tickets SAML contienen un XML firmado donde se especifican las credenciales públicas del SP en cuestión. Estas credenciales deben estar acreditadas por un tercero de confianza.
- Que el usuario porte un ticket SAML2 firmado por el SP donde se solicita una autenticación en el nodo Clave.

Se firma un ticket SAML2 para que el usuario, mediante una redirección, de tipo POST o de tipo Redirect, vaya al Intermediador. El usuario porta en una cabecera HTML el ticket SAML2 para que, al entrar en el servidor, éste lo reciba como parámetro “SAMLRequest” (paso 2).

Seguidamente, el intermediador realiza las tareas de validación del ticket SAML2 y muestra al usuario una página web donde se pide que se escoja el IdP en el que se quiere realizar la autenticación.

Después, en función del IdP seleccionado por el propio usuario, se va al IdP en cuestión (paso 3) donde presentará su credencial personal para ser identificado.

Ya con los datos del usuario, se sigue el camino contrario: el IdP responde al intermediador (paso 4), que a su vez responde al proveedor de servicios (paso 5), con un ticket que contiene una firma electrónica y un contenido cifrado. Dentro de este contenido cifrado se encuentran los datos del ciudadano que se han recuperado en el IdP.

Esta funcionalidad aporta un nivel de seguridad y confianza adicional en el tránsito de la información del usuario, ya que los SP’s no se comunican directamente con los IdP’s, sino que siempre existen unos elementos

intermedios que protegen y aseguran los datos solicitados y entregados. El proveedor de servicio utiliza este ticket como fuente de información veraz y procede a iniciar la sesión del usuario en sus sistemas (Paso 6).

En resumen, se parte de una solicitud de autenticación firmada por el SP, la plataforma Cl@ve procesa la credencial presentada y prepara una respuesta firmada que contiene los datos del usuario, sin necesidad de que el proveedor de servicios tenga que implementar lógica para la introducción de credenciales de usuario ni necesite conexión con fuentes de datos de identidad.

5.1 Flujo de Negocio SAML2

El protocolo SAML2 (Security Assertion Markup Language 2.0) es un estándar propuesto por OASIS para el intercambio de datos de autenticación y autorización (basado en un estándar de XML) entre entidades conocidas como proveedores de identidad (IdP) y proveedores de servicios (SP).

Dada las características de seguridad que requiere la protección de la identidad digital se realizan dos procesos sobre los mensajes SAML2:

- Se firman digitalmente los mensajes en el proceso de intercambio de información.
- Los datos sensibles del usuario quedan cifrados en los mensajes SAML2 empleado la clave pública presente en la firma electrónica del ticket SAML de origen, para que sean legible sólo por el poseedor de la clave privada, es decir, el destino al que van dirigidas.

La firma electrónica y el cifrado implican:

- La integridad del mensaje: No hay alteraciones en su contenido.
- La autenticidad del mensaje: Una autoridad asegura mediante evidencias que el mensaje es auténtico. Para ello se emplea un certificado electrónico cualificado.
- No repudio: El mensaje tiene validez como prueba de que la comunicación se produjo.
- Cifrado: Se asegura la información sensible del contenido de forma que solo el destinatario y ningún otro pueda leerlo.
- El protocolo además permite que diferentes sistemas puedan hablar el mismo idioma. Es decir, permite la interoperabilidad entre sistemas heterogéneos.

En un ciclo de autenticación intervienen dos elementos de SAML2: las aserciones de petición (AuthnRequest) y respuesta (Response).

Durante el proceso se intercambian los siguientes mensajes SAML2:

1) 1º AuthnRequest que el SP realiza para el intermediador Cl@ve.
2) 2º AuthnRequest que el intermediador Cl@ve prepara para el IdP.
3) 1º Response, la que el IdP realiza para el intermediador Cl@ve. El IdP responde los valores solicitados, firmados y cifrados (en el caso de que se requiera).
4) 2º Response, la que intermediador Cl@ve realiza al SP que inició la petición de autenticación.

Un SP sólo necesita tener en cuenta las peticiones de los puntos 1 y 2.

5.2 Peticiones SAML2

Las peticiones SAML2 (AuthnRequest) se inician desde un SP y son firmadas por éste. Estas peticiones se envían al nodo Clave mediante HTTP Post o HTTP Redirect, aunque se recomienda el tipo POST.

El nodo Clave, tras recibir a un usuario que porta un ticket SAML2 de autenticación, verifica la integridad/autenticidad de los mensajes de la petición. Esto implica los siguientes pasos:

1) Extraer el certificado de firma del SP y validarlo.
2) Validar la firma del mensaje SAML2 enviado en la petición.

Las peticiones que no se puedan validar de esta forma son rechazadas.

Posteriormente se extraerá la lista de IdP's habilitados y se continuará con el flujo de negocio normal para realizar la autenticación.

Dentro de la petición es obligatorio que esté la etiqueta 'headers'. Tal y como se ha diseñado el sistema Clave se requiere que, dentro de la etiqueta 'headers' se incluya el atributo 'Referer', que se corresponde con la URL desde donde se viene, es decir, la URL origen. Por ello, se recomienda [1]que el SP que lleve a cabo la comunicación con Clave tenga definida alguna de las siguientes políticas de seguridad 'Referrer-Policy':

- **origin:** Con este modificador, el valor 'Referrer' que se envía no especificará la dirección URL completa, y solo llevará en nombre del dominio y el protocolo. Es decir, se elimina el *Path* de la dirección URL de origen del hipervínculo.
- **origin-when-cross-origin:** En este caso, cuando el hipervínculo es dentro del mismo dominio, se envía la URL completa, y cuando el hipervínculo es a otro origen, se envía entonces solo el dominio y el protocolo de la URL, sin el *Path*.
- **unsafe-url:** En este caso, se envía la URL completa a cualquier destino, pero sin valores en los parámetros de la URL (para evitar el envío de nombres de usuarios, cookies de sesión o información sensible).

Dentro de la petición, también está la etiqueta 'postData', la cual contiene la etiqueta 'params, y dentro de la cual se encuentra la 'SAMLRequest'.

5.2.1 Detalle de la Cabecera Enviada desde el SP2 hacia Clave Pasarela (PROXY2) en el Entorno de PRE WILDFLY

En este apartado se muestra la petición en formato HAR (HTTP Archive) que hace el SP2 hacia Clave Pasarela (entorno de PRE Pasarela Wildfly).

[1] La definición de la política de seguridad 'Referrer-Policy' a utilizar por el SP es responsabilidad del propio SP. Sea cual sea la utilizada deberá obligar que el atributo 'Referer' esté presente dentro de la cabecera, tal y como se ha descrito.

A continuación, se adjunta el ejemplo en cuestión:


```
"request": {
  "method": "POST",
  "url": "https://pre-pasarela.clave.gob.es/Proxy2/ServiceProvider",
  "httpVersion": "HTTP/1.1",
  "headers": [
    {
      "name": "Accept",
      "value":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7"
    },
    {
      "name": "Accept-Encoding",
      "value": "gzip, deflate, br"
    },
    {
      "name": "Accept-Language",
      "value": "es-ES,es;q=0.9,ca;q=0.8,en;q=0.7,de;q=0.6,ca-
ES;q=0.5,eu;q=0.4,gl;q=0.3"
    },
    {
      "name": "Cache-Control",
      "value": "max-age=0"
    },
    {
      "name": "Connection",
      "value": "keep-alive"
    },
    {
      "name": "Content-Length",
      "value": "7268"
    },
    {
```

```
            "name": "Content-Type",
            "value": "application/x-www-form-urlencoded"
          },
          {
            "name": "Host",
            "value": "pre-pasarela.clave.gob.es"
          },
          {
            "name": "Origin",
            "value": "https://pre-pasarela.clave.gob.es"
          },
          {
            "name": "Referer",
            "value": "https://pre-pasarela.clave.gob.es/SP2/IndexPage"
          },
          {
            "name": "Sec-Fetch-Dest",
            "value": "document"
          },
          {
            "name": "Sec-Fetch-Mode",
            "value": "navigate"
          },
          {
            "name": "Sec-Fetch-Site",
            "value": "same-origin"
          },
          {
            "name": "Upgrade-Insecure-Requests",
            "value": "1"
          },
          {
            "name": "User-Agent",
            "value": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36"
```

```
            },
            {
              "name": "sec-ch-ua",
              "value": "\"Chromium\";v=\"112\", \"Google Chrome\";v=\"112\",
\"Not:A-Brand\";v=\"99\""
            },
            {
              "name": "sec-ch-ua-mobile",
              "value": "?0"
            },
            {
              "name": "sec-ch-ua-platform",
              "value": "\"Windows\""
            }
          ],
          "queryString": [],
          "cookies": [],
          "headersSize": 932,
          "bodySize": 7268,
          "postData": {
            "mimeType": "application/x-www-form-urlencoded",
            "text":
"RelayState=_EIZfJ9f&SAMLRequest=PHNhbWwycDpBdXRoblJlcXVlc3QgeG1sbnM6ZHM9Imh0dHA
6Ly93d3cudzMub3JnLzIwMDAvMDkveG1sZHNpZyMiIHhtbG5zOmVpZGFzPSJodHRwOi8vZWlkYXMuZXV
yb3BhLmV1L3NhbWwtZXh0ZW5zaW9ucyIgeG1sbnM6ZWlkYXMtbmF0dXJhbD0iaHR0cDovL2VpZGFzLmV
1cm9wYS5ldS9hdHRyaWJ1dGVzL25hdHVyYWxzZXJzb24iIHhtbG5zOnNhbWwyPSJ1cm46b2FzaXM6bmF
tZXM6dGM6U0FNTDoyLjA6YXNzZXJ0aW9uIiB4bWxuczpzYW1sMnA9InVybjpvYXNpczpuYW1lczp0Yzp
TQU1MOjIuMDpwcm90b2NvbCIgQXNzZXJ0aW9uQ29uc3VtZXJTZXJ2aWNlVVJMPSJodHRwczovL3ByZS1
wYXNhcmVsYS5jbGF2ZS5nb2IuZXMvU1AyL1JldHVyblBhZ2UiIENvbnNlbnQ9InVybjpvYXNpczpuYW1
lczp0YzpTQU1MOjIuMDpjb25zZW50OnVuc3BlY2lmaWVkIiBEZXN0aW5hdGlvbj0iaHR0cHM6Ly9wcmU
tcGFzYXJlbGEuY2xhdmUuZ29iLmVzL1Byb3h5Mi9TZXJ2aWNlUHJvdmlkZXIiIEZvcmNlQXV0aG49ImZ
hbHNlIiBJRD0iX0FCZXhaal9KUml1UmxaOXJHdjRhUzB4cWVxS1ViNHU0X1R3MlhHLVFoamFDX0s0MHZ
aLUxvbEF4aUIxcDZ6WSIgSXNQYXNzaXZlPSJmYWxzZSIgSXNzdWVJbnN0YW50PSIyMDIzLTA1LTAzVDA
4OjU0OjE4LjIyOFoiIFByb3ZpZGVyTmFtZT0iMTExMTExMTFIX0UwNDk5NTkwMjtEZW1vLVNQIiBWZXJ
zaW9uPSIyLjAiPjxkczpTaWduYXR1cmU%2BPGRzOlNpZ25lZEluZm8%2BPGRzOkNhbm9uaWNhbGl6YXR
pb25NZXRob2QgQWxnb3JpdGhtPSJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzEwL3htbC1leGMtYzE0biM
iLz48ZHM6U2lnbmF0dXJlTWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8wNC9
4bWxkc2lnLW1vcmUjcnNhLXNoYTUxMiIvPjxkczpSZWZlcmVuY2UgVVJJPSIjX0FCZXhaal9KUml1Umx
aOXJHdjRhUzB4cWVxS1ViNHU0X1R3MlhHLVFoamFDX0s0MHZaLUxvbEF4aUIxcDZ6WSI%2BPGRzOlRyY
W5zZm9ybXM%2BPGRzOlRyYW5zZm9ybSBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvMDk
veG1sZHNpZyNlbnZlbG9wZWQtc2lnbmF0dXJlIi8%2BPGRzOlRyYW5zZm9ybSBBbGdvcml0aG09Imh0d
```

HA6Ly93d3cudzMub3JnLzIwMDEvMTAveG1sLWV4Yy1jMTRuIyI%2BPGVjOkluY2x1c2l2ZU5hbWVzcGF
jZXMgeG1sbnM6ZWM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMTAveG1sLWV4Yy1jMTRuIyIgUHJlZml
4TGlzdD0iZWlkYXMtbmF0dXJhbCIvPjwvZHM6VHJhbnNmb3JtPjwvZHM6VHJhbnNmb3Jtcz48ZHM6RGl
nZXN0TWV0aG9kIEFsZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWxlbmMjc2hhNTE
yIi8%2BPGRzOkRpZ2VzdFZhbHVlPjdnMVArSmtkWE54cWZIcTVDaWIwaFRtMGMyV2lSZzdmVngyUXYzb
nZ4N2RIV2ZRd3hnbjc5VTVWM01LWk15WVpZV0VXRTBFdFNPRFVpTDBtd3JSTjZBPT08L2RzOkRpZ2Vzd
FZhbHVlPjwvZHM6UmVmZXJlbmNlPjwvZHM6U2lnbmVkSW5mbz48ZHM6U2lnbmF0dXJlVmFsdWU%2BWmI
vVG1zamcxYUYybDNtYzlRQmtwTUorbGRVejdVM2wyMk5WM2RKZU5KSGlPQWJGSjZYeU1VNHZHQ2VUczg
yMWFHREZobjE3Vmd2cGFvREJQSmJqM1BhK2ZBZVVIYlF5Nm1SUUhpOVJKYmw1aCsyTC9velVQRTV2YW1
jUmFrU0Faa3M5MTQvbSt2THljVUpBcG9EbG5kR0pjV3pHNHQySXZxMThEemNjNEJqK1RVK2xSZG83SXh
heVRLNXpIYmNyV0U4NzQzSjQ2MVlQMHdDTHh0NXhQbmdGMUZob3BaNlhjRVU0dzdVaFFqU2pidU5Kd0x
ydm1qSnJPR1VOMnRTUnhyWEVMZkt4TkJSSDRwaHVsTjh5VkhTQWU2NVMrL01pNm55NW5kbHpsa3RIak4
5Z2oxaFp3eG9SQ2RneWZxUDBrcVVRbFdLc2VONUpEMjc5U3JUSW93PT08L2RzOlNpZ25hdHVyZVZhbHV
lPjxkczpLZXlJbmZvPjxkczpYNTA5RGF0YT48ZHM6WDUwOUNlcnRpZmljYXRlPk1JSUhyakNDQnBhZ0F
3SUJBZ0lRZnQ3KzBkZUFrNUpoRlFsUkJQSzluakFOQmdrcWhraUc5dzBCQVFzRkFEQkhNUXN3Q1FZRFZ
RUUcKRXdKRlV6RVJNQThHQTFVRUNnd0lSazVOVkMxU1EwMHhKVEFqQmdOVkJBc01IRUZESUVOdmJYQnZ
ibVZ1ZEdWeklFbHVabTl5YmNPaApkR2xqYjNNd0hoY05NakV3T0RFeU1URTBNekV6V2hjTk1qUXdPREV
5TVRFME16RXlXakNCN0RFTE1Ba0dBMVVFQmhNQ1JWTXhEekFOCkJnTlZCQWNNQmsxQlJGSkpSREZDTUV
BR0ExVUVDZ3c1VFVsT1NWTlVSVkpKVHlCRVJTQkJVMVZPVkU5VElFVkRUMDVQVFVsRFQxTWcKV1NCVVV
rRk9VMFpQVWsxQlEwbFBUaUJFU1VkSlZFRk1NVFV3TXdZRFZRUUxEQ3hUUlVOU1JWUkJVa2xCSUVkRlR
rVlNRVXdnUkVVZwpRVVJOU1U1SlUxUlNRVU5KVDA0Z1JFbEhTVlJCVERFU01CQUdBMVVFQlJNSlV6STR
NREExTmpoRU1SZ3dGZ1lEVlFSaERBOVdRVlJGClV5MVRNamd3TURVMk9FUXhJekFoQmdOVkJBTU1HbE5
GVEV4UElFVk9WRWxFUVVRZ1UwZEJSQ0JRVWxWRlFrRlRNSUlCSWpBTkJna3EKaGtpRzl3MEJBUUVGQUF
PQ0FROEFNSUlCQ2dLQ0FRRUF2ZE9CL21SS3pGSlNaS2JEckJ2Wm9vbW8rWXVjK0lLcjl1aVlJUnZUZ3o
reQpFcXVjVlJlalJoTWpzSWgxTWlUMUdsUmIrVjlpU1Q2cFJqN3QvYVM4SDZTcXppekl5NzU2VGdzSno
4R1ZSYk9mWDJBMVhDTjVRSzBGCm85NkhtY0FEVmowMU0xOHplK1ZRejdZRzBRL29uYmR4NUlad05uY3h
PbjNlMGZHdzJURWI4NXd1eW5oQk5EM2NpMjM0MSt6aC96aGMKSEVkMHJNWHY2TkprVGkyRGlTNWFWeDg
vb3U0TGpGdW05SG1GQnJJT2ZiVnY4citRNVcxcTQ5NEhsUnFHcS9yblR5R2lzcTNZQUMyaQpFRS9jdEp
KZjg2ZHphL2IwOGxiOXlGVCtXQm1XNlpzNUFhM0N2STUrZG5HckVKay9PM3YrSmlCTWFEYU16a3dGWjh
Oa0NRSURBUUFCCm80SUQ3akNDQStvd0RBWURWUjBUQVFIL0JBSXdBRENCZ1FZSUt3WUJCUVVIQVFFRWR
UQnpNRHNHQ0NzR0FRVUZCekFCaGk5b2RIUncKT2k4dmIyTnpjR052YlhBdVkyVnlkQzVtYm0xMExtVnp
MMjlqYzNBdlQyTnpjRkpsYzNCdmJtUmxjakEwQmdnckJnRUZCUWN3QW9ZbwphSFIwY0RvdkwzZDNkeTV
qWlhKMExtWnViWFF1WlhNdlkyVnlkSE12UVVORFQwMVFMbU55ZERDQ0FUUUdBMVVkSUFTQ0FTc3dnZ0V
uCk1JSUJHQVlLS3dZQkJBR3NaZ01KRXpDQ0FRZ3dLUVlJS3dZQkJRVUhBZ0VXSFdoMGRIQTZMeTkzZDN
jdVkyVnlkQzVtYm0xMExtVnoKTDJSd1kzTXZNSUhhQmdnckJnRUZCUWNDQWpDQnpReUJ5a05sY25ScFp
tbGpZV1J2SUdOMVlXeHBabWxqWVdSdklHUmxJSE5sYkd4dgpJR1ZzWldOMGNzT3pibWxqYnlCelpXZkR
1bTRnY21WbmJHRnRaVzUwYnlCbGRYSnZjR1Z2SUdWSlJFRlRMaUJUZFdwbGRHOGdZU0JzCllYTWdZMjl
1WkdsamFXOXVaWE1nWkdVZ2RYTnZJR1Y0Y0hWbGMzUmhjeUJsYmlCc1lTQkVVRU1nWkdVZ1JrNU5WQzF
TUTAwZ1kyOXUKSUU1SlJqb2dVVEk0TWpZd01EUXRTaFVuXk5S2IzSm5aU0JLZFdGdUlERXdOaTB5T0R
Bd09TMU5ZV1J5YVdRdFJYTndZY094WVNrdwpDUVlIQkFDTDdFQUJBVEE0QmdOVkhSUVNVEF2cEMwd0t
6RXBNQ2NHQ1NzR0FRUUJyR1lCQ0F3YVUwVk1URThnUlU1VVNVUkJSQ0JUClIwRkVJRkJTVlVWQ1FWTXd
EZ1lEVlIwUEFRSC9CQVFEQWdYZ01CMEdBMVVkRGdRV0JCUi9XVm1wUkxhd2xIeGV3QzRsSnNxdEVnUUU
KTnpDQnNBWUlLd1lCQlFVSEFRTUVnYU13Z2FBd0NBWUdCQUNPUmdFQk1Bc0dCZ1FBamtZQkF3SUJEekF
UQmdZRUFJNUdBUVl3Q1FZSApCQUNPUmdFR0FqQnlCZ1lFQUk1R0FRVXdhREF5Rml4b2RIUndjem92TDN
kM2R5NWpaWEowTG1adWJYUXVaWE12Y0dSekwxQkVVMTlEClQwMVFYMlZ6TG5Ca1poTUNaWE13TWhZc2F
IUjBjSE02THk5M2QzY3VZMlZ5ZEM1bWJtMTBMbVZ6TDNCa2N5OVFSRk5mUTA5TlVGOWwKYmk1d1pHWVR
BbVZ1TUI4R0ExVWRJd1FZTUJhQUZCbjRXQzhVMXFiTW13U1lDQTFNMTZzQXA0TmxNSUhnQmdOVkhSOEV
nZGd3Z2RVdwpnZEtnZ2MrZ2djeUdnWjVzWkdGd09pOHZiR1JoY0dOdmJYQXVZMlZ5ZEM1bWJtMTBMbVZ
6TDBOT1BVTlNUREVzVDFVOVFVTWxNakJECmIyMXdiMjVsYm5SbGN5VXlNRWx1Wm05eWJXRjBhV052Y3l
4UFBVWk9UVlF0VWtOTkxFTTlSVk0vWTJWeWRHbG1hV05oZEdWU1pYWnYKWTJGMGFXOXVUR2x6ZER0aWF

XNWhjbmsvWW1GelpUOXZZbXBsWTNSamJHRnpjejFqVWt4RWFYTjBjbWxpZFhScGIyNVFiMmx1ZElZcAp
hSFIwY0RvdkwzZDNkeTVqWlhKMExtWnViWFF1WlhNdlkzSnNjMk52YlhBdlExSk1NUzVqY213d0RRWUp
Lb1pJaHZjTkFRRUxCUUFECmdnRUJBQ1N5Tm5qM0tkMDVvQjdmU2NGaisyc3k1Q0t0SHJ5aUNJd3JsTEF
HSFdTMHJpazIwV1NyeDEraGdrUkp4Z3NLdnFINW14RHoKZGRXZmc5MytOd0VQNkpDSFBNdFBxZldNYUx
IR3VIUWZSWXhLSngwMGFrdWZoZ0gya25wQUVGSmhCNVNHRVUwZ1F1UlNpanZGNSthKwpWaFg5ZXZMdDN
Sd25HWUl3Y1BvMzRNVGxyK1BIUjdQVGd6akd4WW4vOGc4ZXRyT2VHZTNlWkFJcDBZTlREU1VBMUJMU2R
oRGpNNnpQCjdEZFRPRkpHY2Z1M3BKQXgzRXB1RUJhU1FseXhjMkZXRHFXclA2eFBLZ3o1VmRxc0o0Wmx
la2thMFNSSSt1Yi9kTlE0STQ0UGorcXMKMTVGV1NVRjNNdCtWQk5pemcyb21IcGRjY2NCMnhwZ0JzQk4
1azBHelQ1MD08L2RzOlg1MDlDZXJ0aWZpY2F0ZT48L2RzOlg1MDlEYXRhPjwvZHM6S2V5SW5mbz48L2R
zOlNpZ25hdHVyZT48c2FtbDJwOkV4dGVuc2lvbnM%2BPGVpZGFzOlJlcXVlc3RlZEF0dHJpYnV0ZXM%2
BPGVpZGFzOlJlcXVlc3RlZEF0dHJpYnV0ZSBGcmllbmRseU5hbWU9IlJlbGF5U3RhdGUiIE5hbWU9Imh
0dHA6Ly9lcy5taW5oYWZwLmNsYXZlL1JlbGF5U3RhdGUiIE5hbWVGb3JtYXQ9InVybjpvYXNpczpuYW1
lczp0YzpTQU1MOjIuMDphdHRybmFtZS1mb3JtYXQ6dXJpIiBpc1JlcXVpcmVkPSJmYWxzZSI%2BPGVpZ
GFzOkF0dHJpYnV0ZVZhbHVlIHhtbG5zOnhzaT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS9YTUxTY2hlb
WEtaW5zdGFuY2UiIHhzaTp0eXBlPSJlaWRhcy1uYXR1cmFsOlBlcnNvbklkZW50aWZpZXJUeXBlIj5fN
UZMNzl2azwvZWlkYXM6QXR0cmlidXRlVmFsdWU%2BPC9laWRhczpSZXF1ZXN0ZWRBdHRyaWJ1dGU%2BP
C9laWRhczpSZXF1ZXN0ZWRBdHRyaWJ1dGVzPjwvc2FtbDJwOkV4dGVuc2lvbnM%2BPHNhbWwycDpOYW1
lSURQb2xpY3kgQWxsb3dDcmVhdGU9InRydWUiIEZvcm1hdD0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw
6MS4xOm5hbWVpZC1mb3JtYXQ6dW5zcGVjaWZpZWQiLz48c2FtbDJwOlJlcXVlc3RlZEF1dGhuQ29udGV
4dCBDb21wYXJpc29uPSJtaW5pbXVtIj48c2FtbDI6QXV0aG5Db250ZXh0Q2xhc3NSZWY%2BaHR0cDovL
2VpZGFzLmV1cm9wYS5ldS9Mb0EvbG93PC9zYW1sMjpBdXRobkNvbnRleHRDbGFzc1JlZj48L3NhbWwyc
DpSZXF1ZXN0ZWRBdXRobkNvbnRleHQ%2BPC9zYW1sMnA6QXV0aG5SZXF1ZXN0Pg%3D%3D",
"params": [
{
"name": "RelayState",
"value": "_EIZfJ9f"
},
{
"name": "SAMLRequest",
"value":
"PHNhbWwycDpBdXRoblJlcXVlc3QgeG1sbnM6ZHM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvMDkveG1
sZHNpZyMiIHhtbG5zOmVpZGFzPSJodHRwOi8vZWlkYXMuZXVyb3BhLmV1L3NhbWwtZXh0ZW5zaW9ucyI
geG1sbnM6ZWlkYXMtbmF0dXJhbD0iaHR0cDovL2VpZGFzLmV1cm9wYS5ldS9hdHRyaWJ1dGVzL25hdHV
yYWxwZXJzb24iIHhtbG5zOnNhbWwyPSJ1cm46b2FzaXM6bmFtZXM6dGM6U0FNTDoyLjA6YXNzZXJ0aW9
uIiB4bWxuczpzYW1sMnA9InVybjpvYXNpczpuYW1lczp0YzpTQU1MOjIuMDpwcm90b2NvbCIgQXNzZXJ
0aW9uQ29uc3VtZXJTZXJ2aWNlVVJMPSJodHRwczovL3ByZS1wYXNhcmVsYS5jbGF2ZS5nb2IuZXMvU1A
yL1JldHVyblBhZ2UiIENvbnNlbnQ9InVybjpvYXNpczpuYW1lczp0YzpTQU1MOjIuMDpjb25zZW50OnV
uc3BlY2lmaWVkIiBEZXN0aW5hdGlvbj0iaHR0cHM6Ly9wcmUtcGFzYXJlbGEuY2xhdmUuZ29iLmVzL1B
yb3h5Mi9TZXJ2aWNlUHJvdmlkZXIiIEZvcmNlQXV0aG49ImZhbHNlIiBJRD0iX0FCZXhaal9KUml1Umx
aOXJHdjRhUzB4cWVxS1ViNHU0X1R3MlhHLVFoamFDX0s0MHZaLUxvbEF4aUIxcDZ6WSIgSXNQYXNzaXZ
lPSJmYWxzZSIgSXNzdWVJbnN0YW50PSIyMDIzLTA1LTAzVDA4OjU0OjE4LjIyOFoiIFByb3ZpZGVyTmF
tZT0iMTExMTExMTFIX0UwNDk5NTkwMjtEZW1vLVNQIiBWZXJzaW9uPSIyLjAiPjxkczpTaWduYXR1cmU
%2BPGRzOlNpZ25lZEluZm8%2BPGRzOkNhbm9uaWNhbGl6YXRpb25NZXRob2QgQWxnb3JpdGhtPSJodHR
wOi8vd3d3LnczLm9yZy8yMDAxLzEwL3htbC1leGMtYzE0biMiLz48ZHM6U2lnbmF0dXJlTWV0aG9kIEF
sZ29yaXRobT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWxkc2lnLW1vcmUjcnNhLXNoYTUxMiI
vPjxkczpSZWZlcmVuY2UgVVJJPSIjX0FCZXhaal9KUml1UmxaOXJHdjRhUzB4cWVxS1ViNHU0X1R3Mlh

HLVFoamFDX0s0MHZaLUxvbEF4aUIxcDZ6WSI%2BPGRzOlRyYW5zZm9ybXM%2BPGRzOlRyYW5zZm9ybSB
BbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvMDkveG1sZHNpZyNlbnZlbG9wZWQtc2lnbmF
0dXJlIi8%2BPGRzOlRyYW5zZm9ybSBBbGdvcml0aG09Imh0dHA6Ly93d3cudzMub3JnLzIwMDEvMTAve
G1sLWV4Yy1jMTRuIyI%2BPGVjOkluY2x1c2l2ZU5hbWVzcGFjZXMgeG1sbnM6ZWM9Imh0dHA6Ly93d3c
udzMub3JnLzIwMDEvMTAveG1sLWV4Yy1jMTRuIyIgUHJlZml4TGlzdD0iZWlkYXMtbmF0dXJhbCIvPjw
vZHM6VHJhbnNmb3JtPjwvZHM6VHJhbnNmb3Jtcz48ZHM6RGlnZXN0TWV0aG9kIEFsZ29yaXRobT0iaHR
0cDovL3d3dy53My5vcmcvMjAwMS8wNC94bWxlbmMjc2hhNTEyIi8%2BPGRzOkRpZ2VzdFZhbHVlPjdnM
VArSmtkWE54cWZIcTVDaWIwaFRtMGMyV2lSZzdmVngyUXYzbnZ4N2RIV2ZRd3hnbjc5VTVWM01LWk15W
VpZV0VXRTBFdFNPRFVpTDBtd3JSTjZBPT08L2RzOkRpZ2VzdFZhbHVlPjwvZHM6UmVmZXJlbmNlPjwvZ
HM6U2lnbmVkSW5mbz48ZHM6U2lnbmF0dXJlVmFsdWU%2BWmIvVG1zamcxYUYybDNtYzlRQmtwTUorbGR
VejdVM2wyMk5WM2RKZU5KSGlPQWJGSjZYeU1VNHZHQ2VUczgyMWFHREZobjE3Vmd2cGFvREJQSmJqM1B
hK2ZBZVVIYlF5Nm1SUUhpOVJKYmw1aCsyTC9velVQRTV2YW1jUmFrU0Faa3M5MTQvbSt2THljVUpBcG9
EbG5kR0pjV3pHNHQySXZxMThEemNjNEJqK1RVK2xSZG83SXhheVRLNXpIYmNyV0U4NzQzSjQ2MVlQMHd
DTHh0NXhQbmdGMUZob3BaNlhjRVU0dzdVaFFqU2pidU5Kd0xydm1qSnJPR1VOMnRTUnhyWEVMZkt4TkJ
SSDRwaHVsTjh5VkhTQWU2NVMrL01pNm55NW5kbHpsa3RIak45Z2oxaFp3eG9SQ2RneWZxUDBrcVVRbFd
Lc2VONUpEMjc5U3JUSW93PT08L2RzOlNpZ25hdHVyZVZhbHVlPjxkczpLZXlJbmZvPjxkczpYNTA5RGF
0YT48ZHM6WDUwOUNlcnRpZmljYXRlPk1JSUhyakNDQnBhZ0F3SUJBZ0lRZnQ3KzBkZUFrNUpoRlFsUkJ
QSzluakFOQmdrcWhraUc5dzBCQVFzRkFEQkhNUXN3Q1FZRFZRUUcKRXdKRlV6RVJNQThHQTFVRUNnd0l
SazVOVkMxU1EwMHhKVEFqQmdOVkJBc01IRUZESUVOdmJYQnZibVZ1ZEdWeklFbHVabTl5YmNPaApkR2x
qYjNNd0hoY05NakV3T0RFeU1URTBNekV6V2hjTk1qUXdPREV5TVRFME16RXlXakNCN0RFTE1Ba0dBMVV
FQmhNQ1JWTXhEekFOCkJnTlZCQWNNQmsxQlJGSkpSREZDTUVBR0ExVUVDZ3c1VFVsT1NWTlVSVkpKVHl
CRVJTQkJVMVZPVkU5VElFVkRUMDVQVFVsRFQxTWcKV1NCVVVrRk9VMFpQVWsxQlEwbFBUaUJFU1VkSlZ
FRk1NVFV3TXdZRFZRUUxEQ3hUUlVOU1JWUkJVa2xCSUVkRlRrVlNRVXdnUkVVZwpRVVJOU1U1SlUxUlN
RVU5KVDA0Z1JFbEhTVlJCVERFU01CQUdBMVVFQlJNSlV6STRNREExTmpoRU1SZ3dGZ1lEVlFSaERBOVd
RVlJGClV5MVRNamd3TURVMk9FUXhJekFoQmdOVkJBTU1HbE5GVEV4UElFVk9WRWxFUVVRZ1UwZEJSQ0J
RVWxWRlFrRlRNSUlCSWpBTkJna3EKaGtpRzl3MEJBUUVGQUFPQ0FROEFNSUlCQ2dLQ0FRRUF2ZE9CL21
SS3pGSlNaS2JEckJ2Wm9vbW8rWXVjK0lLcjl1aVlJUnZUZ3oreQpFcXVjVlJlalJoTWpzSWgxTWlUMUd
sUmIrVjlpU1Q2cFJqN3QvYVM4SDZTcXppekl5NzU2VGdzSno4R1ZSYk9mWDJBMVhDTjVRSzBGCm85Nkh
tY0FEVmowMU0xOHplK1ZRejdZRzBRL29uYmR4NUlad05uY3hPbjNlMGZHdzJURWI4NXd1eW5oQk5EM2N
pMjM0MSt6aC96aGMKSEVkMHJNWHY2TkprVGkyRGlTNWFWeDgvb3U0TGpGdW05SG1GQnJJT2ZiVnY4cit
RNVcxcTQ5NEhsUnFHcS9yblR5R2lzcTNZQUMyaQpFRS9jdEpKZjg2ZHphL2IwOGxiOXlGVCtXQm1XNlp
zNUFhM0N2STUrZG5HckVKay9PM3YrSmlCTWFEYU16a3dGWjhOa0NRSURBUUFCCm80SUQ3akNDQStvd0R
BWURWUjBUQVFIL0JBSXdBRENCZ1FZSUt3WUJCUVVIQVFFRWRUQnpNRHNHQ0NzR0FRVUZCekFCaGk5b2R
IUncKT2k4dmIyTnpjR052YlhBdVkyVnlkQzVtYm0xMExtVnpMMjlqYzNBdlQyTnpjRkpsYzNCdmJtUmx
jakEwQmdnckJnRUZCUWN3QW9ZbwphSFIwY0RvdkwzZDNkeTVqWlhKMExtWnViWFF1WlhNdlkyVnlkSE1
2UVVORFQwMVFMbU55ZERDQ0FUUUdBMVVkSUFTQ0FTc3dnZ0VuCk1JSUJHQVlLS3dZQkJBR3NaZ01KRXp
DQ0FRZ3dLUVlJS3dZQkJRVUhBZ0VXSFdoMGRIQTZMeTkzZDNjdVkyVnlkQzVtYm0xMExtVnoKTDJSd1k
zTXZNSUhhQmdnckJnRUZCUWNDQWpDQnpReUJ5a05sY25ScFptbGpZV1J2SUdOMVlXeHBabWxqWVdSdkl
HUmxJSE5sYkd4dgpJR1ZzWldOMGNzT3pibWxqYnlCelpXZkR1bTRnY21WbmJHRnRaVzUwYnlCbGRYSnZ
jR1Z2SUdWSlJFRlRMaUJUZFdwbGRHOGdZU0JzCllYTWdZMjl1WkdsamFXOXVaWE1nWkdVZ2RYTnZJR1Y
0Y0hWbGMzUmhjeUJsYmlCc1lTQkVVRU1nWkdVZ1JrNU5WQzFTUTAwZ1kyOXUKSUU1SlJqb2dVVEk0TWp
Zd01EUXRTaUFvUXk5S2IzSm5aU0JLZFdGdUlERXdOaTB5T0RBd09TMU5ZV1J5YVdRdFJYTndZY094WVN
rdwpDUVlIQkFDTDdFQUJBVEE0QmdOVkhSRUVNVEF2cEMwd0t6RXBNQ2NHQ1NzR0FRUUJyR1lCQ0F3YVU
wVk1URThnUlU1VVNVUkJSQ0JUClIwRkVJRkJTVlVWQ1FWTXdEZ1lEVlIwUEFRSC9CQVFEQWdYZ01CMEd
BMVVkRGdRV0JCUi9XVm1wUkxhd2xIeGV3QzRsSnNxdEVnUUUKTnpDQnNBWUlLd1lCQlFVSEFRTUVnYU1
3Z2FBd0NBWUdCQUNPUmdFQk1Bc0dCZ1FBamtZQkF3SUJEekFUQmdZRUFJNUdBUVl3Q1FZSApCQUNPUmd
FR0FqQnlCZ1lFQUk1R0FRVXdhREF5Rml4b2RIUndjem92TDNkM2R5NWpaWEowTG1adWJYUXVaWE12Y0d
SekwxQkVVMTlECldwMVFYMlZ6TG5Ca1poTUNaWE13TWhZc2FIUjBjSE02THk5M2QzY3VZMlZ5ZEM1bWJ
tMTBMbVZ6TDNCa2N5OVFSRk5mUTA5TlVGOWwKYmk1d1pHWVRBbVZ1TUI4R0ExVWRJd1FZTUJhQUZCbjR
XQzhVMXFiTW13U1lDQTFNMTZzQXA0TmxNSUhnQmdOVkhSOEVnZGd3Z2RVdwpnZEtnZ2MrZ2djeUdnWjV

zWkdGd09pOHZiR1JoY0dOdmJYQXVZMlZ5ZEM1bWJtMTBMbVZ6TDBOT1BVTlNUREVzVDFVOVFVTWxNakJ
ECmIyMXdiMjVsYm5SbGN5VXlNRWx1Wm05eWJXRjBhV052Y3l4UFBVWk9UVlF0VWtOTkxFTTlSVk0vWTJ
WeWRHbG1hV05oZEdWU1pYWnYKWTJGMGFXOXVUR2x6ZER0aWFXNWhjbmsvWW1GelpUOXZZbXBsWTNSamJ
HRnpjejFqVWt4RWFYTjBjbWxpZFhScGIyNVFiMmx1ZElzcAphSFIwY0RvdkwzZDNkeTVqWlhKMExtWnV
iWFF1WlhNdlkzSnNjMk52YlhBdlExSk1NUzVqY213d0RRWUpLb1pJaHZjTkFRUxCUUFECmdnRUJBQ1N
5Tm5qM0tkMDVvQjdmU2NGaisyc3k1Q0t0SHJ5aUNJd3JsTEFHSFdTMHJpazIwV1NyeDEraGdrUkp4Z3N
LdnFINW14RHoKZGRXZmc5MytOd0VQNkpDSFBNdFBxZldNYUxIR3VIUWZSWXhLSngwMGFrdWZoZ0gya25
wQUVGSmhCNVNHRVUwZ1F1UlNpanZGNSthKwpWaFg5ZXZMdDNSd25HWUl3Y1BvMzRNVGxyK1BIUjdQVGd
6akd4WW4vOGc4ZXRyT2VHZTNlWkFJcDBZTlREU1VBMUJMU2RoRGpNNnpQCjdEZFRPRkpHY2Z1M3BKQXg
zRXB1RUJhU1FseXhjMkZXRHFXclA2eFBLZ3o1VmRxc0o0Wmxla2thMFNSSSt1Yi9kTlE0STQ0UGorcXM
KMTVGV1NVRjNNdCtWQk5pemcyb21IcGRjY2NCMnhwZ0JzQk41azBHelQ1MD08L2RzOlg1MDlDZXJ0aWZ
pY2F0ZT48L2RzOlg1MDlEYXRhPjwvZHM6S2V5SW5mbz48L2RzOlNpZ25hdHVyZT48c2FtbDJwOkV4dGV
uc2lvbnM%2BPGVpZGFzOlJlcXVlc3RlZEF0dHJpYnV0ZXM%2BPGVpZGFzOlJlcXVlc3RlZEF0dHJpYnV
0ZSBGcmllbmRseU5hbWU9IlJlbGF5U3RhdGUiIE5hbWU9Imh0dHA6Ly9lcy5taW5oYWZwLmNsYXZlL1J
lbGF5U3RhdGUiIE5hbWVGb3JtYXQ9InVybjpvYXNpczpuYW1lczp0YzpTQU1MOjIuMDphdHRybmFtZS1
mb3JtYXQ6dXJpIiBpc1JlcXVpcmVkPSJmYWxzZSI%2BPGVpZGFzOkF0dHJpYnV0ZVZhbHVlIHhtbG5zO
nhzaT0iaHR0cDovL3d3dy53My5vcmcvMjAwMS9YTUxTY2hlbWEtaW5zdGFuY2UiIHhzaTp0eXBlPSJla
WRhcy1uYXR1cmFsOlBlcnNvbklkZW50aWZpZXJUeXBlIj5fNUZMNzl2azwvZWlkYXM6QXR0cmlidXRlV
mFsdWU%2BPC9laWRhczpSZXF1ZXN0ZWRBdHRyaWJ1dGU%2BPC9laWRhczpSZXF1ZXN0ZWRBdHRyaWJ1d
GVzPjwvc2FtbDJwOkV4dGVuc2lvbnM%2BPHNhbWwycDpOYW1lSURQb2xpY3kgQWxsb3dDcmVhdGU9InR
ydWUiIEZvcm1hdD0idXJuOm9hc2lzOm5hbWVzOnRjOlNBTUw6MS4xOm5hbWVpZC1mb3JtYXQ6dW5zcGV
jaWZpZWQiLz48c2FtbDJwOlJlcXVlc3RlZEF1dGhuQ29udGV4dCBDb21wYXJpc29uPSJtaW5pbXVtIj4
8c2FtbDI6QXV0aG5Db250ZXh0Q2xhc3NSZWY%2BaHR0cDovL2VpZGFzLmV1cm9wYS5ldS9Mb0EvbG93P
C9zYW1sMjpBdXRobkNvbnRleHRDbGFzc1JlZj48L3NhbWwycDpSZXF1ZXN0ZWRBdXRobkNvbnRleHQ%2
BPC9zYW1sMnA6QXV0aG5SZXF1ZXN0Pg%3D%3D"

```
                }
              ]
            }
          },
```

5.3 Respuestas SAML2

En el caso de las respuestas SAML2, éstas también se envían firmadas. Adicionalmente, la aserción que contiene el propio mensaje de respuesta, donde residen los datos personales del ciudadano, puede firmarse también.

Las aserciones SAML2 contienen información personal por lo que deben cifrarse empleando para ello el certificado obtenido de la petición original, que es el empleado por el origen al que va dirigida la respuesta. Es decir, en cada paso de vuelta se descifran y vuelven a cifrar los datos personales del usuario. Es un cifrado adicional al cifrado que introduce el canal de comunicación SSL/TLS.

Por lo tanto, una vez identificado a un ciudadano, el IdP y posteriormente el intermediador responderán al SP con una respuesta SAML2 firmada y con los datos personales cifrados con el certificado indicado por el propio SP en su ticket de solicitud.

Cada eslabón de la cadena debe verificar la integridad/autenticidad de los mensajes SAML2 antes de procesar la respuesta. Esto implica los siguientes pasos:

1) Extraer el certificado de firma y validarlo.
2) Validar la firma del mensaje SAML2 enviado en la respuesta.
3) Si la aserción SAML2 está firmada, su firma debe ser validada también.
4) Descifrar los datos personales contenidos en la respuesta. Sólo el nodo de destino puede hacerlo.

Las respuestas que no se puedan validar de esta forma serán rechazadas.

5.4 Calidad de la Credencial (LoA) y Atributos Personales

Un elemento esencial en el marco de interoperabilidad del reglamento eIDAS es el nivel de seguridad de la autenticación. El nivel de seguridad caracteriza el grado de confianza de un medio de identificación electrónica para establecer la identidad de una persona, garantizando así que la persona que afirma poseer una identidad determinada la tiene.

El nivel de seguridad depende del grado de confianza que aporte este medio de identificación electrónica sobre la identidad pretendida o declarada por una persona, teniendo en cuenta los procedimientos técnicos, (por ejemplo, prueba y verificación de la identidad, autenticación), los procedimientos de gestión (métodos utilizados por entidad para expedir la identidad electrónica) y los controles aplicados a todo el proceso.

El reglamento eIDAS establece los tres niveles de seguridad siguientes:

- el nivel de seguridad **bajo** se referirá a un medio de identificación electrónica, en el contexto de un sistema de identificación electrónica, que establece un **grado limitado de confianza** en la identidad pretendida o declarada de una persona y se describe en referencia a las especificaciones técnicas, las normas y los procedimientos del mismo, entre otros los controles técnicos, y cuyo objetivo es **reducir el riesgo de uso indebido o alteración de la identidad**;
- el nivel de seguridad **sustancial** se referirá a un medio de identificación electrónica, en el contexto de un sistema de identificación electrónica, que establece un **grado sustancial de confianza** en la identidad pretendida o declarada de una persona y se describe en referencia a las especificaciones técnicas, las normas y los procedimientos del mismo, entre otros los controles técnicos, y cuyo objetivo es **reducir sustancialmente el riesgo de uso indebido o alteración de la identidad**;
- el nivel de seguridad **alto** se referirá a un medio de identificación electrónica, en el contexto de un sistema de identificación electrónica, que establece un **grado de confianza** en la identidad pretendida o declarada de una persona **superior** al medio de identificación electrónica con un nivel de seguridad sustancial, y se describe en referencia a las especificaciones técnicas, las normas y los procedimientos del mismo, entre otros los controles técnicos, cuyo objetivo es **evitar el uso indebido o alteración de la identidad**.

El nivel de LoA devuelto por los IDPs varía en función de distintos aspectos, como el IDP escogido, el nivel de registro (bajo/alto), el modo de registro (presencial, certificado SW, videoasistencia…), el soporte, etc.

Como resumen, los valores LoA que puede devolver los IDPs (http://eidas.europa.eu/LoA/) son los siguientes:

- AFIRMA: nivel máximo High,

- IDP @firma: Petición de usuario con certificado distinto a DNIe o certificado no emitido en dispositivo seguro (ej: FNMT): devuelve en la respuesta LoA=susbtancial.
- IDP @firma: Petición de usuario con certificado DNIe o certificado emitido en dispositivo seguro: devuelve en la respuesta LoA=high.

- EIDAS: nivel máximo High,
  - IDP EIDAs: Petición de usuario con certificado distinto a DNIe (ej: FNMT): devuelve error porque solo permite DNIe.
  - IDP EIDAs: Petición de usuario con certificado DNIe: devuelve en la respuesta LoA=High.
- AEAT: nivel máximo Substancial,
  - IDP Clave PIN/ IDP Móvil: Petición de usuario con registro Bajo: devolverá LoA=low.
  - IDP Clave PIN/ IDP Móvil: Petición de usuario con registro Alto con certificado electrónico (DNIe u otro personal): devuelve LoA=Substancial.
- GISS: nivel máximo Sustancial,
  - IDP Clave Permanente: petición de usuario registro débil: loa =Low.
  - IDP Clave Permanente: petición de usuario con registro fuerte sin OTP (loA entrada = low): LoW.
  - IDP Clave Permanente: petición de usuario con registro fuerte con OTP (loA entrada = sustancial): Substancial.

Además, el reglamento eIDAS establece los siguientes datos a proporcionar para las **personas físicas**:

- Un conjunto de **datos obligatorios** que deben proporcionarse en todos los casos. Estos datos son:
  - **Identificador de unicidad**: Se trata de un identificador vinculado de manera única a una persona determinada, que permite asociar a la misma persona autenticaciones sucesivas. Se debe hacer notar que este identificador garantiza que no habrá dos personas con el mismo identificador, como es el caso del NIF español.
  - **Nombre** (en general uno o varios nombres)
  - **Apellido** (en general los apellidos del ciudadano)
  - **Fecha de nacimiento**: fecha de nacimiento de la persona (solo disponible en algunos IdP de la pasarela Clave2)
- Un conjunto de **datos opcionales** que el emisor de la autenticación puede decidir proporcionar o no, en función de las características del esquema de identificación utilizado. El propósito de estos datos opcionales es facilitar la asociación de los datos de identificación del ciudadano en autenticaciones sucesivas. Estos datos opcionales son:
  - Nombre al nacer.
  - Apellido al nacer.
  - Lugar de nacimiento.
  - Dirección actual.

- Género.

6 Elementos que Componen la Arquitectura

6.1 Proveedor de Servicios (Service Provider o SP)

Es una aplicación (generalmente web), API o conjunto de aplicaciones que aportan la funcionalidad al usuario, a excepción del proceso de autenticación. En nuestro contexto se referirá a la aplicación de una entidad privada que ofrece servicios de cara al ciudadano.

Como SP, una entidad que interactúa con el nodo Cl@ve debe:

- Generar los tickets SAML2 de petición de autenticación que serán enviados al nodo Cl@ve.
- Validar tickets de respuesta SAML2 y descifrarlas para obtener la información que se desea conocer del ciudadano.
- En el nuevo modelo, los SP's no necesitan preguntar por los atributos personales que desea conocer del usuario. En su lugar, envía una petición vacía y la plataforma devuelve todos los atributos que es posible enviar al SP.

6.2 Proveedor de Identidad (Identity Provider o IdP)

De forma general, un IdP es el componente que almacena las identidades, y por tanto autentica e identifica a usuarios, atendiendo a las peticiones de datos que llegan desde los SP's pasando por Cl@ve o directamente.

La autenticación se obtiene lanzando un desafío al usuario que éste debe superar con algo que tiene, algo que sabe y/o algo que es.

6.3 Pasarela Cl@ve (Proxy entre SP's e IdP's)

Es el elemento central que permite encaminar al usuario hacia el IdP adecuado, atendiendo a los requisitos del proveedor de servicio. Recibe al usuario mediante una redirección web, que porta un ticket SAML firmado por un SP, que es validado por el intermediador. El intermediador prepara a su vez un ticket similar y lanza la solicitud contra el IdP. Posteriormente recibe la respuesta SAML del IdP, la descifra y valida y procede a preparar un ticket de respuesta similar para el SP.

Al tratarse de elementos centrales se simplifica el establecimiento de la confianza, ya que tanto los proveedores de servicio como los proveedores de identidad sólo se comunican con la pasarela. Esta simplificación robustece el sistema y protege la credencial del usuario, que nunca sale de la relación directa con el IdP.

6.4 eIDAS y el Identity Matching

La red de nodos eIDAS permite la identificación de **ciudadanos europeos** a través del sistema de identificación transfronteriza de la UE. Cada nodo eIDAS está conectado a los IdP de ese país.

Para este escenario aparecen dos nuevos componentes **intermediadores** además de la pasarela clave: el nodo eIDAS español (actuando como conector) y el nodo eIDAS del otro país de la UE que el ciudadano indica. Será este segundo nodo transfronterizo quien llamará al IdP de ese país para identificar al ciudadano y proporcionar sus credenciales de identidad, incluyendo un identificador conforme a las reglas de ese IdP transfronterizo y ese país.

En el caso del nodo español, se incorpora el componente adicional "ID UE Provider - Servicio de Identificación de ciudadanos eIDAS" de la Agencia Tributaria AEAT para localizar el identificador nacional del ciudadano en España (DNI, NIE, NIF-L o NIF-M) y proporcionarlo a través de un nuevo atributo, el **NationalPersonIdentifier**. Este proceso, por el que se asocia un identificador español a una credencial de identidad transfronteriza, se conoce como ***Identity Matching*** y garantiza la identificación unívoca y su persistencia temporal conforme a los requisitos del Real Decreto 203/2021 (artículos 27 y 28).

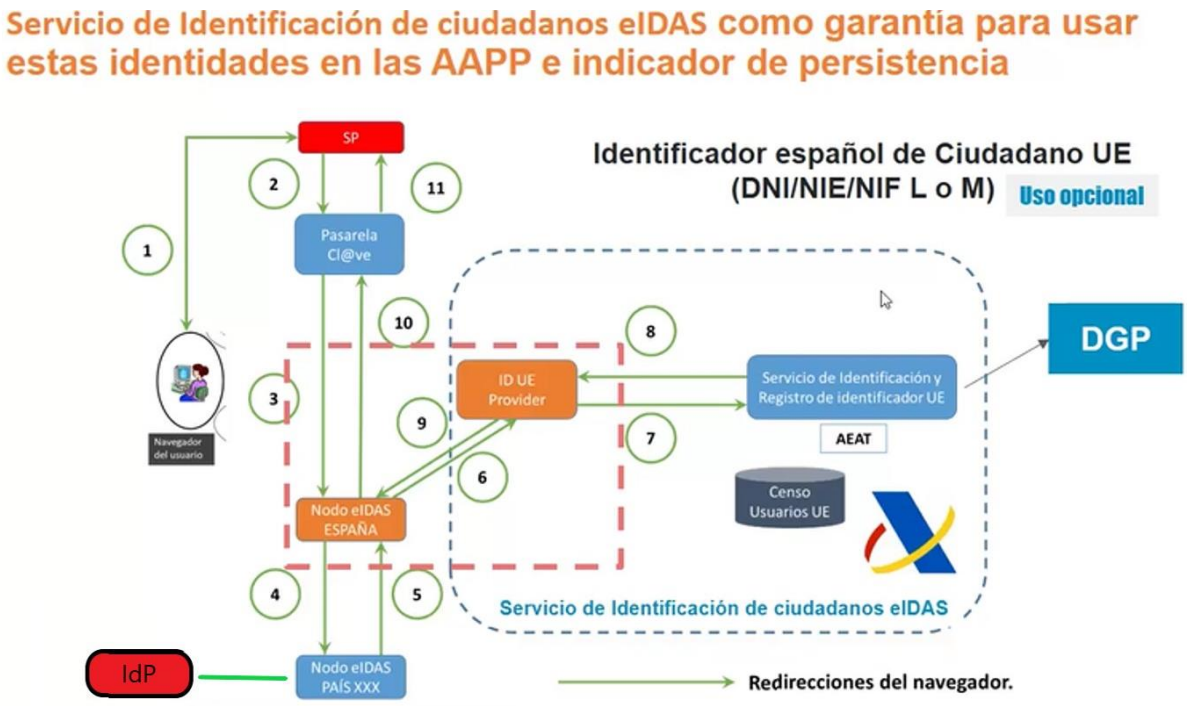


Ilustración 2: Flujos por los que pasa una petición de identificación transfronteriza eIDAS

Hay que tener en cuenta que el atributo **NationalPersonIdentifier** es **opcional**, en varios sentidos:

- El servicio tiene potestad de solicitarlo o no.
- En el caso de solicitarlo, siempre la petición SAML hacia Pasarela Cl@ve debe contener el atributo *NationalPersonIdentifier* como un campo no obligatorio (isRequired="false") – ver sección 8.3.1.4. Sólo en este caso se lanzará el proceso de *Identity Matching*.
- En la respuesta obtenida puede no llegar informado este atributo si no se ha localizado un identificador nacional asociado a la credencial transfronteriza del ciudadano.

Durante el proceso de *Identity Matching* puede ser necesario en algunos casos la interacción del ciudadano con el servicio de la AEAT:

- La primera vez que un ciudadano accede al servicio con una credencial de identidad transfronteriza y el servicio de la AEAT localiza un identificador nacional con alta probabilidad de pertenecer a ese ciudadano, le pide **que facilite su identificador nacional como medida de refuerzo** para garantizar que efectivamente el resultado de la localización es correcto, además de un **correo electrónico** para habilitar un mecanismo de recuperación y de comunicación de posibles problemas que puedan surgir con su autenticación. Estos datos junto a los datos de la credencial transfronteriza se guardan en el *Censo de Ciudadanos UE* que gestiona la AEAT.
- Si se detecta que puede haber un identificador nacional en el Censo de Ciudadanos UE asociado a la credencial transfronteriza pero **no se tiene certeza**, se le pide que lo introduzca como **dato de contraste**. Si no lo recuerda, puede solicitar su recuperación.
- Si no se consigue localizar un identificador nacional que pueda pertenecer al ciudadano, se le indica que puede **personarse en un consulado** para solicitarlo.

7 Proceso de Alta y Requisitos de Acceso

El funcionamiento del sistema se basa en el establecimiento de un círculo de confianza entre el proveedor de servicios y la plataforma Cl@ve. Para ello será necesario que el proveedor de servicios obtenga un certificado para la firma de los tokens SAML2, cuya parte pública remitirá al gestor del nodo Cl@ve.

Esto supone que esos integradores tendrán que disponer de la capacidad para crear y procesar tickets SAML2. Para ello, se puede optar por dos estrategias de integración:

- Si no se dispone de un motor capaz de crear tickets SAML2, se proporcionan unas librerías de integración que incluyen el motor SAML2 y otras herramientas para integrarse. Estas utilidades, descritas en el presente documento, incluye el paquete de integración en Java, PHP y .Net, así como una plataforma de demo que se puede consultar como referencia, y contra la que integrar un SP localmente.
- Si se dispone de un motor SAML2, la integración puede hacerse directamente a través de intercambio de tickets SAML2 de acuerdo con el formato especificado.

Por su parte, el gestor del intermediador proporcionará al proveedor de servicio la parte pública del certificado usado para la firma de sus propios tokens SAML.

Adicionalmente, en este mismo documento se indican las URL, accesibles desde Internet, a las cuales se deben enviar las peticiones del entorno de pruebas, para que el proveedor de servicio pueda realizar todas las pruebas de conexión necesarias antes del paso a producción, cuya URL también se facilitará una vez que se haya determinado que la integración en el entorno de pruebas ha sido correcta.

8 Parámetros y Mensajes SAML

Como se ha descrito en los anteriores apartados, la integración con el sistema de identificación Cl@ve 2, y los diferentes componentes que lo conforman, se lleva a cabo mediante el intercambio de mensajes SAML

v2.0, formato eIDAS, los cuales viajan como parámetros de las peticiones HTTP POST realizadas a través de redirecciones desde el navegador del usuario.

En este apartado se especificarán los parámetros HTTP admitidos por el componente pasarela del sistema de identificación Cl@ve 2, así como los atributos y particularidades de los mensajes SAML intercambiados.

8.1 Parámetros HTTP

Los parámetros HTTP admitidos por el componente pasarela del sistema de identificación Cl@ve 2 son los siguientes:

- **RelayState**. Es un parámetro alfanumérico externo que dificulta ataques de repetición. Debe ser incluido en las peticiones y se devolverá en modo eco en las respuestas y en las peticiones de Logout con el mismo parámetro.
- **SAMLRequest**. Este parámetro transporta el token SAML, codificado en Base64, mediante el que un SP solicita el inicio de un proceso de autenticación, y de sesión SSO, de un usuario.
- **SAMLResponse.** Este parámetro transporta el token SAML de respuesta, codificado en Base64, mediante el que el componente pasarela comunica el resultado de una solicitud de autenticación, e inicio de sesión SSO, de un usuario a un SP.
- **logoutRequest.** Este parámetro transporta el token SAML, codificado en Base64, mediante el que un SP solicita el cierre de sesión SSO de un usuario.
- **logoutResponse**. Este parámetro transporta el token SAML de respuesta, codificado en Base64, mediante el que el componente pasarela comunica el resultado de una solicitud de cierre de sesión SSO a un SP.

8.2 Peticiones SAML de Autenticación

En este apartado se detallan los atributos aceptados por el componente pasarela en peticiones SAML de solicitud de autenticación o inicio de sesión SSO.

8.2.1 Atributos de Personas Físicas

La pasarela Cl@ve solo identifica personas físicas y devuelve los atributos de la persona conectada que proporcione el IDP seleccionado; por tanto, el SP no es necesario que indique en la petición que atributos quiere.

A modo de documentación en la petición se pueden indicar los siguientes atributos:

- **http://eidas.europa.eu/attributes/naturalperson/CurrentFamilyName.** Devuelve el nombre del usuario. Se tiene que indicar que es obligatorio.
- **http://eidas.europa.eu/attributes/naturalperson/CurrentGivenName.** Devuelve los apellidos del usuario. Se tiene que indicar que es obligatorio.

- **http://eidas.europa.eu/attributes/naturalperson/PersonIdentifier.** Devuelve el identificador único (NIF) del Usuario. Se tiene que indicar que es obligatorio.
- **http://eidas.europa.eu/attributes/naturalperson/DateOfBirth**. Devuelve la fecha de nacimiento. Se tiene que indicar que es opcional, ya que solo lo devuelven ciertos IDP habilitados en la pasarela Cl@ve. En caso de indicar que es obligatorio, la pasarela Cl@ve genera un error.
- **http://es.minhafp.clave/NationalPersonIdentifier**. Devuelve el identificador único español para usuarios extranjeros. Se tiene que indicar que es opcional, ya que solo lo devuelve el IDP de eIDAS "Credencial europea". En caso de indicar que es obligatorio, la pasarela Cl@ve genera un error.

8.2.2 Deshabilitar Proveedores de Identificación en la Pasarela Cl@ve

El SP puede deshabilitar uno o varios IdPs para que no estén disponibles para la identificación del usuario, mediante la incorporación de diferentes atributos en la petición SAML enviada al componente pasarela, en los que se informa el atributo **isRequired** con el valor false (con el valor). Deshabilitar un IdP supondrá que no aparecerá en la Pasarela Cl@ve.

El atributo enviado en la petición SAML de este ejemplo para para deshabilitar el IdP "Certificado electrónico" es el siguiente:

```
<eidas:RequestedAttribute FriendlyName="AFirmaIdP"
Name="http://es.minhafp.clave/AFirmaIdP"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" isRequired="false"/>
```

Los atributos disponibles para deshabilitar el IdP correspondiente son los siguientes:

- Un atributo personal de tipo "http://es.minhafp.clave/AFirmaIdP" para deshabilitar el IdP "Certificado electrónico" en la ventana de selección de la pasarela.
- Un atributo personal de tipo "http://es.minhafp.clave/GISSIdP" para deshabilitar el IdP "Clave permanente" en la ventana de selección de la pasarela.
- Un atributo personal de tipo "http://es.minhafp.clave/AEATIdP" para deshabilitar el IdP "PIN24H" en la ventana de selección de la pasarela.
- Un atributo personal de tipo "http://es.minhafp.clave/EIDASIdP" para deshabilitar el IdP "Credencial europea" en la ventana de selección de la pasarela.
- Un atributo personal de tipo "http://es.minhafp.clave/CLVMOVILIdP" para deshabilitar el IdP "Móvil" en la ventana de selección de la pasarela.

8.2.3 Atributo RelayState

Es posible incluir dentro de la petición SAML un atributo personal de tipo "http://es.minhafp.clave/RelayState" que será devuelto en el SAML de respuesta con el mismo valor que se ha enviado en la petición.

Es similar al parámetro POST que se intercambia por fuera del token SAML con la diferencia de que este se devuelve firmado por la pasarela.

Este atributo puede ser usado por los SPs para cualquier tipo de comprobación que quieran realizar entre la petición y respuestas correspondientes o para persistir información en los ciclos asíncronos de autenticación.

8.2.4 Atributo ForceAuthn

El atributo **ForceAuthn** del tag **saml2p:AuthnRequest** indica que el SP solicita expresamente que se fuerce la autenticación en la pasarela Cl@ve. Pero este atributo se modifica en la pasarela Clave según la lógica de identificación y de SSO definida en la pasarela.

En la petición al IDP seleccionado, el atributo **ForceAuthn** indicará el siguiente valor:

- True, si sucede alguna de las siguientes situaciones:
  - El SP solicita expresamente que se fuerce la autenticación.
  - Es el primer proceso de autenticación solicitado para la sesión SSO activa.
  - El IdP establecido para completar el proceso ya ha sido empleado en alguna autenticación previa, almacenada en la sesión SSO activa, pero no cumple los requisitos de calidad exigidos para la credencial, o ha sido excluido en la petición.
- False, si se cumplen algunas de las siguientes condiciones:
  - El IdP establecido para completar el proceso ya ha sido empleado en alguna autenticación previa, almacenada en la sesión SSO activa, cumple los requisitos de calidad exigidos para la credencial, y no ha sido excluido en la petición.

8.3 Respuestas SAML de Autenticación

En este apartado se detallan los atributos que pueden ser retornados por el componente pasarela en respuestas SAML de solicitud de autenticación o inicio de sesión SSO de usuarios, previamente enviadas por los SPs.

8.3.1 Atributos Recibidos en Respuestas Correctas de Autenticación

Los atributos para personas físicas que retornará de forma obligatoria la pasarela a un proveedor de servicios son los siguientes:

- **http://eidas.europa.eu/attributes/naturalperson/CurrentFamilyName**[2]. Contiene los apellidos del usuario. Es un atributo personal obligatorio que se recibirá siempre.

[2] Es posible que, en alguno de los IdP, si la persona física a autenticar es extranjera, no devuelva este atributo. La información de "Nombre y Apellidos" aparecerá unificada en el atributo http://eidas.europa.eu/attributes/naturalperson/CurrentGivenName

- **http://eidas.europa.eu/attributes/naturalperson/CurrentGivenName**. Contiene el nombre del usuario. Se puede recibir tanto un nombre simple como uno compuesto. Es un atributo personal obligatorio que se recibirá siempre.
- **http://eidas.europa.eu/attributes/naturalperson/PersonIdentifier**. Contiene el identificador único (NIF) del Usuario. Es un atributo personal obligatorio que se recibirá siempre.
- **http://es.minhafp.clave/SelectedIdP**[3]. Identifica el proveedor de identidades utilizado para la identificación. Este atributo puede poseer alguno de los siguientes valores:
    - **"AFIRMA"**. IdP Certificado electrónico.
    - **"EIDAS"**. IdP Credencial europea.
    - **"IDP_MOVIL"**. IdP Cl@ve móvil.

No obstante, y con carácter opcional, podrán ser retornados además los siguientes atributos

- **http://es.minhafp.clave/FirstSurname**. Contiene el primer apellido del usuario. Es un atributo personal opcional que solo será retornado si es solicitado.
- **http://eidas.europa.eu/attributes/naturalperson/DateOfBirth**. Contiene la fecha de nacimiento. Solo retornado en la autenticación mediante DNI Electrónico, a través del IdP de @firma/Certificado digital o el IdP de EIDAS "Credencial Europea".
- **http://es.minhafp.clave/NationalPersonIdentifier**. Contiene el identificador único español para usuarios extranjeros. Solo retornado en la autenticación mediante el IDP de eIDAS "Credencial europea", si la petición del SP lo incluía.
- **http://es.minhafp.clave/RelayState**. En caso de ser informado en la petición, este parámetro presentará en la respuesta el mismo valor previamente enviado en la petición.
- **http://es.minhafp.clave/PartialAfirma**. Contiene la respuesta parcial de @firma en Base64, en la que se incluyen el mapeo de campos extraídos del certificado presentado para la identificación. Solo retornado en la autenticación mediante el IdP de @firma/Certificado electrónico.
- **http://es.minhafp.clave/RegisterType.** Indica el tipo de registro en el sistema indicado. Solo retornado en la autenticación mediante el IdP de PIN24H y Clave Permanente. Valores:
    - 2: CSV
    - 3: Certificado
    - 4: Presencial
    - 5: Videoidentificación
    - 6: Videoasistencia
- Otros datos proporcionados sobre la persona identificada por el proveedor de identidades, para los que esta otorgo su consentimiento de consulta y/o comunicación.

Un ejemplo de la información retornada en las respuestas SAML por Cl@ve sobre la persona identificada es la ilustrada mediante la siguiente imagen:

[3] Los valores que puede poseer este atributo son dinámicos, es decir, pueden cambiar con el tiempo debido a la administración de los IdPs en la Plataforma Cl@ve.

Ilustración 3: Atributos Respuesta SAML Retornada por el IdP "Certificados Electrónicos" (Captura de DemoSP).

8.3.1.1 Respuesta Parcial de @firma

Este atributo es incorporado a las repuestas SAML cuando la autenticación ha sido realizada mediante el IdP de certificados, incluyendo el mismo parte de la respuesta de validación de certificado retornada por la plataforma @firma. Con el objetivo de reducir el tamaño global del token SAML, en el atributo **PartialAfirma** se devuelven solamente dos nodos de toda la respuesta de @firma:

- El nodo **<dss:Result>** que contiene el resultado de la validación del certificado. Este nodo es redundante ya que es equivalente a la propia respuesta de Cl@ve. Pero se ha incluido porque en caso de resultar inválido, el *ResultMinor* puede resultar interesante para conocer la causa de la invalidez.
- El nodo **<afxp:ReadableCertificateInfo>** que contiene los mapeos propios de @firma y que puede utilizarse para obtener información adicional sobre el certificado:
    - Clasificación del certificado: persona física, persona jurídica, representante de las AA.PP. etc.
    - Razón Social para determinados tipos de certificado
    - OID o política del certificado para detectar si es un DNI,
    - Emisor del certificado,
    - Etc.

Los datos disponibles dentro del campo PartialAfirma están codificados en Base64, aunque su contenido es un fragmento XML, que incluye los nodos de la respuesta retornada por @firma especificados, agrupados por un nodo raíz, denominado "Partial_Afirma_Response", en el que se establecen los diferentes *namespaces* empleados por el resto de nodos. En los siguientes apartados se muestra un ejemplo del contenido de este atributo decodificado.

8.3.1.2 Certificados de Persona Física

Cl@ve solamente soporta certificados de persona física y certificados de persona jurídica anteriores al eIDAS y a extinguir.

Por tanto, Cl@ve solo devolverá respuesta válida para certificados con las siguientes clasificaciones.

- Clasificación = 0 – Persona física – certificado cualificado de firma
- Clasificación = 1 – Persona jurídica (no cualificado)
- Clasificación = 5 – Empleado Público
- Clasificación = 11 – Persona física representante ante las AAPP de persona jurídica
- Clasificación = 12 – Persona física representante ante las AAPP de entidad sin persona jurídica

Puede consultarse el documento "Tratamiento de Certificados en @firma" que puede encontrarse en el PAe en la dirección https://administracionelectronica.gob.es/ctt/afirma/descargas.

8.3.1.3 Certificados Emitidos por Prestadores Españoles

Cl@ve solamente soporta certificados emitidos por prestadores españoles. Por tanto, Cl@ve solo devolverá respuesta válida para certificados con el valor "ES" en el nodo "país".

Ejemplo de respuesta de @firma devuelta por Cl@ve en Base64:

```
<Partial_Afirma_Response xmlns:afxp="urn:afirma:dss:1.0:profile:XSS:schema"
xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema">
    <dss:Result>
        <dss:ResultMajor>urn:oasis:names:tc:dss:1.0:resultmajor:Success</dss:ResultMajor>
        <dss:ResultMinor>urn:oasis:names:tc:dss:1.0:profiles:XSS:resultminor:valid:certificate:Definitive</ds
s:ResultMinor>
        <dss:ResultMessage xml:lang="es">El certificado es válido</dss:ResultMessage>
    </dss:Result>
        <afxp:ReadableCertificateInfo>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>OrganizacionEmisora</afxp:FieldIdentity>
                        <afxp:FieldValue>Agencia Notarial de Certificacion S.L.U. - CIF
B83395988</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>versionPolitica</afxp:FieldIdentity>
                        <afxp:FieldValue>23</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>usoCertificado</afxp:FieldIdentity>
                        <afxp:FieldValue>digitalSignature | nonRepudiation | keyEncipherment |
dataEncipherment</afxp:FieldValue>
                </afxp:ReadableField>
```

```
                <afxp:ReadableField>
                        <afxp:FieldIdentity>pais</afxp:FieldIdentity>
                        <afxp:FieldValue>ES</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>subject</afxp:FieldIdentity>
                        <afxp:FieldValue>CN=JUAN EJEMPLO
ESPAÑOL,serialNumber=11111111H,givenName=JUAN,SN=EJEMPLO ESPAÑOL,T=DIRECTOR,OU=Certificado Corporativo
Personal,OU=DEPARTAMENTO DE RRHH,O=EMPRESA CON CONVENIO CCP S.A.,C=ES</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>certQualified</afxp:FieldIdentity>
                        <afxp:FieldValue>YES</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>numeroSerie</afxp:FieldIdentity>
                        <afxp:FieldValue>110065934850310895284456546275444347252</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>NombreApellidosResponsable</afxp:FieldIdentity>
                        <afxp:FieldValue>JUAN EJEMPLO ESPAÑOL</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>idPolitica</afxp:FieldIdentity>
                        <afxp:FieldValue>MITyC</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>tipoCertificado</afxp:FieldIdentity>
                        <afxp:FieldValue>ANCERT PF Corporativos Personales Firma 2010 SW</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>certClassification</afxp:FieldIdentity>
                        <afxp:FieldValue>ESIG</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>clasificacion</afxp:FieldIdentity>
                        <afxp:FieldValue>0</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>NIFResponsable</afxp:FieldIdentity>
                        <afxp:FieldValue>11111111H</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
```

```
                        <afxp:FieldIdentity>nombreResponsable</afxp:FieldIdentity>
                        <afxp:FieldValue>JUAN</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>unidadOrganizativa</afxp:FieldIdentity>
                        <afxp:FieldValue> OrganizationUnit=DEPARTAMENTO DE RRHH OrganizationUnit=Certificado
Corporativo Personal</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>ApellidosResponsable</afxp:FieldIdentity>
                        <afxp:FieldValue>EJEMPLO ESPAÑOL</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>segundoApellidoResponsable</afxp:FieldIdentity>
                        <afxp:FieldValue>ESPAÑOL</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>organizacion</afxp:FieldIdentity>
                        <afxp:FieldValue>EMPRESA CON CONVENIO CCP S.A.</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>primerApellidoResponsable</afxp:FieldIdentity>
                        <afxp:FieldValue>EJEMPLO</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>validoHasta</afxp:FieldIdentity>
                        <afxp:FieldValue>2019-05-04 sáb 10:12:17 +0200</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>validoDesde</afxp:FieldIdentity>
                        <afxp:FieldValue>2016-05-04 mié 10:12:17 +0200</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>email</afxp:FieldIdentity>
                        <afxp:FieldValue>no-reply@ancert.com</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>qscd</afxp:FieldIdentity>
                        <afxp:FieldValue>UNKNOWN</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>idEmisor</afxp:FieldIdentity>
```

```
                        <afxp:FieldValue>CN=ANCERT Corporativos Personales V2,O=Agencia Notarial de
Certificacion S.L.U. - CIF B83395988,L=Paseo del General Martinez Campos 46 6a planta 28010
Madrid,C=ES</afxp:FieldValue>
                </afxp:ReadableField>
                <afxp:ReadableField>
                        <afxp:FieldIdentity>politica</afxp:FieldIdentity>
                        <afxp:FieldValue>1.3.6.1.4.1.18920.2.1.1.2.4</afxp:FieldValue>
                </afxp:ReadableField>
        </afxp:ReadableCertificateInfo>
</Partial_Afirma_Response>
```

Por ejemplo:

- El OID, que devolvía la anterior pasarela Clave1, se recupera de: <afxp:ReadableField><afxp:FieldIdentity>politica</afxp:FieldIdentity><afxp:FieldValue>1.3.6.1.4.1.18920.2.1.1.2.4</afxp:FieldValue></afxp:ReadableField>
- El email, en caso que este informado en el certificado, se recupera de: <afxp:ReadableField><afxp:FieldIdentity>email</afxp:FieldIdentity><afxp:FieldValue>no-reply@ancert.com</afxp:FieldValue></afxp:ReadableField>

8.3.1.4 Atributo NationalPersonIdentifier

En el caso de que en la petición inicial del SP se haya solicitado el atributo ***NationalPersonIdentifier*** y la autentificación se realiza a través del nodo eIDAS, este nodo remite una solicitud al *Servicio de Identificación de ciudadanos eIDAS de la Agencia Tributaria* (AEAT) para recuperar, mediante la lógica interna de este servicio, el identificador nacional del ciudadano europeo (ver detalle en sección 6.4). Si el servicio de la AEAT recupera el identificador nacional del ciudadano europeo, la respuesta SAML que retorna la Pasarela Cl@ve incluirá el identificador nacional del ciudadano (DNI, NIE, NIF-L o NIF-M) dentro del atributo ***NationalPersonIdentifier***, proporcionado por el servicio de la AEAT, además del identificador europeo del ciudadano dentro del atributo ***PersonIdentifier***, proporcionado por el IdP transfronterizo junto con el nombre, apellidos y fecha de nacimiento. El atributo ***PersonIdentifier*** tiene el formato **XX/YY/ZZZZZZZZZZZZZ** (p.e. SE/ES/194911172296) con un máximo de 250 caracteres:

- **XX** es el código del país del IdP, el proveedor de la autenticación.
- **YY** es el código del país del SP, el solicitante de la autenticación, que siempre será ***ES*** en referencia a España.
- El campo **ZZZZZZZZZZZZZ** corresponde al identificador de la persona en su país.

Esta situación transfronteriza que ha involucrado a los nodos eIDAS se indica con el valor “EIDAS” en el atributo ***SelectedIdP*** de la respuesta SAML. Este caso se ilustra en el siguiente ejemplo:

```
<saml2p:Response [..]
                >
    <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">https://se-
pasarela.clave.gob.es</saml2:Issuer>
    <ds:Signature>[..] </ds:Signature>
    <saml2p:Status>
        <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
        <saml2p:StatusMessage>urn:oasis:names:tc:SAML:2.0:status:Success</saml2p:StatusMessage>
    </saml2p:Status>
    <saml2:Assertion ID="_OnDwOGIDBBgAKdz2opp.m2moH-VhSlnk.JG5R3qUOcAeyGwzer-hB3qQDRpBD8b"
                     IssueInstant="2023-12-01T11:28:48.390Z"
                     Version="2.0"
                     >
        <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">https://se-
pasarela.clave.gob.es</saml2:Issuer>
        <saml2:Subject>[..] </saml2:Subject>
        <saml2:Conditions NotBefore="2023-12-01T11:28:48.390Z"
                          NotOnOrAfter="2023-12-01T11:43:48.390Z"
                          >[..]
        </saml2:Conditions>
        <saml2:AuthnStatement AuthnInstant="2023-12-01T11:28:48.390Z">
            <saml2:AuthnContext>

<saml2:AuthnContextClassRef>http://eidas.europa.eu/LoA/substantial</saml2:AuthnContextClassRef>
                <saml2:AuthnContextDecl/>
            </saml2:AuthnContext>
        </saml2:AuthnStatement>
        <saml2:AttributeStatement>
            <saml2:Attribute FriendlyName="FirstName"
                             Name="http://eidas.europa.eu/attributes/naturalperson/CurrentGivenName"
                             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                             >
                <saml2:AttributeValue xmlns:eidas-
natural="http://eidas.europa.eu/attributes/naturalperson"
                                      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                      xsi:type="eidas-natural:CurrentGivenNameType"
                                      >Sven</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute FriendlyName="DateOfBirth"
                             Name="http://eidas.europa.eu/attributes/naturalperson/DateOfBirth"
                             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                             >
                <saml2:AttributeValue xmlns:eidas-
natural="http://eidas.europa.eu/attributes/naturalperson"
```

```
                                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                    xsi:type="eidas-natural:DateOfBirthType"
                                    >1949-11-17</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute FriendlyName="FamilyName"
                             Name="http://eidas.europa.eu/attributes/naturalperson/CurrentFamilyName"
                             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                             >
                <saml2:AttributeValue xmlns:eidas-
natural="http://eidas.europa.eu/attributes/naturalperson"
                                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                    xsi:type="eidas-natural:CurrentFamilyNameType"
                                    >Sturesson</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute FriendlyName="PersonIdentifier"
                             Name="http://eidas.europa.eu/attributes/naturalperson/PersonIdentifier"
                             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                             >
                <saml2:AttributeValue xmlns:eidas-
natural="http://eidas.europa.eu/attributes/naturalperson"
                                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                    xsi:type="eidas-natural:PersonIdentifierType"
                                    >SE/ES/194911172296</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute FriendlyName="NationalPersonIdentifier"
                             Name="http://es.minhafp.clave/NationalPersonIdentifier"
                             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                             >
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                    xsi:type="eidas-natural:string"
                                    >M0288517M</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute FriendlyName="SelectedIdP"
                             Name="http://es.minhafp.clave/SelectedIdP"
                             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                             >
                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                    xsi:type="xs:string"
                                    >EIDAS</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute FriendlyName="RelayState"
                             Name="http://es.minhafp.clave/RelayState"
                             NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
```

```
                              >
                <saml2:AttributeValue xmlns:eidas-
natural="http://eidas.europa.eu/attributes/naturalperson"
                                      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                      xsi:type="eidas-natural:PersonIdentifierType"
                                      >_gEHSNaU</saml2:AttributeValue>
            </saml2:Attribute>
        </saml2:AttributeStatement>
    </saml2:Assertion>
</saml2p:Response>
```

En caso de que el *Servicio de Identificación de ciudadanos eIDAS de la Agencia Tributaria* (AEAT) no recuperara el identificador nacional del ciudadano europeo, en la respuesta SAML sólo se incluirá el atributo ***PersonIdentifier***, no así el atributo ***NationalPersonIdentifier***. Si se ha incluido el atributo *NationalPersonIdentifier* en la petición SAML pero este no se incluye en la respuesta SAML, esta situación no se considera un caso de fallo de la autenticación ya que la autenticación ha sido proporcionada por el IdP transfronterizo, por lo que el campo **<saml2p:StatusCode>** será *Success* y en el campo **<saml2p:StatusMessage>** se indicará la causa que ha impedido proporcionar el *NationalPersonIdentifier* según lo reportado por la AEAT de entre uno de los siguientes motivos:

[-1] AEAT - Error técnico

[-1] AEAT - Petición cancelada

[2] AEAT - Error al asignar Id

[2] AEAT - Id introducido incorrecto

[2] AEAT - PersonIdentifier sin Id

```
<saml2p:Status>
       <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
       <saml2p:StatusMessage>[2] AEAT - PersonIdentifier sin Id /</saml2p:StatusMessage>
   </saml2p:Status>
```

8.3.2 Respuestas SAML de Error

En este apartado se recopilan los atributos retornados a los SPs en respuestas SAML de error por el componente pasarela. Estos son los siguientes:

- **http://es.minhafp.clave/ProcessId**. Identificador del proceso de autenticación.
- **http://es.minhafp.clave/SPRequest**. Petición SAML de autenticación realizada por el SP, codificada en base 64.

- **http://es.minhafp.clave/IDPResponses**. Respuesta SAML de autenticación enviada por los diferentes IdPs empleado en el proceso de autenticación, codificada en base 64 (opcional, si se logró obtener respuesta de algún IdP).
- **http://es.minhafp.clave/Trace**. Traza de error de la pasarela (opcional, en caso de producirse un error en la pasarela).
- **http://es.minhafp.clave/RelayState.** En caso de ser informado en la petición, este parámetro presentará en la respuesta el mismo valor previamente enviado en la petición.
- **http://es.minhafp.clave/SelectedIdP**. IdP seleccionado para autenticar al usuario (opcional, en caso de haber sido seleccionado alguno).
- **http://es.minhafp.clave/RemoteIP**. IP desde donde se realizó la petición SAML.
- **http://es.minhafp.clave/ForceAuth**. Indicador de autenticación requerida. Indicador de autenticación requerida (opcional, solo peticiones de autenticación).

En el anexo 12.3 se detalla la lista de posibles errores que genera la pasarela Cl@ve.

A continuación, se presenta un ejemplo de respuesta SAML de error retornada por el componente pasarela:

```
<saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:eidas="http://eidas.europa.eu/attributes/naturalperson"
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:xs="http://www.w3.org/2001/XMLSchema"
Consent="urn:oasis:names:tc:SAML:2.0:consent:obtained"
Destination="http://ESBCN01L41XGLQ2:8888/SP2/ReturnPage"
ID="_rmvXT97_Zsz5EbwGsNd_URZgQL2KsSAwoHua1SHltH34OZmdkBKQci8BKNn-s9m"
InResponseTo="_Uwj7ND3pNyqIijOeRONl.bz8l7g-hfJZ3p4jJglViJcylnRKjp2loHSogVsTJMB" IssueInstant="2019-10-
02T15:06:26.925Z" Version="2.0">

        <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">http://C-
PEPS.gov.xx</saml2:Issuer>

        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

                <ds:SignedInfo>

                        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

                        <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha512"/>

                        <ds:Reference URI="#_rmvXT97_Zsz5EbwGsNd_URZgQL2KsSAwoHua1SHltH34OZmdkBKQci8BKNn-
s9m">

                                <ds:Transforms>

                                        <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>

                                        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">

                                                <ec:InclusiveNamespaces
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="xs"/>

                                        </ds:Transform>

                                </ds:Transforms>

                                <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>


        <ds:DigestValue>GRFIij+kS2kkFfFuA82FFGA/Mkrc4Jho7kbLuhIWCGUB8biPZYT5zNa7lvtnBXNNy0DCNRLk4mqQyCSmbqpuE
A==</ds:DigestValue>

                        </ds:Reference>

                </ds:SignedInfo>
```

```
        <ds:SignatureValue>fOlK3Jv1+dMJ1NBACPPEjjNJqBvg4LZ2Dsx8foJ2sFLJk4mYspkkD2v/……..</ds:SignatureValue>
                <ds:KeyInfo>
                        <ds:X509Data>

        <ds:X509Certificate>MIIDQTCCAikCBFcx/LAwDQYJKoZIhvcNAQELBQAwZTELMAkGA1UEBhMCRVMxDzANBgNVBAgMBk1h
……../16YbScBXkFQO
oPwYYvawavhQovrRJVYfMaAwIxfStT1MlJeo9uJBSJnVkLS18dA6</ds:X509Certificate>
                        </ds:X509Data>
                </ds:KeyInfo>
        </ds:Signature>
        <saml2p:Status>
                <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Requester">
                        <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:RequestDenied"/>
                </saml2p:StatusCode>
                <saml2p:StatusMessage>200007 - sp.not.identified</saml2p:StatusMessage>
        </saml2p:Status>
        <saml2:Assertion ID="_klpOLU-YVHE4VtxEnBooda_tKyMUdkX9MA5f3IxXQbLWKjhL7E5reX-riz2yQ8S"
IssueInstant="2019-10-02T15:06:26.925Z" Version="2.0">
                <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">http://C-
PEPS.gov.xx</saml2:Issuer>
                <saml2:Subject>
                        <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
NameQualifier="http://C-PEPS.gov.xx">NotAvailable</saml2:NameID>
                        <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
                                <saml2:SubjectConfirmationData Address="192.168.56.1"
InResponseTo="_Uwj7ND3pNyqIijOeRONl.bz8l7g-hfJZ3p4jJglViJcylnRKjp2loHSogVsTJMB" NotOnOrAfter="2019-10-
02T15:11:26.925Z" Recipient="http://ESBCN01L41XGLQ2:8888/SP2/ReturnPage"/>
                        </saml2:SubjectConfirmation>
                </saml2:Subject>
                <saml2:Conditions NotBefore="2019-10-02T15:06:26.925Z" NotOnOrAfter="2019-10-
02T15:11:26.925Z">
                        <saml2:AudienceRestriction>
                                <saml2:Audience>http://ESBCN01L41XGLQ2:8888/SP2/ReturnPage</saml2:Audience>
                        </saml2:AudienceRestriction>
                </saml2:Conditions>
                <saml2:AuthnStatement AuthnInstant="2019-10-02T15:06:26.925Z">
                        <saml2:AuthnContext>

        <saml2:AuthnContextClassRef>http://eidas.europa.eu/LoA/low</saml2:AuthnContextClassRef>
                                <saml2:AuthnContextDecl/>
                        </saml2:AuthnContext>
                </saml2:AuthnStatement>
                <saml2:AttributeStatement>
                        <saml2:Attribute FriendlyName="ProcessId" Name="http://es.minhafp.clave/ProcessId"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
```

```
                                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">_4a46be0615e18471ee6428a7ae6aaf9a</saml2:AttributeValue>

                        </saml2:Attribute>

                        <saml2:Attribute FriendlyName="SPRequest" Name="http://es.minhafp.clave/SPRequest"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">

                                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">PHNhbWwycDpBdXRoblJlcXVlc3QgeG1sbnM6c2FtbDJwPSJ1cm46b2FzaXM6bmFtZXM6dGM6U0FNTDoyLjA6cHJv
dG9jb2wiIHhtbG5zOmRzPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwLzA5L3htbGRzaWcjIiB4bWxuczplaWRhcz0iaHR0cDovL2VpZGFzLmV1c
…….ENvbVlc3Q+</saml2:AttributeValue>

                        </saml2:Attribute>

                        <saml2:Attribute FriendlyName="Trace" Name="http://es.minhafp.clave/Trace"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">

                                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">eu.eidas.auth.commons.exceptions.InvalidParameterEIDASException: sp.not.identified&#13;

        at es.clave.sp.SpUtils.validateProviderName(SpUtils.java:129)&#13;

        at es.clave.proxy.ServletAction.doPost(ServletAction.java:148)&#13;

……..…..

                        </saml2:Attribute>

                        <saml2:Attribute FriendlyName="RemoteIP" Name="http://es.minhafp.clave/RemoteIP"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">

                                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">192.168.56.1</saml2:AttributeValue>

                        </saml2:Attribute>

                        <saml2:Attribute FriendlyName="ForceAuth" Name="http://es.minhafp.clave/ForceAuth"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">

                                <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">false</saml2:AttributeValue>

                        </saml2:Attribute>

                </saml2:AttributeStatement>

        </saml2:Assertion>

</saml2p:Response>
```

8.3.3 Datos Devueltos por Cl@ve en Caso de Certificados de Representante de Persona Jurídica (Clasificación 11 o 12)

Como se ha comentado en el apartado **8.3.1.2**, Cl@ve acepta certificados de persona jurídica con clasificación 11 y 12. El sistema siempre devolverá los datos asociados a la persona física, por lo que, en estos casos, devolverá obligatoriamente los datos del representante:

- Identificador de unicidad (serialnumber): Se trata de un identificador vinculado de manera única a una persona determinada, que permite asociar a la misma persona autenticaciones sucesivas. Se debe hacer notar que este identificador garantiza que no habrá dos personas con el mismo identificador, como es el caso del NIF español.
- Nombre (en general uno o varios nombres) (givenname)
- Apellido (en general los apellidos del ciudadano) (SN)

No obstante, los certificados de persona jurídica, cumpliendo la normativa ***ETSI EN 319 412-3 Electronic Signatures and Infrastructures (ESI); Certificate Profiles; Part 3: Certificate profile for certificates issued to legal persons***, deben incluir en el **Subject** la siguiente información:

- countryName (C): lugar en el que la persona jurídica está establecida. Clave sólo acepta countryname=ES
- organizationName (O); nombre con el que la persona jurídica está registrado
- organizationIdentifier (2.5.4.97); identificación de la organización, que será el CIF-NIF
- commonName (CN): nombre del representante

Si por necesidades del servicio fuera necesario conocer estos datos, y siempre que el IDP seleccionado sea Afirma con certificado digital con clasificación 11 o 12, se podrá encontrar esta información en el atributo PartialAfirma, incluido en la respuesta SAML que devuelve Cl@ve.

Ejemplo del atributo PartialAfirma:

```
<saml2:Attribute FriendlyName="PartialAfirma" Name=http://es.minhafp.clave/PartialAfirma
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">

<saml2:AttributeValue xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:type="xs:string">PFBhcnRpYWxfQWZpcm1hX1Jlc3BvbnNlIHhtbG5zOmFmeHA9InVybjphZmlybWE6ZHNzOjEuMDpwcm9maWxlO
lhTUzpzY2hlbWEiIHhtbG5zOmRzcz0idXJuOm9hc2lzOm5hbWVzOnRjOmRzczoxLjA6Y29yZTpzY2hlbWEiPjxkc3M6UmVzdWx0Pjxkc3M
6UmVzdWx0TWFqb3I+dXJuOm9hc2lzOm5hbWVzOnRjOmRzczoxLjA6cmVzdWx0bWFqb3I6U3VjY2VzczwvZHNzOlJlc3VsdE1ham9yPjxkc
3M6UmVzdWx0TWlub3I+dXJuOm9hc2lzOm5hbWVzOnRjOmRzczoxLjA6cHJvZmlsZXM6WFNTOnJlc3VsdG1pbm9yOnZhbGlkOmNlcnRpZml
jYXRlOkRlZmluaXRpdmU8L2RzczpSZXN1bHRNaW5vcj48ZHNzOlJlc3VsdE1lc3NhZ2UgeG1sOmxhbmc9ImVzIj5FbCBjZXJ0aWZpY2Fkb
yBlcyB2w6FsaWRvPC9kc3M6UmVzdWx0TWVzc2FnZT48L2RzczpSZXN1bHQ+PGFmeHA6UmVhZGFibGVDZXJ0aWZpY2F0ZUluZm8+PGFmeHA
6UmVhZGFibGVGaWVsZD48YWZ4cDpGaWVsZElkZW50aXR5Pm51bWVyb1Nlcmll PC9hZnhwOkZpZWxkSWRlbnRpdHk+PGFmeHA6RmllbGRWY
Wx1ZT4xNjYxMDAwMjE0NTQzNjMxOTU4OTE0MjkxNTQ0ODg1NjM4NzI4NjM8L2FmeHA6RmllbGRWYWx1ZT48L2FmeHA6UmVhZGFibGVGaWV
sZD48YWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOkZpZWxkSWRlbnRpdHk+c2VndW5kb0FwZWxsaWRvUmVzcG9uc2FibGU8L2FmeHA6Rmllb
GRJZGVudGl0eT48YWZ4cDpGaWVsZFZhbHVlPkFDVDwvYWZ4cDpGaWVsZFZhbHVlPjwvYWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOlJlYWR
hYmxlRmllbGQ+PGFmeHA6RmllbGRJZGVudGl0eT5wcmltZXJBcGVsbGlkb1Jlc3BvbnNhYmxlPC9hZnhwOkZpZWxkSWRlbnRpdHk+PGFme
HA6RmllbGRWYWx1ZT5GSUNUSUNJTzwvYWZ4cDpGaWVsZFZhbHVlPjwvYWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOlJlYWRhYmxlRmllbGQ
+PGFmeHA6RmllbGRJZGVudGl0eT5lbnRpZGFkU3VzY3JpcHRvcmE8L2FmeHA6RmllbGRJZGVudGl0eT48YWZ4cDpGaWVsZFZhbHVlPk1JT
klTVEVSSU8gREVMIElOVEVSSU9SPC9hZnhwOkZpZWxkVmFsdWU+PC9hZnhwOlJlYWRhYmxlRmllbGQ+PGFmeHA6UmVhZGFibGVGaWVsZD4
8YWZ4cDpGaWVsZElkZW50aXR5PnN1YmplY3Q8L2FmeHA6RmllbGRJZGVudGl0eT48YWZ4cDpGaWVsZFZhbHVlPkNOPUZVTkNJT05BUklPI
EZJQ1RJQ01PIEFDVCAtIEROSSA5OTk5OTAxOEQgKEFVVEVOVElDQUNJT04pLGdpdmVuTmFtZT1GVU5DSU9OQVJJTyxTTj1GSUNUSUNJTyB
BQ1QgLSBETkkgOTk5OTkwMThELHNlcmlhbE51bWJlcj1JRENFUy05OTk5OTAxOEQsT1U9QU1CSVRPIERFTCBDVUVSUE8gTkFDSU9OQUwgR
EUgUE9MSUNJQSxPVT1DRVJUSUZJQ0FETyBFTEVDVFJPTklDTyBERSBFTVBMRUFETyBQVUJMSUNPLE89TUlOSVNURVJJTyBERUwgSU5URVJ
JT1IsQz1FUzwvYWZ4cDpGaWVsZFZhbHVlPjwvYWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOlJlYWRhYmxlRmllbGQ+PGFmeHA6RmllbGRJZ
GVudGl0eT5pZEVtaXNvcjwvYWZ4cDpGaWVsZElkZW50aXR5PjxhZnhwOkZpZWxkVmFsdWU+Q049QUMgREdQIDAwNCwyLjUuNC45Nz0jMEM
wRjU2NDE1NDQ1NTMyRDUzMzIzODMxMzYzMDMxMzU0OCxPVT1DTlAsTz1ESVJFQ0NJT04gR0VORVJBTCBERSBMQSBQT0xJQ0lBLEM9RVM8L
2FmeHA6RmllbGRWYWx1ZT48L2FmeHA6UmVhZGFibGVGaWVsZD48YWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOkZpZWxkSWRlbnRpdHk+SUR
fZXVyb3BlbzwvYWZ4cDpGaWVsZElkZW50aXR5PjxhZnhwOkZpZWxkVmFsdWU+SURDRVMtOTk5OTkwMThEPC9hZnhwOkZpZWxkVmFsdWU+P
C9hZnhwOlJlYWRhYmxlRmllbGQ+PGFmeHA6UmVhZGFibGVGaWVsZD48YWZ4cDpGaWVsZElkZW50aXR5PnVuaWRhZE9yZ2FuaXphdGl2YTw
vYWZ4cDpGaWVsZElkZW50aXR5PjxhZnhwOkZpZWxkVmFsdWU+Q05QPC9hZnhwOkZpZWxkVmFsdWU+PC9hZnhwOlJlYWRhYmxlRmllbGQ+P
GFmeHA6UmVhZGFibGVGaWVsZD48YWZ4cDpGaWVsZElkZW50aXR5Pm51bWVyb0lkZW50aWZpY2FjaW9uUGVyc29uYWw8L2FmeHA6RmllbGR
JZGVudGl0eT48YWZ4cDpGaWVsZFZhbHVlPjk5OTk5OTwvYWZ4cDpGaWVsZFZhbHVlPjwvYWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOlJlY
WRhYmxlRmllbGQ+PGFmeHA6RmllbGRJZGVudGl0eT5wdWVzdG88L2FmeHA6RmllbGRJZGVudGl0eT48YWZ4cDpGaWVsZFZhbHVlPlNVQkl
OU1BFQ1RPUjwvYWZ4cDpGaWVsZFZhbHVlPjwvYWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOlJlYWRhYmxlRmllbGQ+PGFmeHA6RmllbGRJZ
GVudGl0eT5leHRlbnNpb25Vc29DZXJ0aWZpY2FkbzwvYWZ4cDpGaWVsZElkZW50aXR5PjxhZnhwOkZpZWxkVmFsdWU+S2V5UHVycG9zZUl
kIDA6ICBUTFMgV2ViIGNsaWVudCBhdXRoZW50aWNhdGlvbgpLZXlQdXJwb3NlSWQgMTogIEUtbWFpbCBwcm90ZWN0aW9uCktleVB1cnBvc
2VJZCAyOiAgMS4zLjYuMS40LjEuMzExLjIwLjIuMjwvYWZ4cDpGaWVsZFZhbHVlPjwvYWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOlJlYWR
hYmxlRmllbGQ+PGFmeHA6RmllbGRJZGVudGl0eT52ZXJzaW9uUG9saXRpY2E8L2FmeHA6RmllbGRJZGVudGl0eT48YWZ4cDpGaWVsZFZhb
HVlPjExMjwvYWZ4cDpGaWVsZFZhbHVlPjwvYWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOlJlYWRhYmxlRmllbGQ+PGFmeHA6RmllbGRJZGV
udGl0eT5OSUZSZXNwb25zYWJsZTwvYWZ4cDpGaWVsZElkZW50aXR5PjxhZnhwOkZpZWxkVmFsdWU+OTk5OTkwMThEPC9hZnhwOkZpZWxkV
mFsdWU+PC9hZnhwOlJlYWRhYmxlRmllbGQ+PGFmeHA6UmVhZGFibGVGaWVsZD48YWZ4cDpGaWVsZElkZW50aXR5Pm9yZ2FuaXphY2lvbjw
vYWZ4cDpGaWVsZElkZW50aXR5PjxhZnhwOkZpZWxkVmFsdWU+RElSRUNDSU9OIEdFTkVSQUwgREUgTEEgUE9MSUNJQTwvYWZ4cDpGaWVsZ
FZhbHVlPjwvYWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOlJlYWRhYmxlRmllbGQ+PGFmeHA6RmllbGRJZGVudGl0eT5Pcmdhbml6YWNpb25
FbWlzb3JhPC9hZnhwOkZpZWxkSWRlbnRpdHk+PGFmeHA6RmllbGRWYWx1ZT5ESVJFQ0NJT04gR0VORVJBTCBERSBMQSBQT0x0XJQ0lBPC9hZ
nhwOkZpZWxkVmFsdWU+PC9hZnhpOlJlYWRhYmxlRmllbGQ+PGFmeHA6RmllbGRWYWx1ZT48YWZ4cDpGaWVsZElkZW50aXR5PnRpcG9
DZXJ0aWZpY2FkbzwvYWZ4cDpGaWVsZElkZW50aXR5PjxhZnhwOkZpZWxkVmFsdWU+REdQIDAwNCBGVU5DSU9OQVJJTyBBVVRFRTlRJQ0FDS
U9OPC9hZnhwOkZpZWxkVmFsdWU+PC9hZnhpOlJlYWRhYmxlRmllbGQ+PGFmeHA6RmllbGRWYWx1ZT48YWZ4cDpGaWVsZElkZW50aXR
5PmVtYWlsPC9hZnhwOkZpZWxkSWRlbnRpdHk+PGFmeHA6RmllbGRWYWx1ZT5wa2lAcG9saWNpYS5lczwvYWZ4cDpGaWVsZFZhbHVlPjwvY
```

```
WZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOlJlYWRhYmxlRmllbGQ+PGFmeHA6RmllbGRJZGVudGl0eT52YWxpZG9EZXNkZTwvYWZ4cDpGaWV
sZElkZW50aXR5PjxhZnhwOkZpZWxkVmFsdWU+MjAxOC0xMC0zMCBtYXIgMTA6Mjg6MTcgKzAxMDA8L2FmeHA6RmllbGRWYWx1ZT48L2Fme
HA6UmVhZGFibGVGaWVsZD48YWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOkZpZWxkSWRlbnRpdHk+aWRQb2xpdGljYTwvYWZ4cDpGaWVsZEl
kZW50aXR5PjxhZnhwOkZpZWxkVmFsdWU+REVGQVVMVDwvYWZ4cDpGaWVsZFZhbHVlPjwvYWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOlJlY
WRhYmxlRmllbGQ+PGFmeHA6RmllbGRJZGVudGl0eT5BcGVsbGlkb3NSZXNwb25zYWJsZTwvYWZ4cDpGaWVsZElkZW50aXR5PjxhZnhwOkZ
pZWxkVmFsdWU+RklDVElDSU8gQUNUPC9hZnhwOkZpZWxkVmFsdWU+PC9hZnhwOlJlYWRhYmxlRmllbGQ+PGFmeHA6UmVhZGFibGVGaWVsZ
D48YWZ4cDpGaWVsZElkZW50aXR5PnZhbGlkb0hhc3RhPC9hZnhwOkZpZWxkSWRlbnRpdHk+PGFmeHA6RmllbGRWYWx1ZT4yMDIzLTEwLTM
wIGx1biAxMDoyODoxNyArMDEwMDwvYWZ4cDpGaWVsZFZhbHVlPjwvYWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOlJlYWRhYmxlRmllbGQ+P
GFmeHA6RmllbGRJZGVudGl0eT5PSV9FdXJvcGVvPC9hZnhwOkZpZWxkSWRlbnRpdHk+PGFmeHA6RmllbGRWYWx1ZT5WQVRFUy1TMjgxNjA
xNUg8L2FmeHA6RmllbGRWYWx1ZT48L2FmeHA6UmVhZGFibGVGaWVsZD48YWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOkZpZWxkSWRlbnRpd
Hk+dXNvQ2VydGlmaWNhZG88L2FmeHA6RmllbGRJZGVudGl0eT48YWZ4cDpGaWVsZFZhbHVlPmRpZ2l0YWxTaWduYXR1cmU8L2FmeHA6Rml
lbGRWYWx1ZT48L2FmeHA6UmVhZGFibGVGaWVsZD48YWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOkZpZWxkSWRlbnRpdHk+Tm9tYnJlQXBlb
GxpZG9zUmVzcG9uc2FibGU8L2FmeHA6RmllbGRJZGVudGl0eT48YWZ4cDpGaWVsZFZhbHVlPkZVTkNJT05BUklPIEZJQ1RJQ0lPIEFDVDw
vYWZ4cDpGaWVsZFZhbHVlPjwvYWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOlJlYWRhYmxlRmllbGQ+PGFmeHA6RmllbGRJZGVudGl0eT5wY
WlzPC9hZnhwOkZpZWxkSWRlbnRpdHk+PGFmeHA6RmllbGRWYWx1ZT5FUzwvYWZ4cDpGaWVsZFZhbHVlPjwvYWZ4cDpSZWFkYWJsZUZpZWx
kPjxhZnhwOlJlYWRhYmxlRmllbGQ+PGFmeHA6RmllbGRJZGVudGl0eT5wb2xpdGljYTwvYWZ4cDpGaWVsZElkZW50aXR5PjxhZnhwOkZpZ
WxkVmFsdWU+Mi4xNi43MjQuMS4yLjEuMTAyLjUyLDIuMTYuNzI0LjEuMy41LjcuMSwwLjQuMC4yMDQyLjEuMjwvYWZ4cDpGaWVsZFZhbHV
lPjwvYWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOlJlYWRhYmxlRmllbGQ+PGFmeHA6RmllbGRJZGVudGl0eT5ub21icmVSZXNwb25zYWJsZ
TwvYWZ4cDpGaWVsZElkZW50aXR5PjxhZnhwOkZpZWxkVmFsdWU+RlVOQ0lPTkFSSU88L2FmeHA6RmllbGRWYWx1ZT48L2FmeHA6UmVhZGF
ibGVGaWVsZD48YWZ4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOkZpZWxkSWRlbnRpdHk+TklGRW50aWRhZFN1c2NyaXB0b3JhPC9hZnhwOkZpZ
WxkSWRlbnRpdHk+PGFmeHA6RmllbGRWYWx1ZT5TMjgxNjAxNUg8L2FmeHA6RmllbGRWYWx1ZT48L2FmeHA6UmVhZGFibGVGaWVsZD48YWZ
4cDpSZWFkYWJsZUZpZWxkPjxhZnhwOkZpZWxkSWRlbnRpdHk+Y2xhc2lmaWNhY2lvbjwvYWZ4cDpGaWVsZElkZW50aXR5PjxhZnhwOkZpZ
WxkVmFsdWU+
NTwvYWZ4cDpGaWVsZFZhbHVlPjwvYWZ4cDpSZWFkYWJsZUZpZWxkPjwvYWZ4cDpSZWFkYWJsZUNlcnRpZmljYXRlSW5mbz48L1BhcnRpYW
xfQWZpcm1hX1Jlc3BvbnNlPg== </saml2:AttributeValue>

</saml2:Attribute>
```


Para ello, será necesario seleccionar dicho atributo, decodificarlo en base64 y buscar los parámetros indicados en el SUBJECT.

Ejemplo del asunto (subject) incluido en el PartialAfirma una vez decodificado en base 64:

```
<afxp:FieldIdentity>subject</afxp:FieldIdentity>
                                                              <afxp:FieldValue>C=ES,O=PRUEBA,2.5.4.97=VATES-
Q00000003J,CN=00000000T PRUEBASPF APELLIDOUNOPF (R: Q0000000J),SN=APELLIDOUNOPF
APELLIDODOSPF,givenName=PRUEBASPF,serialNumber=IDCES-00000000T,description=Ref:FNMT-RCM/FNMT-
RCM0009/PUESTO 1/11857/13122022123503</afxp:FieldValue>
```

Adicionalmente, para saber la clasificación que @Firma otorga al certificado, debe consultarse, también dentro del atributo *PartialAfirma*, la clasificación otorgada en el campo clasificación:

Ejemplo de clasificación del certificado dentro de incluido en el *PartialAfirma* una vez decodificado en base64:

```
<afxp:FieldIdentity>clasificacion</afxp:FieldIdentity>
<afxp:FieldValue>11</afxp:FieldValue>
```

9 Paquetes de Integración para los SP's

Existen tres paquetes de integración distintos acordes a las tecnologías que los SP's dispongan (Java, ASP.NET y PHP). Se detallan a continuación los elementos que componen cada uno y su procedimiento de implantación.

Un integrador debe utilizar el software de demo como un ejemplo funcional que permite utilizar la pasarela Cl@ve. Para una integración satisfactoria, es responsabilidad del SP integrar la funcionalidad en sus propios

flujos de negocio, incluyendo las adaptaciones que sean necesarias. Los procesos necesarios para adaptar el software a las necesidades específicas del SP quedan fuera del alcance del presente paquete de integración.

9.1 Paquete de Integración Java

Este paquete está asociado a la integración de aplicaciones J2EE y permitirá realizar ciclos de autenticación.

Requisitos del sistema mínimo para un correcto funcionamiento de la solución:

Concepto	Valores testados	Valores soportados
Maven	2.7	2.X 3.X
Versión de Java	1.8	1.7 o superior
Servidores de aplicaciones	Wildfly 10.1.Final Apache Tomcat 7.X	Wildfly 10.1.Final Apache Tomcat 7.X Apache Tomcat 8.X Glassfish 4.X Weblogic 12C
IDE	Eclipse	Eclipse Netbeans InetlliJ IDEA JDeveloper 11G

Otros requisitos requeridos para la implantación:

- Disponer de un certificado válido en formato PKCS#12 (extensiones *.p12 o *.pfx) y PKCS#8 (extensiones *.pem y *.der), expedido por una CA reconocida.
- Para entornos pre-productivos este certificado puede ser auto-firmado. Dado que el SP debe firmar los mensajes XMLs el certificado debe soportar esta funcionalidad.

La solución se basa en Maven, por tanto, las librerías y resto de clases necesarias estarán incluidas en las dependencias descritas en el fichero "***pom.xml***". En el paquete de integración se proveen los proyectos compilados de los artefactos que no están publicadas en los repositorios centrales de Maven.

Accediendo desde un IDE de desarrollo, debería de visualizarse algo similar a la siguiente captura de pantalla:

Ilustración 4: Paquete Integración Java - Visualización Proyecto en IDE

9.1.1 Estrategias de Integración

9.1.1.1 Despliegue Desde Cero

Se entregan unas librerías de Cl@ve v2.0, ya importadas en un ejemplo funcional, compiladas con Java 1.8 y preparadas como artefactos Maven.

- Con este ejemplo funcional se entregan diferentes Servlet de Spring para preparar una *SAMLRequest* y recibir una *SAMLResponse* utilizando las anteriores librerías.
- Es un ejemplo completo que no incluye ningún mecanismo de sesión más allá de mostrar los datos recibidos.
- Este ejemplo funcional consiste en un flujo web simple que en última instancia llama a la clase "ReturnAction.java".

9.1.1.2 Migración desde Cl@ve 1.0

Si lo que se desea es cambiar librerías en un SP prexistente, ya integrado con Cl@ve 1.0, el propio ejemplo constituye una referencia comentada sobre los nuevos objetos a manejar y los métodos que invocar para seguir el protocolo Cl@ve. En esta clase se indica:

- La instancia del motor SAML requiere unos ficheros de configuración nuevos, pero muy similares al modelo Stork del anterior paquete de integración, con un fichero SamlEngine.xml como fichero de configuración maestro que invoca a un fichero SignModule para el acceso al JKS de firma y validación y a un fichero de atributos soportados llamado "saml-engine-eidas-attributes.xml". Se instancia de forma similar a Stork.
- Dicha instancia pasa de ser "STORKSAMLEngine" a ser una instancia de la interfaz "ProtocolEngineNoMetadataI".
- Que para obtener el objeto que constituye una SAMLRequest legítima se deja de utilizar "STORKAuthnRequest" y se pasa a utilizar "IAuthenticationRequestNoMetadata".
- Que para manejar un objeto que constituye una SAMLResponse a enviar se deja de utilizar "STORKAuthnResponse" y se pasa a utilizar "IResponseMessageNoMetadata" .
- Que para validar una request SAML se deja de invocar el método "engine.validateSTORKAuthnRequest(byte[])" y se pasa a invocar el método "engine.unmarshallRequestAndValidate(byte[])".
- Que para firmar y cifrar una response SAML se deja de invocar el método "engine.generateSTORKAuthnResponse" y se pasa a invocar el método "engine.generateResponseMessage".

- El nuevo motor cifra la aserción que contiene los atributos personales del usuario. Es configurable firmar esta aserción siendo su valor por defecto, true.
- La lista de atributos disponibles utiliza nuevos constructores y deben ajustarse a un esquema descrito en un XML. Se provee de un ejemplo de inyección de atributos.

9.1.1.3 Pasos para la Migración

El kit de integración incluye el documento ***"Migración Kit Clave1 a Clave2.txt"*** que detalla pormenorizadamente los cambios que hay que realizar al kit de Cl@ve 1.0 para migrarlo a Cl@ve 2.0 y que funcione en este nuevo entorno. También se incluyen los diferentes parches *git* correspondientes a cada *commit* al realizar la migración.

Se puede usar como referencia para entender cómo adaptar a Cl@ve 2.0 las aplicaciones que fueron desplegadas para conectarse a Cl@ve 1.0 y que se basaron en el kit de integración de ese entorno.

9.1.1.4 Discontinuación del Kit de Cl@ve 1.0

Es importante recordar que con la publicación del paquete de integración de Cl@ve 2.0 se discontinúa y se deja de dar soporte al paquete de Cl@ve 1.0. Por tanto, si decide adaptarse un desarrollo basado en el kit de Cl@ve 1.0 será responsabilidad del organismo mantener actualizadas las librerías correspondientes. El kit de Cl@ve 1.0 estaba basado en Struts mientras que el de Cl@ve 2.0 está basado en Spring.

9.1.2 Notas Importantes para Conectar el Kit contra Clave2 Pasarela del Entorno de Servicios Estables (SE)

La configuración del presente Kit utiliza un certificado de prueba, a modo de ejemplo, para poder conectarse con Servicios Estables. Dicha configuración debe cambiarse por la que utilice el Organismo en Servicios Estables. Hay que tener en cuenta que todo Organismo que vaya a utilizar Clave debe realizar una solicitud de alta en el servicio, y que el certificado que vaya a utilizar debe de ser emitido para el Organismo. En otro caso podría pasar que varios Organismos utilizaran para sus accesos el certificado de serie añadido en los kits.

9.1.2.1 Configuración de Prueba de Serie

Certificado 'Config\Sello_Kit_Pruebas.p12': se encuentra importado (import key pair) dentro del 'KeyStore.jks'. Este certificado, que contiene parte pública y privada, permite conectar el Kit con la PASARELA de CLAVE2.

Certificado 'Config\sello_kit_de_pruebas_ac_sector_p_blico_.cer' debe estar importado (a través de WEBPORTAL de SERVICIOS ESTABLES) en la tabla de certificados del SP '21114293V_E04975701' (SP configurado en 'sp.properties'). Una vez cargado el certificado en el SP, hay que esperar a que se ejecute el demonio 'AutoDiscoverDaemon' que sincroniza los certificados des de la base de datos hacia los KeyStores. Actualmente dicho SP ya tiene configurado este certificado público.

Certificado 'Config\sello_entidad_sgad_pruebas.cer'. En el 'KeyStore.jks', se añade este certificado público, que permite confiar con el certificado que firma las peticiones SAML que vuelven de CLAVE2 PASARELA hacia el Kit. Es responsabilidad del organismo, actualizar este certificado en dicho KeyStore, con el certificado activo correspondiente en el momento de uso de este Kit.

9.1.3 Ficheros de Propiedades

La ruta hacia la carpeta en la cual residen los ficheros de configuración se declara en el fichero interno de propiedades ubicado en "SP\src\main\resources\spEnvironmentContext.xml".

El fichero de propiedades "sp.properties" contiene información de configuración como es el nombre del proveedor, la URL del módulo, la URL a la que se retornará la respuesta de la pasarela Cl@ve (después de realizar la petición SAML).

El fichero "***saml-engine-additional-attributes.xml***" permite incluir nuevos atributos soportados al esquema. No todos los atributos pueden ser recabados, ya que hace falta un IdP que pueda proveerlos.

9.1.4 Descripción de los Ficheros de Configuración de un SP

Para configurar el sistema se utilizan ficheros de configuración. Para definir las rutas de base donde buscar dichos ficheros se debe proveer la variable como un parámetro de sistema o un parámetro de Java:

- `CLAVE_SP2_CONFIG_REPOSITORY`. Ruta a la configuración del módulo. Por ejemplo {ruta}/Config/SP/

La implementación del motor presente en el módulo de integración actúa de una forma específica, permitiendo la escalabilidad de este aspecto. Por ejemplo, el sistema ya incluye una implementación para poder obtener datos de configuración:

- De un fichero maestro donde se apunta a los otros ficheros.
- Ficheros donde se define la configuración para firma. Suelen apuntar a un JKS.
- Ficheros donde se define la configuración para la capa de cifrado.
- Ficheros donde se definen aspectos del protocolo, como, por ejemplo, las validaciones habilitadas.
- Ficheros donde se define el set de atributos para atributos adicionales.
- Ficheros de tipo sp.properties donde se configuran URLS y que contienen el ProviderName empleado o una lista blanca de algoritmos de firma y cifrado.

Por lo tanto, existe una carpeta "***config***" se encuentra todos los ficheros de configuraciones que permiten ajustar el SP al sistema en el que vaya a implantarse.

Los cambios a realizar en los ficheros serán los siguientes:

SignModule_SP.xml:

- La etiqueta *<entry key="**keystorePath**">* debe tener la ruta en la que se vaya a guardar el almacén de certificados (donde estará el certificado privado de SP para realizar firmas y también los certificados públicos en los que se confía).
- La etiqueta <entry key="**keyStorePassword**"> hace referencia a la contraseña del almacén.
- La etiqueta <entry key="**keyPassword**"> hace referencia a la contraseña del certificado que se desea usar.
- La etiqueta <entry key="**issuer**"> hace referencia a los datos que deberá tener el certificado.
- La etiqueta <entry key="**serialNumber**"> hace referencia al número de serie del certificado.
- La etiqueta <entry key="**signature.algorithm**"> hace referencia a la URI del algoritmo de firma a utilizar y soportado por el certificado que se desea usar. El valor a establecer debe de estar permitido por la plataforma Pasarela.
  Para firmar la petición SAML con un certificado de tipo ECC se deberá indicar el algoritmo utilizado. Por ejemplo:
  ```
  <entry key="signature.algorithm">http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha384</entry>
  ```
  Es responsabilidad del propio integrador el validar que el algoritmo de curva elíptica configurado en esta etiqueta está disponible en el certificado a utilizar.
- La etiqueta <entry key="**signature.algorithm.whitelist**"> hace referencia a las URIs de algoritmos de firma que permite el SP al interpretar una respuesta SAML. Es decir, si la respuesta SAML no viene firmada con alguno de los algoritmos establecidos en esta etiqueta no se dará por válida la respuesta.

sp.properties:

- **provider.name:** Nombre identificativo de la aplicación a tres niveles de profundidad, es decir, indicando un nombre de institución, un sector dentro de la institución y una aplicación dentro del sector. Este nombre debe ser consensuado durante el proceso de alta.
- **sp.aplication**: Identifica a una aplicación como subconjunto dentro del SP.
- **service.url:** URL del servicio Cl@ve.
- **sp.return**: dirección en la que se espera la respuesta procedente de la pasarela Clave.

EncryptModule_SP.xml:

- La etiqueta *<entry key="**keystorePath**">* ha de cambiarse por la dirección hacia dónde esté el almacén de certificados.
- La etiqueta <entry key="**keyStorePassword**"> hace referencia a la contraseña del almacén.
- La etiqueta <entry key="**keyPassword**"> hace referencia a la contraseña del certificado que se desea usar.
- La etiqueta <entry key="**responseDecryptionIssuer**"> hace referencia a los datos del certificado del SP.
- La etiqueta <entry key="**serialNumber**"> hace referencia al número de serie del certificado.
- La etiqueta <entry key="**encryption.algorithm.whitelist**"> hace referencia a las URIs de algoritmos de cifrado que permite el SP al descifrar una respuesta SAML y que debe de contener los algoritmos de cifrado de datos utilizados por clave y detallados en el anexo 12.1.10. Es decir, si la respuesta

SAML no viene cifrada con alguno de los algoritmos establecidos en esta etiqueta no se dará por válida la respuesta. Este listado de algoritmos hace referencia al cifrado de datos que vienen en la respuesta SAML y no al cifrado de clave (KeyInfo).

Un SP que inicia el proceso de autenticación para solicitar una determinada información que no conoce, si hace uso de este Kit de Integración, sólo llevará a cabo proceso de descifrado cuando la respuesta de Pasarela viniera cifrada. Esto indica que el Kit de Integración no conlleva operaciones de cifrado, sino sólo de descifrado (si fuera necesario)

En resumen, para una integración solo es necesario tomar en cuenta los ficheros “sp.properties”, “EncryptModule_SP.xml” y “SignModule_SP.xml”.

9.1.5 Preparación del Entorno de Desarrollo

Las dependencias del proyecto están gestionadas mediante Maven. Por ello, existe un fichero descriptivo donde se indican los siguientes datos:

pom.xml:

- **artifactId**: Identificador con el que se reconocerá al SP. En función de este parámetro se generará el WAR, aunque es posible cambiar el nombre posteriormente antes de desplegarlo en el servidor de aplicaciones.
- **name**: nombre que recibe el propio SP.
- **parent-version**: versión del SP.
- **Description**: Descripción del propio SP.

Las dependencias que no están provistas en los repositorios centrales de Maven se proveen en la carpeta “Dependencias”. Dentro de esta carpeta se entrega el POM padre y las librerías compiladas y *mavenizadas*.

Los JAR están compilados empleando Java 1.8. En caso de que sea necesario compilarlos con otra versión de Java, se provee también el código fuente de los proyectos.

9.1.6 Certificados

La comunicación del módulo SP se basa en certificados electrónicos. Por lo tanto, la pasarela Cl@ve tiene que confiar en el certificado del SP y éste tiene que confiar en el certificado de la pasarela. El intercambio de certificados se produce durante un proceso de alta y registro previo.

En el fichero “SignModule_SP.xml” se configura la ruta del almacén de certificados, el cual tiene que contener el certificado privado que usará SP para realizar firmas, así como los certificados públicos en los que se confía.

En el fichero “EncryptModule_SP.xml” se configura la ruta del almacén de certificados, el cual tiene que contener el certificado privado que usará SP para realizar descifrados.

Requisitos en el apartado de certificados

Es necesario que el JKS disponga del certificado propio del servidor.

Dado que un SP debe comunicarse con la pasarela, la disponibilidad de este certificado en el contendor de certificados es un requisito para su correcta configuración.

En el caso de querer hacer uso de certificados de curva elíptica (ECC) no hay que hacer ningún tratamiento especial más allá de establecer los parámetros necesarios en los ficheros de configuración (SignModule_SP.xml) **y que el algoritmo de firma a utilizar con curva elíptica en el parámetro esté soportado por el propio certificado**.

Limitaciones de los certificados

Ha de destacarse que **no** podrá haber dos certificados en el almacén de certificados que contengan el mismo ***issuer y serial number*.**

CONSIDERACIONES ADICIONALES

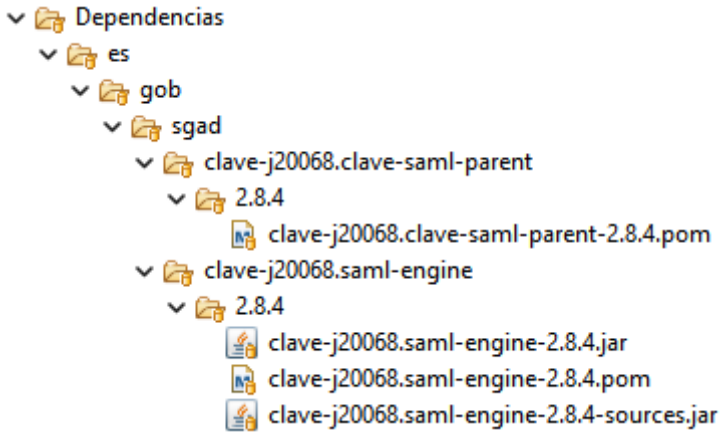
Los tickets SAML2 valen para un único ciclo. No deben reutilizarse. Si es necesario enviar una nueva solicitud, esta debe ser calculada de nuevo.

El sistema se basa en redirecciones del usuario que porta un ticket SAML2 firmado por la entidad que redirecciona y que debe validar la entidad que recibe la visita. Por lo tanto, sólo es necesario que cada elemento del sistema sea visible por parte del usuario, excluyendo la necesidad de que los servidores sean visibles entre sí.

9.1.7 Dependencias a Importar

Librerías entregadas en el directorio 'Dependencias' que se utilizan para compilar el integrador de Clave, y que no se pueden descargar desde el repositorio central de Maven. Algunas de estas librerías son necesarias para generar los tickets SAML:

- es.gob.sgad.clave-j20068.clave-saml-parent
- es.gob.sgad.clave-j20068.eidas-saml-engine



El proyecto de ejemplo se provee con un fichero "pom.xml" para organizar las dependencias mediante Maven. Al final del fichero se ha declarado un repositorio local que permite dejar automáticamente disponibles para Maven las dependencias en los BuildingBlocks de eIDAS v1.4.

Aquellas dependencias que no se encuentran en el repositorio central de Maven se proveen en el propio paquete de integración, en la carpeta /fuentes/sp/Dependencias, incluyendo su fichero POM. No obstante, es posible descargar el motor SAML de los Building Blocks de eIDAS disponibles en:

https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eIDAS-Node+-+All+releases

9.1.8 Exposición de Servicios

La funcionalidad del módulo actuando como SP implica la necesidad de servir una URL que se corresponde con un *servlet*. Se parte de la premisa de que ya existe una página web en el SP donde se deberá incluir un botón que comience el proceso de autenticación, por lo que se requiere una URL donde recibir la respuesta.

Durante un ciclo de autenticación se espera una respuesta de forma asíncrona, por lo que se expone una URL donde se espera recibir un ticket SAMLResponse firmado por la pasarela. Esta respuesta SAML tiene un atributo llamado "InResponseTo" que contiene el mismo valor que el atributo ID de la petición. De esta manera, se puede enlazar la respuesta con su origen.

Para servir esta URL, como se puede observar en el fichero "WEB-INF/web.xml", en el apartado "Servlet definition", se define un endpoint en "https://<dominio>/ReturnPage" que ejecuta el "doGet" y "doPost" de la clase "es.clave.sp.ReturnAction". Este método valida la respuesta SAML, la descifra y da como resultado una página JSP que contiene los atributos recibidos. Esta funcionalidad se comenta más adelante. A continuación, se puede observar la declaración del *servlet* utilizando Struts2.

```
<servlet>
   <description>ReturnPage</description>
   <display-name>ReturnPageServlet</display-name>
   <servlet-name>ReturnPageServlet</servlet-name>
```

```
    <servlet-class>es.clave.sp.actions.ReturnAction</servlet-class>
</servlet>
```

Posteriormente, se asocial el end-point con el servlet

```
<servlet-mapping>
        <servlet-name>ReturnPageServlet</servlet-name>
        <url-pattern>/ReturnPage</url-pattern>
</servlet-mapping>
```

En el ejemplo hay otros servlets adicionales para:

- PopulateIndexPage: Popular la página de inicio
- IndexPage: Firmar una petición SAML y enviarla a la pasarela.
- Capturar errores.

Una vez que se ha configurado la solicitud a enviar, un usuario pulsa sobre el botón enviar. Este botón envía el formulario a un *servlet* llamado "IndexPage". Como se puede observar en el fichero "WEB-INF/web.xml", se define un *endpoint* en "https://<dominio>/IndexPage" que ejecuta el "doPost" de la clase "es.clave.sp.IndexAction". Este método prepara una solicitud de autenticación SAML2 con el formulario recibido y escribe el resultado en el JSP llamado "redirect.jsp" que hace una redirección vía HTTP POST hacia la URL de la pasarela.

9.1.9 Generación de un Ticket SAML2

Véase el método "doGet" de la clase "sp.clave.sp.IndexAction". Para crear un ticket SAML2 empleando el motor SAML2 de eIDAS se requieren las siguientes llamadas:

En primer lugar, es necesario instanciar el motor SAML2. Para ello se ha de invocar estáticamente a la factoría incluida en el paquete de integración "SpProtocolEngineFactory" indicando como parámetro el nombre del módulo en cuestión. En este caso, el módulo se ha llamado SP.:

```
ProtocolEngineNoMetadataI protocolEngine = SpProtocolEngineFactory.getSpProtocolEngine("SPNoMetadata");
```

Para levantar la instancia del motor, en primer lugar, se va a cargar el fichero "SPSamlEngine.xml", y se va a buscar una instancia con el identificador indicado:

<instance name="SPNoMetadata">

A partir de este bloque de configuración se obtienen los nombres de los otros ficheros de configuración del motor SAML necesarios para la firma y el cifrado. En el paquete de integración estos ficheros de configuración se llaman "SignModule_SP.xml" y "EncryptModule_SP.xml".

Una vez que se dispone de una instancia del motor SAML2 se puede empezar a utilizar para diferentes cosas. Para solicitar la creación de una nueva SAMLRequest, primero hay que construir un objeto de tipo "EidasAuthenticationRequestNoMetadata" e introducir la configuración que se desea:

```
// Se genera la estructura de datos para configurar una nueva solicitud SAML
EidasAuthenticationRequestNoMetadata.Builder reqBuilder = new EidasAuthenticationRequestNoMetadata.Builder();
// Se indica la URL de destino (hacia dónde va dirigido el ticket)
reqBuilder.destination(nodeUrl);
// Se indica el id del SP, obtenido durante el proceso de alta. Será el DIR3_NIF;SPApplication
reqBuilder.providerName(providerName);
// Se incluye el array de atributos personales que se desean conocer
reqBuilder.requestedAttributes(reqAttrMap);
// Se indica el nivel mínimo de la credencial que puede utilizar el usuario
reqBuilder.levelOfAssuranceComparison(LevelOfAssuranceComparison.fromString("minimum").stringValue());
reqBuilder.levelOfAssurance(LevelOfAssurance.LOW.stringValue());
// Se indica el sector al que pertenece el SP, en clave siempre será public
reqBuilder.spType(SpType.PUBLIC.toString());
// Se calcula un ID-token aleatorio. No debe haber dos tickets distintos con el mismo ID.
reqBuilder.id(SAMLEngineUtils.generateNCName());
```

Una vez que se dispone de una solicitud configurada, se ha de proceder a firmarla y enviarla. Para ello se invoca al método siguiente de la instancia SAML2 disponible:

```
// Se firma la petición indicando los datos del destino
IRequestMessageNoMetadata bytes = protocolEngine.generateRequestMessage(reqBuilder.build(), true);
// Se extraen los bytes
byte[] ticket = bytes.getMessageBytes();
// Se codifican en Base64 para su envío
String samlRequest = EidasStringUtil.encodeToBase64(ticket);
```

Importante: la librería **clave-j20068.eidas-saml-engine** hace uso internamente del proveedor criptográfico *BouncyCastle* para la firma de la petición, por lo que es necesario que el integrador establezca la política

de configuración de este proveedor como mejor se adapte a su sistema tecnológico. En el caso de que no esté disponible este proveedor criptográfico en el momento de realizar la firma de la petición se puede forzar su inclusión mediante el método:

```
// Se fuerza hacer uso del proveedor criptográfico BC (BouncyCastle)
Security.addProvider(new BouncyCastleProvider());
```

Hay que tener en cuenta que al forzar el uso de este proveedor criptográfico puede afectar al comportamiento de otras aplicaciones que compartan el mismo servidor de aplicaciones.

Para incluir el atributo '**NationalPersonIdentifier**' en la generación de la petición SAML, se adjunta el siguiente código:

```
reqAttrMapBuilder.put(new
AttributeDefinition.Builder<String>().nameUri("http://es.minhafp.clave/NationalPerson
Identifier")
      .friendlyName("NationalPersonIdentifier")
      .personType(PersonType.NATURAL_PERSON)
      .required(false)
      .uniqueIdentifier(false)
      .xmlType("http://www.w3.org/2001/XMLSchema", "string",      "eidas-natural")
      .attributeValueMarshaller(new StringAttributeValueMarshaller())
      .build());
```

Para incluir el atributo '**CLVMOVILIdP**' en la generación de la petición SAML, se adjunta el siguiente código:

```
reqAttrMapBuilder.put(new
AttributeDefinition.Builder<String>().nameUri("http://es.minhafp.clave/CLVMOVILIdP")
      .friendlyName("CLVMOVILIdP")
      .personType(PersonType.NATURAL_PERSON)
      .required(false)
      .uniqueIdentifier(true)
      .xmlType("http://www.w3.org/2001/XMLSchema", "CLVMOVILIdP", "cl")
      .attributeValueMarshaller(new StringAttributeValueMarshaller())
      .build());
```

9.1.10 Validación de una Respuesta SAML2

Véase el método " doPost" de la clase "es.clave.sp.ReturnAction". Para validar un ticket SAML2 empleando el motor SAML se requiere el siguiente código.

En primer lugar, es necesario instanciar el motor SAML2. Para ello se ha de invocar estáticamente a la factoría incluida en el paquete de integración "SpProtocolEngineFactory" indicando como parámetro el nombre del módulo en cuestión. En este caso, el módulo se ha llamado SP:

```
ProtocolEngineNoMetadataI protocolEngine = SpProtocolEngineFactory.getSpProtocolEngine("SPNoMetadata");
```

Para levantar la instancia del motor, en primer lugar se va a cargar el fichero "SPSamlEngine.xml", y se va a buscar una instancia con el identificador indicado:

```
<instance name="SPNoMetadata">
```

El código para validar el ticket es el siguiente.

```
// Primero se extraen los bytes de la respuesta recibida
byte[] decSamlTicket = EidasStringUtil.decodeBytesFromBase64(SAMLResponse);
// Se lee y se valida
IAuthenticationResponseNoMetadata authnResponse;
try {
    // En caso de error durante la validación se produce una excepción
    authnResponse = engine.unmarshallResponseAndValidate(decSamlTicket,
request.getRemoteHost(), 0, 0, configs.getProperty(Constants.SP_RETURN ));
    // En caso de que se haya recibido una respuesta cifrada, se puede descifrar
inependientemente
    if (SPUtil.isEncryptedSamlResponse(decSamlTicket)) {
       byte[] eidasTicketSAML = engine.checkAndDecryptResponse(decSamlTicket);
       samlUnencryptedResponseXML =
SPUtil.extractAssertionAsString(EidasStringUtil.toString(eidasTicketSAML));
    }
} catch (EIDASSAMLEngineException e) {
    logger.error(e.getMessage(), e);
}
// Con el objeto obtenido se puede comprobar si es una autenticción exitosa y
extraer los datos recibidos.
if (authnResponse.isFailure()) {
    throw new ApplicationSpecificServiceException("Saml Response is fail",
authnResponse.getStatusMessage());
} else {
    attrMap = authnResponse.getAttributes().getAttributeMap();
}
```

9.1.10.1 Validación del Período de Validez del Ticket SAML

El código proporcionado valida el período de validez del ticket SAML si está informado. Es decir, valida que la fecha actual este entre los atributos NotBefore y NotOnOrAfter del tiquet SAML.

Para que esta validación sea efectiva el servidor del SP y el servidor de la plataforma Cl@ve tiene que estar sincronizados.

En caso de desfase en las horas, se pude indicar clockskew adecuado en la validación, en el 3r y 4ª parámetro del método **unmarshallResponseAndValidate**:

- 3r parámetro: skewclock al validar el NotBefore en milisegundos
- 4ª parámetro: skewclock al validar el NotOnOrAfter en milisegundos

Se acepta una tolerancia de entre 3 y 5 minutos.

Ejemplo

Si el servidor del SP se atrasa unos segundos, se debería indicar un numero negativo en milisegundos; por ejemplo, -60000 para un desfase de 1 minuto en el 3r parámetro:

```
engine.unmarshallResponseAndValidate(
        decSamlTicket
        , request.getRemoteHost()
        , -60000
        , 0
        , configs.getProperty(Constants.SP_RETURN ));
```

9.1.11 Generación de un Ticket SAML2 de Petición de Logout

Véase el método “generateLogout” de la clase “es.clave.sp.ReturnAction”. Para generar un ticket de logout empleando el motor SAML2 se requieren las siguientes llamadas:

En primer lugar, es necesario instanciar el motor SAML2. Para ello se ha de invocar estáticamente a la factoría incluida en el paquete de integración “SPUtil.getProtocolEngine()”.

defaultProtocolEngineFactory = new
ProtocolEngineFactoryNoMetadata(protocolEngineConfigurationFactory)

Para levantar la instancia del motor, en primer lugar, se va a cargar el fichero “IdPSamlEngine.xml”, y se va a buscar una instancia con el identificador indicado:

<instance name="SPNoMetadata">

Una vez que se dispone de una instancia del motor SAML2 se puede empezar a utilizar para diferentes cosas. Para solicitar la validación de una LogoutRequest, primero hay que construir un objeto de tipo "LogoutRequest" e introducir la configuración que se desea:

LogoutRequest authnRequest = null;

protocolEngine.generateLogoutRequestMessage(assertionConsumerUrl, providerName, destination, nonce)

Una vez que se dispone de una solicitud creada, se puede codificar en Base64 y proceder a su envío a la pasarela.

9.1.12 Validación de una Respuesta SAML2 de Logout

Véase el método "doPost" de la clase "es.clave.sp.ReturnAction". Para validar un ticket SAML2 empleando el motor SAML se requiere el siguiente código.

En primer lugar, es necesario instanciar el motor SAML2. Para ello se ha de invocar estáticamente a la factoría incluida en el paquete de integración

ProtocolEngineNoMetadataI protocolEngine = SpProtocolEngineFactory.getSpProtocolEngine(Constants.SP_CONF);

Para levantar la instancia del motor, en primer lugar, se va a cargar el fichero "IdPSamlEngine.xml", y se va a buscar una instancia con el identificador indicado:

<instance name="SPNoMetadata">

El código para validar el ticket es el siguiente:

```
LogoutResponse logoutResp = null;
try {
	byte[] decSamlToken = EidasStringUtil.decodeBytesFromBase64(logoutResponse);
	//validate SAML Token
	logoutResp = protocolEngine.unmarshallLogoutResponseAndValidate(decSamlToken,
                                        request.getRemoteHost(), 0, 0,
configs.getProperty(Constants.SP_RETURN));
} catch (Exception e) {
	logger.error("No se pudo validar la respuesta", e);
}
// Check session
String requestSamlId = SessionHolder.sessionsSAML.get(relayState);
```

```
if (requestSamlId == null || !requestSamlId.equals(logoutResp.getInResponseTo())) {
      throw new InvalidParameterEIDASException("La respuesta recibida no
corresponde con ninguna request o no coincide el RelayState");
}
```

La siguiente imagen ilustra los métodos que permiten recuperar información de la respuesta de cierre de sesión:

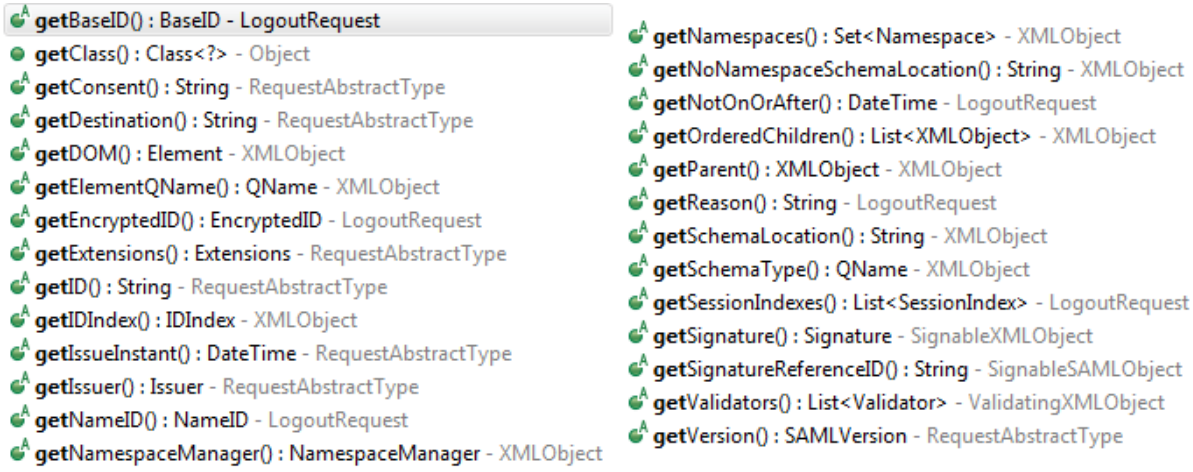


Ilustración 5: Paquete de Integración Java - Métodos de Acceso a Información a una Respuesta SAML2 de Logout

9.1.13 Guía de Integración de SP

Un SP puede integrarse fácilmente con cualquier servidor descrito en el apartado 9.1 del presente documento.

A modo de ejemplo y por facilidad de entendimiento usaremos un servidor Apache Tomcat v7.0.53 para la realización del procedimiento básico de configuración. Los pasos a realizar son:

- Instalación del certificado del SP e importación del mismo en el JKS, incluyendo la clave privada.

```
keytool -import -alias tomcat -keystore <your_keystore_filename> -file
<your_certificate_filename>
```

9.1.14 Información Adicional

Para extraer URLs de servicio y extraer el certificado en base al entorno a configurar:

- Servicios estables.
- Producción

Al igual que en el procedimiento anterior lo incluimos en el JKS.

```
keytool -import -alias tomcat -keystore <your_keystore_filename> -file
<your_certificate_filename>
```

- Configuración del puerto SSL del servidor de aplicaciones:

En nuestro caso de ejemplo se ha de modificar el fichero *server.xml* que se encuentra en la carpeta **conf** del servidor de aplicaciones.
El puerto a *configurar* es el 443. Los cambios más importantes son:

- Habilitar tráfico SSL sin autenticación de cliente.
- Indicar los ficheros de certificados correspondientes para el handshake SSL (hay que indicar el certificado con el que se identifica el servidor).
- El protocolo que maneja el tráfico entrante es *HTTP/1.1* y el esquema es HTTPS.

Para un servidor Tomcat, la configuración de los puertos quedaría más o menos así:

```
<Connector SSLEnabled="true" clientAuth="false" connectionTimeout="20000" keystoreFile=[…]
keystorePass=[…] keystoreType="JKS" port="443" protocol="HTTP/1.1" scheme="https"
secure="true" />
```

- Modificar los ficheros:
  - "pom.xml": con la información del servicio o despliegue "artifactId", y demás valores que requiere el fichero.
  - "SignModule_SP.xml", incluyendo los datos referentes a los certificados.
  - "EncryptModule_SP.xml": con los datos referentes a los certificados.
- Desplegar el fichero .war en la carpeta "*deployments*" correspondiente (en el caso de Tomcat, en "*/webapps*") y arrancar el servidor.

Para una solución existente que se quiere integrar se requiere:

- Disponer en el CLASS_PATH del servidor de aplicaciones los paquetes definidos en Dependencias a importar
- Incluir en el web.xml de la aplicación a integrar los servlets de acceso como se indica en Exposición de servicios.
- Hay que importar las funcionalidades definidas en Generación de un ticket SAML y Validación de una respuesta SAML.
- La aplicación a integrar tan solo debe incluir en su interfaz de usuario un único botón que inicie el proceso de autenticación. No es necesario importar la funcionalidad que hay en el ejemplo, que permite a un usuario configurar la SAMLRequest. En un escenario real, el propio SP es quien decide la configuración del ticket.

9.1.14.1 Proceso de Verificación

Se podrá comprobar el correcto funcionamiento del sistema (una vez se hayan configurado los parámetros anteriores), accediendo desde un navegador al nombre cualificado de su servicio: https://FQDN/SP, o en el caso de que disponga de un proxy inverso, compruébelo directamente accediendo al servidor y solicitando la página para localhost, https://localhost/SP2 (siempre que su servidor de aplicaciones escuche en este interfaz).

NOTA: El sistema se quejará del certificado, ya que no está expedido para localhost y es un certificado *autofirmado* sin valor como evidencia.

Verifique por favor la salida de logs de su servidor de aplicaciones, en busca de posibles errores de configuración.

9.1.15 Actualización Automática de los Certificados de Clave en los SPs

Desde el sistema CLAVE se permite distribuir, hacia los diferentes SP, los certificados (parte pública) que se usan para firmar sus mensajes.

En este apartado se dan todos los detalles técnicos y funcionales destacados para reducir la comunicación con los SP cuando se renuevan los certificados de CLAVE.

Desde el punto de vista funcional, los certificados, existentes en la base de datos de CLAVE, se obtiene a través de una URL que invocará un servlet y se devuelven a través de un XML en formato BASE64. Una vez *parseados* estos certificados se compararán con los ya existentes en el repositorio de certificados del Integrador, y utilizando un algoritmo se actualizarán o no.

El repositorio de certificados del Integrador debe ser una carpeta configurable, la cual contendrá un fichero de control utilizado por el algoritmo.

La ejecución de dicho algoritmo de recuperación y actualización de certificados para los SPs será periódica y a través de un proceso de tipo Daemon. Su intervalo de ejecución debe ser configurable y por defecto cada 24 horas.

Para que el integrador funcione, tiene que existir la siguiente variable de sistema dentro de las variables de entorno del sistema operativo:

- `CLAVE_SP2_CONFIG_REPOSITORY: Ruta donde se encuentran los ficheros de configuración del proyecto`

El único módulo 'SP2' existente en el proyecto del Integrador de Java 8, ya está provisto de Spring Framework, pero incorpora la siguiente configuración para que se escaneen las clases dentro del paquete 'es.clave.sp.daemon':

- `'applicationContext.xml'` → `'<context:component-scan base-package="es.clave.sp" />'`

- 'DaemonConfig.java' → Es una clase de configuración (tag '@Configuration') con la única finalidad de ser escaneada por el Spring e indicarle a través del tag '@EnableScheduling' que se pueda ejecutar el Daemon.

La clase 'DaemonCertificateProxy2.java' contiene el proceso demonio a ejecutar periódicamente (a través del tag Spring '@Scheduled') y que se encargará de conectarse al módulo 'Proxy2' de CLAVE a través de una URL y así descargarse los certificados públicos de firma y de cifrado. En un entorno local, por ejemplo, esta URL es:

'http://localhost:8888/Proxy2/Certificates' (sin seguridad)

'https://localhost:8443/Proxy2/Certificates' (con seguridad)

Este proceso demonio se ejecutará siempre y cuando la propiedad 'certProxy2.daemon.activated' del fichero 'certproxy2.properties' tenga el valor de 'true'.

En el fichero 'securityConf.xml' se incluye la propiedad 'certProxy2.daemon.fixedRate' que indica el tiempo en milisegundos de la periodicidad en que se debe ejecutar el demonio 'DaemonCertificateProxy2'. Por defecto se establece en que se ejecute una vez al día (86400000 ms).

9.1.15.1 Fichero de Configuración del Demonio 'certproxy2.properties'

El fichero de configuración 'certproxy2.properties' incorpora las siguientes propiedades:

```
# URL DE DONDE DESCARGAR LOS CERTIFICADOS
# LOC
#certproxy2.endpoint=http://localhost:8888/Proxy2/Certificates
certproxy2.endpoint=https://localhost:8443/Proxy2/Certificates
# INT
#certproxy2.endpoint=https://des-clave2.redsara.es/Proxy2/Certificates
# PRE
#certproxy2.endpoint=https://pre-pasarela.clave.gob.es/Proxy2/Certificates
# SE
#certproxy2.endpoint=https://se-pasarela.clave.gob.es/Proxy2/Certificates
# PRO
#certproxy2.endpoint=https://pasarela.clave.gob.es/Proxy2/Certificates

# PATH DEL KEYSTORE CON LOS SSL DE ACCESO. TIENE COMO PATH BASE LA CARPETA './Config/'
```

```
truststore.path=TrustStore.jks

# PASSWORD DEL KEYSTORE 'TrustStore.jks'
truststore.password=local-demo

# DIRECTORIO DONDE GUARDAR LOS CERTIFICADOS DESCARGADOS
certificates.path=C:/work/repos/_bitbucket_/clave/integradores/24981_clave_integra_desarrollo-de-nue/paquete_sp_java_8/Paquete_Integracion_Java_Clave_8/Certificates

# FLAG PARA ACTIVAR O DESACTIVAR EL NUEVO PROCESO: true / false
certProxy2.daemon.activated=true
```

En el demonio 'DaemonCertificateProxy2' se abre una conexión HTTP de tipo GET que consume el servlet ofrecido por el módulo 'Proxy2' de CLAVE2. La URL de esta conexión se configura en la propiedad 'certproxy2.endpoint' del fichero 'certproxy2.properties'.

A través de esta llamada obtenemos un XML con los diferentes certificados públicos existentes en la base de datos de CLAVE. Estos pueden estar activos (código 2) o inactivos (código 3). Básicamente se obtiene la siguiente información:

- tipo de certificado: de firma (signing), o de cifrado (encryption)
- nombre del certificado: 'proxy_sign_*' o 'proxy_cipher_*'
- contenido del certificado en Base64
- estado certificado: activo / inactivo

9.1.15.2 XML de Certificados Devuelto por el servlet de PROXY2 / CLAVE2

A continuación, se muestra un ejemplo de XML de certificados devuelto por el servlet de PROXY2 / CLAVE2 a través de la llamada a la URL 'https://localhost:8443/Proxy2/Certificates':

```
<md:EntityDescriptor entityID="https://localhost:8443/Proxy2/Certificates">
      <md:IDPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
         <md:KeyDescriptor use="signing">
            <ds:KeyInfo>
               <ds:KeyName>proxy_sign_58764336</ds:KeyName>
               <ds:X509Data>
                  <ds:X509Certificate>
                     MIIDQTCCAikCBFcx/LAwDQYJKoZIhvcNAQELBQAwZTELMAkGA1UEBhz...
                  </ds:X509Certificate>
               </ds:X509Data>
               <ds:MgmtData>active</ds:MgmtData>
            </ds:KeyInfo>
```

```
        </md:KeyDescriptor>
        <md:KeyDescriptor use="signing">
           <ds:KeyInfo>
              <ds:KeyName>proxy_sign_5731fcb0</ds:KeyName>
              <ds:X509Data>
                 <ds:X509Certificate>
                    MIIDQTCCAikCBFcx/LAwDQYJKoZIhvcNAQELBQAwZTELMAkGA1UEBhMCRV...
                 </ds:X509Certificate>
              </ds:X509Data>
              <ds:MgmtData>inactive</ds:MgmtData>
           </ds:KeyInfo>
        </md:KeyDescriptor>
        <md:KeyDescriptor use="encryption">
           <ds:KeyInfo>
              <ds:KeyName>proxy_cipher_5731fcb0</ds:KeyName>
              <ds:X509Data>
                  <ds:X509Certificate>
                    MIIDQTCCAikCBFcx/LAwDQYJKoZIhvcNAQELBQAwZT...
                 </ds:X509Certificate>
              </ds:X509Data>
              <ds:MgmtData>active</ds:MgmtData>
           </ds:KeyInfo>
        </md:KeyDescriptor>
        <md:KeyDescriptor use="encryption">
           <ds:KeyInfo>
              <ds:KeyName>proxy_cipher_58764336</ds:KeyName>
              <ds:X509Data>
                 <ds:X509Certificate>
                    MIIDQTCCAikCBFcx/LAwDQYJKoZIhvcNAQELBQAwZTELMAk...
                 </ds:X509Certificate>
              </ds:X509Data>
              <ds:MgmtData>inactive</ds:MgmtData>
           </ds:KeyInfo>
        </md:KeyDescriptor>
        <md:Organization>
           <md:OrganizationName xml:lang="en">
              General Secretariat of Digital Administration - Ministry of Territorial Policy and Civil
Service
           </md:OrganizationName>
           <md:OrganizationDisplayName xml:lang="en">
              General Secretariat of Digital Administration - Ministry of Territorial Policy and Civil
Service
           </md:OrganizationDisplayName>
```

```
            <md:OrganizationURL xml:lang="en">
               https://administracionelectronica.gob.es/pae_Home/pae_Organizacion/SGAD.html
            </md:OrganizationURL>
         </md:Organization>
      </md:IDPSSODescriptor>
   </md:EntityDescriptor>
```

9.1.15.3 Algoritmo del Proceso de Recuperación y Actualización de Certificados

En la clase 'DaemonCertificateProxy2' se incorpora el algoritmo de actualización y recuperación de certificados públicos. Su funcionamiento es el siguiente:

1) Por una parte, se recuperan los pares atributo/valor del fichero 'certificates.properties'. Este fichero se encuentra en el path de certificados.
2) Por otra parte, se hace la llamada tipo GET al servlet de PROXY2 de CLAVE que devuelve un XML con los certificados públicos de firma y de cifrado. La URL a llamar está indicada en la variable 'certproxy2.endpoint' del fichero 'certproxy2.properties'.
3) Se parsea el XML devuelto a través de la librería DOM Parser para obtener el tipo de certificado (firma o cifrado), el nombre, el estado (activo o inactivo), y el certificado en BASE64.
4) Si el nombre del certificado obtenido a través del XML existe en el fichero de propiedades 'certificates.properties'...
    a. Si el estado del certificado obtenido es igual al del fichero de propiedades...
        i. Se informa en el log que el certificado de firma o de cifrado ya existe.
    b. Si el estado del certificado obtenido es diferente al del fichero de propiedades...
        i. Se descarga el certificado en el repositorio local de certificados (se sobrescribe), y se actualiza la información del certificado en el fichero 'certificates.properties' con el nombre, el estado y la fecha de actualización.
        ii. Se informa en el log que se ha actualizado el certificado de firma.
5) Si el nombre del certificado obtenido a traves del XML no existe en el fichero de propiedades 'certificates.properties'...
    a. Se descarga el certificado en el repositorio local de certificados, y se inserta la info del certificado en el fichero 'certificates.properties' con el nombre, el estado y la fecha de actualización.
    b. Se informa en el log que se ha descargado el certificado de firma.

9.1.15.4 Almacén de Certificados de Confianza "TrustStore.jks"

En el demonio 'DaemonCertificateProxy2' se ha añadido la parte de código que permite hacer llamadas seguras al módulo Proxy2 de CLAVE2 a través de una URL de tipo 'https'. Para este tipo de llamadas se

necesita de un solo almacén de certificados de confianza 'TrustStore.jks', que contiene los certificados públicos de todos los entornos (LOC, INT, PRE, SE y PRO), los cuales se han descargado directamente del navegador a través de las diferentes URLs (caso LOC: 'https://localhost:8443/Proxy2/Certificates').

Para configurar este almacén de confianza, hay dos parámetros en el fichero de propiedades 'Config\certproxy2.properties':

```
# Path del keystore con los SSL de acceso. Tiene como path base la carpeta '.../Config/'
truststore.path=TrustStore.jks
# Password del keystore 'TrustStore.jks'
truststore.password=local-demo
```

9.2 Paquete de Integración C# y ASP.NET

Los requisitos técnicos requeridos para poder implementar la integración para la tecnología .NET son:

- .Net Framework:
  - Disponer de una implementación que soporte el algoritmo de firma RSA-SHA512.
  - El kit se distribuye para la versión 4.7.2 del framework. No se permiten versiones anteriores.
- Un entorno de desarrollo que use SSL; es decir, https://localhost/xxx
- Un IDE para el desarrollo. Por ejemplo, Visual Studio 2017. El IDE incluye un IISExpress para probar el despliegue.

Para comenzar, hay que abrir la solución del proyecto en: ***source\repos\DemoSPSaml2\DemoSPSaml2.Net.sln***.

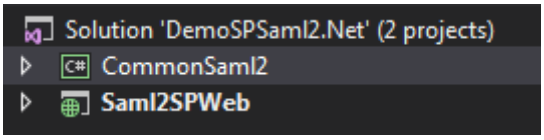


Ilustración 6: Paquete de Integración C# - Fuentes

- ***CommonSaml2*** es una librería que expone un API estático con los métodos necesarios para la lógica del programa (métodos de cifrado, cálculo de firmas, etc.).
- ***Saml2SPWeb*** contiene la parte Web, fundamentalmente el flujo Web para realizar un ciclo de autenticación completo (desde que el usuario se introduce en la página web en su navegador hasta que recibe toda la información desde el IdP). Se trata de una implementación de ejemplo, siendo necesario que el proveedor de servicios lo adecue a su implementación propia. **Nota**: Se ha hecho uso del IIS Express y NuGET para el uso de este proyecto.

En este proyecto existen tres apartados destacados:

- **SP.aspx**: Vista y código de cómo se genera un ticket (permite cambiar la forma de autenticación, el valor del LoA…).

- **ReturnPage.aspx** y **ResponseHandler.cs**: Código que valida la respuesta de la pasarela Cl@ve y redirige al usuario, si todo ha ido bien, a SuccesPage.aspx.
- **SuccesPage.aspx**: tiene como finalidad mostrar los datos propios del usuario y permitir al usuario cerrar sesión, emitiendo otro ticket SAML2, el cual, si todo ha ido bien, le llevará a la vista SP.aspx.

9.2.1 Versión del Framework y Persistencia de la Sesión

Se debe comprobar que se está usando la versión de Framework 4.7.2 en todos los proyectos, para que no aparezcan incompatibilidades en el desarrollo, para ello se ha de hacer *clic derecho* sobre las soluciones, *propiedades/aplicación/plataforma de destino.*

Se debe usar framework 4.7.2 o superior para poder indicar que la cookie de sesión tiene las propiedades:

- SameSite a None
- Secure

El motivo es que la integración con Clave2 realiza redirecciones entre dominios y es necesario mantener la sesión de ASP.Net entre la request y la reponse.

Ver:

https://docs.microsoft.com/en-us/aspnet/samesite/system-web-samesite
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie/SameSite

9.2.2 Notas Importantes para Conectar el Kit contra Clave2 Pasarela del Entorno de Servicios Estables (SE)

La configuración del presente Kit utiliza un certificado de prueba, a modo de ejemplo, para poder conectarse con Servicios Estables. Dicha configuración debe cambiarse por la que utilice el Organismo en Servicios Estables. Hay que tener en cuenta que todo Organismo que vaya a utilizar Clave debe realizar una solicitud de alta en el servicio, y que el certificado que vaya a utilizar debe de ser emitido para el Organismo. En otro caso podría pasar que varios Organismos utilizaran para sus accesos el certificado de serie añadido en los kits.

9.2.2.1 Configuración de Prueba de Serie

El certificado 'Saml2SPWeb\App_Data\Sello_Kit_Pruebas.p12' se encuentra importado (import key pair) dentro del contenedor de certificados 'Firma.p12' con la contraseña 'changeit'. Este certificado, que contiene parte pública y privada, permite conectar el Kit con la PASARELA de CLAVE2.

El certificado 'Saml2SPWeb\App_Data\sello_kit_de_pruebas_ac_sector_p_blico_.cer' debe estar importado (a través de WEBPORTAL de SERVICIOS ESTABLES) en la tabla de certificados del SP '21114293V_E04975701' (SP configurado en 'Saml2SPWeb\App_Data\properties.xml'). Una vez cargado el certificado en el SP, hay

que esperar a que se ejecute el daemon 'AutoDiscoverDaemon' que sincroniza los certificados des de la BDD hacia los KeyStores. Actualmente dicho SP ya tiene configurado este certificado público.

El certificado 'Saml2SPWeb\App_Data\certificado_respuesta_clave-se.cer' es posible importarlo manualmente en el almacén de "Personas de Confianza" del sistema (Certificates\Current User\Trusted People\Certificates). Este certificado es el que devuelve Cl@ve-SE en sus respuestas SAML.

9.2.3 Fichero de Propiedades

Para la demo se ha creado un manejador de la configuración llamado *CommonSaml2/Conf.cs,* el cual contiene las definiciones de los parámetros de configuración. Los valores de estos parámetros de configuración se encuentran en "Saml2SPWeb/APP_DATA/properties.xml".

Este fichero es el lugar donde un integrador debe centrar su atención, pues permite indicar los parámetros de funcionamiento:

```
<properties
    url_clave = "https://se-pasarela-clave.gob.es/Proxy2/ServiceProvider"
    url_redirect_method = "post"
    provider_name = "DEMO-SP"
    dir_p12_signing = "~/App_Data/Firma.p12"
    cert_password_sign = "changeit"
    dir_log = "eidas-error-log.txt"
    WantAssertionsSigned = "false"
    ClockSkewHours = "10"
  >
```

9.2.4 Certificados

Es requisito disponer de un certificado X509 con un par de claves que se usará para el descifrado asimétrico y para las tareas de firma en formato PKCS12 (.p12 o .pfx).

En el fichero properties.xml se encuentran los parámetros que contienen el path hacia estos ficheros disponibles en "Saml2SPWeb/App_Data":

```
    dir_p12_signing = "~/App_Data/Firma.p12"
    cert_password_sign = "changeit"
```

Cabe destacar, que los certificados de confianza serán almacenados en el almacén de certificados propio de Windows dedicado a personas de confianza, sólo disponibles para el usuario actual. En caso de que no se

confíe en un certificado, se mostrará un mensaje de advertencia que permitirá incluir el certificado de forma permanente.

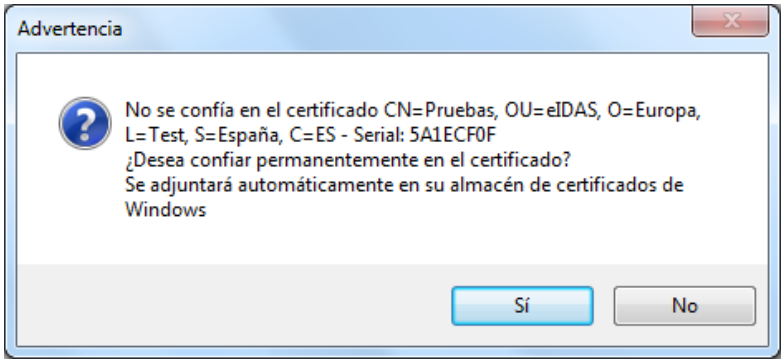


Ilustración 7: Instalación de Certificados en Almacén Windows de Personas - Mensaje de Advertencia

9.2.5 Dependencias a Importar

En cada solución hay un apartado denominado referencias. Ahí se encuentras todas las librerías del proyecto.

Las dependencias se administran mediante el sistema ***Nuget***, y la forma de hacerlo es la siguiente:

Para descargar las dependencias, en Visual Studio se puede lanzar una compilación (Proyecto/compilar) o se puede hacer clic con el botón secundario en cada proyecto. En este caso se abrirá un menú contextual en el cual se puede escoger la opción *Administrar paquetes nuget...* Posteriormente, se abrirá una ventana y en la opción de *Examinar* habrá un buscador. Tras esta tarea, se puede pulsar en el icono de descargar. Esto instalará las dependencias de forma automática en el proyecto seleccionado.

Se debe incluir en este destino las siguientes librerías o descargarlas en el caso que no se disponga ya de las mismas en el apartado de referencias.

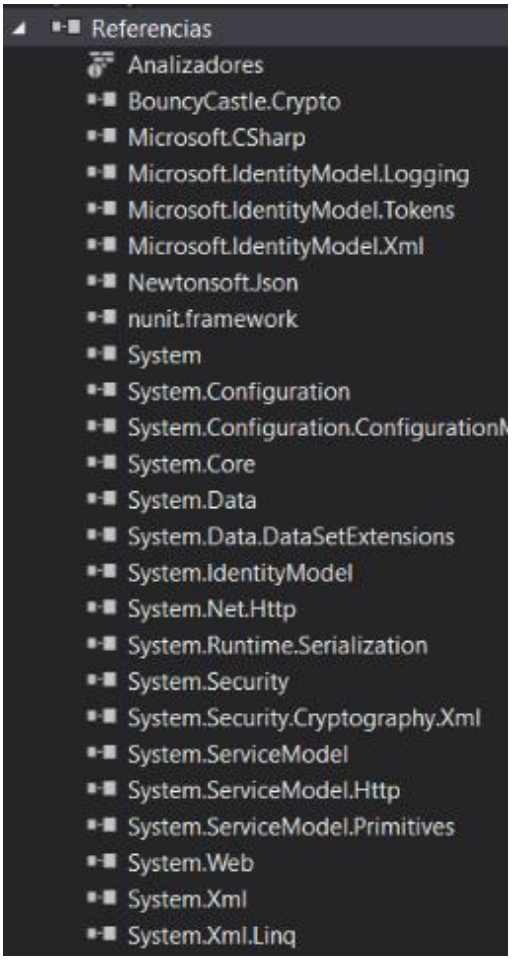


Ilustración 8: Paquete de integración C# - Dependencias

9.2.6 Generación de un Ticket SAML

Accediendo a

http(s)://dominio:puerto/SP.aspx

se mostrará una ventana con siete opciones:

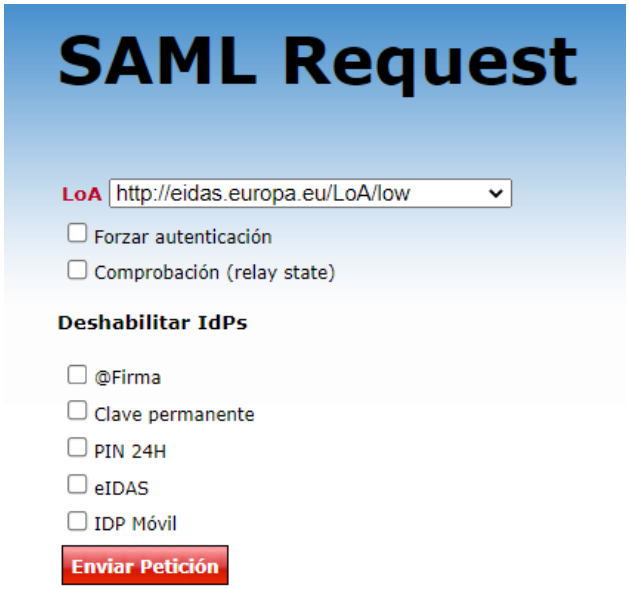


Ilustración 9: Paquete de Integración C# - Demo SP (Generación de Petición SAML)

Hay dos formas de crear un ticket SAML:

Se puede crear uno predeterminado dando en el botón enviar, en el cual te da todas las opciones disponibles para la autenticación del usuario, y otro personalizado de manera que es el mismo usuario el que elije cómo desea autenticarse etc.

El método se encuentra en DemoSPSaml2.Net/ CommonSaml2/ XMLData.cs y son **Saml2RequestDefault.**

Saml2RequestDefault recibe una serie de parámetros que provienen de Conf.cs. Si en la vista SP.aspx se cambiara alguno, serían sustituidos del ticket SAML los usados por defecto por los estos segundos.

El parámetro que puede ser modificado es newLOA (para cambiar el valor de LoA por low, substancial o high). El resto se usan como Flag para enviar o no un atributo en el ticket SAML2. Estos atributos son todos los Boolean.

El array *values* y el string *urlService* deben de salir de la configuración, ya que siempre han de ser los mismos.

9.2.7 Código Saml2RequestDefault

```
public static XmlDocument Saml2RequestDefault(string[] values, string urlService, Boolean
afirma, Boolean clavePermanente, Boolean pin24h, Boolean eidas, Boolean auth, Boolean
relaystate, string newLOA)
        {
            XmlDocument doc = new XmlDocument();
            doc.PreserveWhitespace = false;
```

```
            try
            {
                XmlNode docNode = doc.CreateXmlDeclaration("1.0", "UTF-8", null);
                doc.AppendChild(docNode);
                XmlNode samlreq = doc.CreateElement("saml2p", "AuthnRequest",
"urn:oasis:names:tc:SAML:2.0:protocol");

                XmlAttribute destination = doc.CreateAttribute("Destination");
                destination.Value = urlService;
                samlreq.Attributes.Append(destination);
                //Adding ProviderName
                XmlAttribute providerName = doc.CreateAttribute("ProviderName");
                providerName.Value = Conf.get_provider_name();
                samlreq.Attributes.Append(providerName);
                //Adding a random ID
                XmlAttribute id = doc.CreateAttribute("ID");
                id.Value = "_" + Guid.NewGuid().ToString();
                samlreq.Attributes.Append(id);
                //Adding the hour and date
                XmlAttribute issueInstant = doc.CreateAttribute("IssueInstant");
                issueInstant.Value = Utils.GetDate();
                samlreq.Attributes.Append(issueInstant);
                //Adding SP return URL
                XmlAttribute assertionConsumerURL =
doc.CreateAttribute("AssertionConsumerServiceURL");
                assertionConsumerURL.Value = Conf.get_url_return_response();
                samlreq.Attributes.Append(assertionConsumerURL);

                XmlAttribute attribute0 = doc.CreateAttribute("xmlns", "saml2p",
"http://www.w3.org/2000/xmlns/");
                attribute0.Value = "urn:oasis:names:tc:SAML:2.0:protocol";

                XmlAttribute attribute1 = doc.CreateAttribute("xmlns", "ds",
"http://www.w3.org/2000/xmlns/");
                attribute1.Value = "http://www.w3.org/2000/09/xmldsig#";
```

```
                XmlAttribute attribute2 = doc.CreateAttribute("xmlns", "eidas",
"http://www.w3.org/2000/xmlns/");
                attribute2.Value = "http://eidas.europa.eu/saml-extensions";

                XmlAttribute attribute3 = doc.CreateAttribute("xmlns", "saml2",
"http://www.w3.org/2000/xmlns/");
                attribute3.Value = "urn:oasis:names:tc:SAML:2.0:assertion";

                XmlAttribute attribute4 = doc.CreateAttribute("Consent");
                attribute4.Value = "urn:oasis:names:tc:SAML:2.0:consent:unspecified";

                XmlAttribute attribute6 = doc.CreateAttribute("ForceAuthn");
                if (auth) attribute6.Value = "true";
                else attribute6.Value = values[0];

                XmlAttribute attribute8 = doc.CreateAttribute("IsPassive");
                attribute8.Value = values[1];

                XmlAttribute attribute11 = doc.CreateAttribute("Version");
                attribute11.Value = "2.0";

                samlreq.Attributes.Append(attribute0);
                samlreq.Attributes.Append(attribute1);
                samlreq.Attributes.Append(attribute2);
                samlreq.Attributes.Append(attribute3);
                samlreq.Attributes.Append(attribute4);
                samlreq.Attributes.Append(attribute6);
                samlreq.Attributes.Append(attribute8);
                samlreq.Attributes.Append(attribute11);

                XmlNode NameIDPolicy = doc.CreateElement("saml2p", "NameIDPolicy",
"urn:oasis:names:tc:SAML:2.0:protocol");
                XmlAttribute AllowCreate = doc.CreateAttribute("AllowCreate");
                AllowCreate.Value = values[2];
                NameIDPolicy.Attributes.Append(AllowCreate);
                XmlAttribute Format = doc.CreateAttribute("Format");
```

```
                Format.Value = "urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified";
                NameIDPolicy.Attributes.Append(Format);

                XmlNode extensions = doc.CreateElement("saml2p", "Extensions",
"urn:oasis:names:tc:SAML:2.0:protocol");

                XmlNode sptype = doc.CreateElement("eidas", "SPType",
"http://eidas.europa.eu/saml-extensions");
                sptype.AppendChild(doc.CreateTextNode("public"));
                extensions.AppendChild(sptype);

                XmlNode req = doc.CreateElement("eidas", "RequestedAttributes",
"http://eidas.europa.eu/saml-extensions");
                extensions.AppendChild(req);

                if (afirma)
                {
                    XmlNode re0 = doc.CreateElement("eidas", "RequestedAttribute",
"http://eidas.europa.eu/saml-extensions");
                    XmlAttribute atre00 = doc.CreateAttribute("FriendlyName");
                    atre00.Value = "AFirmaIdP";

                    XmlAttribute atre01 = doc.CreateAttribute("Name");
                    atre01.Value = "http://es.minhafp.clave/AFirmaIdP";

                    XmlAttribute atre02 = doc.CreateAttribute("NameFormat");
                    atre02.Value = "urn:oasis:names:tc:SAML:2.0:attrname-format:uri";

                    XmlAttribute atre03 = doc.CreateAttribute("isRequired");
                    atre03.Value = "false";

                    re0.Attributes.Append(atre00);
                    re0.Attributes.Append(atre01);
                    re0.Attributes.Append(atre02);
                    re0.Attributes.Append(atre03);
```

```
                req.AppendChild(re0);
            }
            if (clavePermanente)
            {
                XmlNode re1 = doc.CreateElement("eidas", "RequestedAttribute",
"http://eidas.europa.eu/saml-extensions");
                XmlAttribute atre10 = doc.CreateAttribute("FriendlyName");
                atre10.Value = "GISSIdP";

                XmlAttribute atre11 = doc.CreateAttribute("Name");
                atre11.Value = "http://es.minhafp.clave/GISSIdP";

                XmlAttribute atre12 = doc.CreateAttribute("NameFormat");
                atre12.Value = "urn:oasis:names:tc:SAML:2.0:attrname-format:uri";

                XmlAttribute atre13 = doc.CreateAttribute("isRequired");
                atre13.Value = "false";

                re1.Attributes.Append(atre10);
                re1.Attributes.Append(atre11);
                re1.Attributes.Append(atre12);
                re1.Attributes.Append(atre13);

                req.AppendChild(re1);
            }
            if (pin24h)
            {
                XmlNode re2 = doc.CreateElement("eidas", "RequestedAttribute",
"http://eidas.europa.eu/saml-extensions");
                XmlAttribute atre20 = doc.CreateAttribute("FriendlyName");
                atre20.Value = "AEATIdP";

                XmlAttribute atre21 = doc.CreateAttribute("Name");
                atre21.Value = "http://es.minhafp.clave/AEATIdP";

                XmlAttribute atre22 = doc.CreateAttribute("NameFormat");
```

```
                    atre22.Value = "urn:oasis:names:tc:SAML:2.0:attrname-format:uri";

                    XmlAttribute atre23 = doc.CreateAttribute("isRequired");
                    atre23.Value = "false";

                    re2.Attributes.Append(atre20);
                    re2.Attributes.Append(atre21);
                    re2.Attributes.Append(atre22);
                    re2.Attributes.Append(atre23);

                    req.AppendChild(re2);
                }
                if (eidas)
                {
                    XmlNode re3 = doc.CreateElement("eidas", "RequestedAttribute",
"http://eidas.europa.eu/saml-extensions");
                    XmlAttribute atre30 = doc.CreateAttribute("FriendlyName");
                    atre30.Value = "EIDASIdP";

                    XmlAttribute atre31 = doc.CreateAttribute("Name");
                    atre31.Value = "http://es.minhafp.clave/EIDASIdP";

                    XmlAttribute atre32 = doc.CreateAttribute("NameFormat");
                    atre32.Value = "urn:oasis:names:tc:SAML:2.0:attrname-format:uri";

                    XmlAttribute atre33 = doc.CreateAttribute("isRequired");
                    atre33.Value = "false";

                    re3.Attributes.Append(atre30);
                    re3.Attributes.Append(atre31);
                    re3.Attributes.Append(atre32);
                    re3.Attributes.Append(atre33);

                    req.AppendChild(re3);
                }
```

```
                if (relaystate)
                {
                    XmlNode re4 = doc.CreateElement("eidas", "RequestedAttribute",
"http://eidas.europa.eu/saml-extensions");
                    XmlAttribute atre40 = doc.CreateAttribute("FriendlyName");
                    atre40.Value = "RelayState";

                    XmlAttribute atre41 = doc.CreateAttribute("Name");
                    atre41.Value = "http://es.minhafp.clave/RelayState";

                    XmlAttribute atre42 = doc.CreateAttribute("NameFormat");
                    atre42.Value = "urn:oasis:names:tc:SAML:2.0:attrname-format:uri";

                    XmlAttribute atre43 = doc.CreateAttribute("isRequired");
                    atre43.Value = "false";

                    re4.Attributes.Append(atre40);
                    re4.Attributes.Append(atre41);
                    re4.Attributes.Append(atre42);
                    re4.Attributes.Append(atre43);

                    req.AppendChild(re4);
                }
                if (idpMovil)
                {
                    XmlNode re5 = doc.CreateElement("idpMovil", "RequestedAttribute",
"http://eidas.europa.eu/saml-extensions");
                    XmlAttribute atre50 = doc.CreateAttribute("FriendlyName");
                    atre50.Value = "CLVMOVILIdP";

                    XmlAttribute atre51 = doc.CreateAttribute("Name");
                    atre51.Value = "http://es.minhafp.clave/CLVMOVILIdP";

                    XmlAttribute atre52 = doc.CreateAttribute("NameFormat");
                    atre52.Value = "urn:oasis:names:tc:SAML:2.0:attrname-format:uri";
```

```
                XmlAttribute atre53 = doc.CreateAttribute("isRequired");
                atre53.Value = "false";

                re5.Attributes.Append(atre50);
                re5.Attributes.Append(atre51);
                re5.Attributes.Append(atre52);
                re5.Attributes.Append(atre53);

                req.AppendChild(re5);
            }

            XmlNode requestAuthnContext = doc.CreateElement("saml2p",
"RequestedAuthnContext", "urn:oasis:names:tc:SAML:2.0:protocol");
            XmlAttribute comparison = doc.CreateAttribute("Comparison");
            comparison.Value = "minimum";
            requestAuthnContext.Attributes.Append(comparison);

            XmlNode authcontextclassref = doc.CreateElement("saml2",
"AuthnContextClassRef", "urn:oasis:names:tc:SAML:2.0:assertion");
            //Adding LoA
            if(newLOA != null)
authcontextclassref.AppendChild(doc.CreateTextNode(newLOA));
            else
authcontextclassref.AppendChild(doc.CreateTextNode(Conf.get_loa_forms()[0]));

            requestAuthnContext.AppendChild(authcontextclassref);

            samlreq.AppendChild(extensions);
            samlreq.AppendChild(NameIDPolicy);
            samlreq.AppendChild(requestAuthnContext);
            //END
            doc.AppendChild(samlreq);
        }
        catch (Exception e)
        {
            //Log: "Error creating the request. Any Conf data could be null: " + e
```

```
            }

            return doc;
        }
```

9.2.8 Validación de una Respuesta

En este caso, se debe recoger la respuesta enviada por parte de la pasarela. Esta respuesta XML contendrá aserciones que han de ser inspeccionadas para comprobar que el servidor ha validado los datos que le hemos proporcionado, y que nos ha mandado los datos correctamente. El ejemplo usado para esta demo se encuentra en Saml2SPWeb/ResponseHandler.cs.

El orden de validación es:

- Recoger el nodo “saml2p:statusMessage” y comprobar que es success.
- Comprobar que el certificado está en nuestro key store de certificados de confianza
- Comprobar la firma.
- Comprobar que la respuesta coincide con la petición.
- Si es necesario, descifrar los datos personales.

```
      //There is the process to get the response from Cl@ve node
                string samlResponseField =
ConfigurationSettingsHelper.GetCriticalConfigSetting("samlResponseField");
                string responseData = context.Request.Form[samlResponseField];

                byte[] reqDataB64 = Convert.FromBase64String(responseData);
                Stream stream = new MemoryStream(reqDataB64);

                //Convert the response to XML form
                XmlDocument xml = new XmlDocument();
                xml.PreserveWhitespace = true;
                xml.Load(stream);
                //Getting the issuer node
                var issuer = xml.GetElementsByTagName("saml2:Issuer");
                string urlMetadata = issuer[0].InnerText;
                //Getting metadata from url to validate
                XmlDocument xmlMetadata = XMLData.ReadMetadata(urlMetadata);
                bool metadataVerified = Utils.VerifyMetadata(xmlMetadata, false);
```

```
                //checks Cl@veNode metadata and SAML response. true as parameter saves cert
in your key store
                if (metadataVerified && Saml2Code.Utils.VerifyResponse(xml))
                {
                    string status =
xml.GetElementsByTagName("saml2p:StatusMessage")[0].InnerText;

                    //checking if request was success in the post method
                    if (status == "urn:oasis:names:tc:SAML:2.0:status:Success")
                    {
                        XmlDocument request = (XmlDocument)context.Session["SAMLRequest"];
                        string id =
request.FirstChild.Attributes.GetNamedItem("ID").InnerText;
                        string inResponseTo =
xml.FirstChild.Attributes.GetNamedItem("InResponseTo").InnerText;
                        if (id != null && inResponseTo != null && id.Equals(inResponseTo))
                        {
                            XmlDocument xml_response = null;
                            if (xml.GetElementsByTagName("saml2:EncryptedAssertion").Count >
0)
                            {
                                var dataDecripted = CipherUtils.Decrypt(xml);
                                xml_response = new XmlDocument();
                                xml_response.LoadXml(dataDecripted);
                            }
                            else
                            {
                                xml_response = xml;
                            }

                            context.Session["SAMLResponse"] = xml_response;
                            context.Session["result"] = "success";
                            //Redirect to the next
                            returnAction = Conf.get_url_response_success();
                        } else
                        {
                            Utils.DoLog(new Exception("InResponseTo field does not match
Request ID"));
                        }
                    }
```

```
                else
                {
                    Utils.DoLog(new Exception("Is a SAMLFail response: " + status));
                }
            }
            else
            {
                Utils.DoLog(new Exception("Response or metadata NOT valid"));
            }
        }
        catch (Exception ex)
        {
            Utils.DoLog(ex);
            returnAction = Conf.get_url_return_if_fail_metadata_eidas_node();
        }
        context.Response.Redirect(returnAction);
```

9.2.8.1 *Validación del Período de Validez del Ticket SAML*

El código proporcionado valida el período de validez del tiquet SAML si está informado. Es decir, valida que la fecha actual este entre los atributos NotBefore y NotOnOrAfter del tiquet SAML.

Para que esta validación sea efectiva el servidor del SP y el servidor de la plataforma Cl@ve tiene que estar sincronizados.

En caso de desfase en las horas, se pude indicar clockskew adecuado en la validación. En el parámetro ClockSkewHours del fichero de propiedades se indica el skew en horas.

9.2.9 Descifrado de una Respuesta

En primer lugar, se debe de realizar el descifrado asimétrico. El resultado del mismo nos permitirá obtener una clave, la cual usaremos en el descifrado simétrico. Los métodos que se muestran a continuación se encuentran en CommonSaml2/CipherUtils.cs.

Para el primer descifrado se debe de utilizar nuestro propio certificado en formato pem (recordemos que todo se obtiene del fichero de configuración CommonSaml2/Conf.cs).

9.2.9.1 *Descifrado Asimétrico*

```
public static byte[] AsymmetricDecrypt(XmlDocument xml)
        {
```

```
        //We must get the cipherValue to decrypt
            string data = xml.GetElementsByTagName("xenc:CipherValue")[0].InnerText;

            IAsymmetricBlockCipher cipher = new OaepEncoding(new RsaEngine(), new
Sha1Digest());

            TextReader reader;

            AsymmetricCipherKeyPair key;
        //We must use our .pem with private and public keys
            using (reader = File.OpenText(@Conf.get_dir_pem()))
            {
                key = (AsymmetricCipherKeyPair)new PemReader(reader).ReadObject();
            }

            byte[] cipheredBytes = Convert.FromBase64String(data);

            RsaKeyParameters privateKey = (RsaKeyParameters)key.Private;

            cipher.Init(false, privateKey);

            byte[] deciphered = cipher.ProcessBlock(cipheredBytes, 0, cipheredBytes.Length);
        //The method will return decripted cipherValue (the symmetric key)
            return deciphered;
         }
```

9.2.9.2 Descifrado Simétrico

```
private static byte[] DecryptWithKey(byte[] encryptedMessage, byte[] key,
            int nonSecretPayloadLength = 0)
        {
//We know that gcm encrypt will have a authentication bytes
            if (key == null || key.Length != 256 / 8)
            {
                throw new ArgumentException(String.Format("Key needs to be {0} bit!", 256),
"key");
            }

            if (encryptedMessage == null || encryptedMessage.Length == 0)
```

```
            {
                throw new ArgumentException("Encrypted Message Required!",
"encryptedMessage");
            }

            using (var cipherStream = new MemoryStream(encryptedMessage))
            using (var cipherReader = new BinaryReader(cipherStream))
            {
                var nonce = cipherReader.ReadBytes(96 / 8);

                var cipher = new GcmBlockCipher(new AesFastEngine());
                var parameters = new AeadParameters(new KeyParameter(key), 128, nonce,
null);
                cipher.Init(false, parameters);

                var cipherText = cipherReader.ReadBytes(encryptedMessage.Length -
nonSecretPayloadLength - nonce.Length);
                var plainText = new byte[cipher.GetOutputSize(cipherText.Length)];

                try
                {
                    var len = cipher.ProcessBytes(cipherText, 0, cipherText.Length,
plainText, 0);
                    cipher.DoFinal(plainText, len);
                }
                catch (InvalidCipherTextException)
                {
                    //Return null if it doesn't authenticate
                    return null;
                }
//the text with decripted attributes from user
                return plainText;
            }
        }
```

El siguiente método se encargará de hacer uso de los anteriores, simplificando el número de llamadas.

```
public static string Decrypt(XmlDocument xml, int nonSecretPayloadLength = 0)
        {
             //Asymmetric decrypt
```

```
            var key = AsymmetricDecrypt(xml);


            byte[] data =
Convert.FromBase64String(xml.GetElementsByTagName("xenc:CipherValue")[1].InnerText);
       //Symmetric decrypt
            var plaintext = DecryptWithKey(data, key, nonSecretPayloadLength);


            return Encoding.ASCII.GetString(plaintext);
        }
```

9.2.10 Log Out

Para realizar el Log out es tan sencillo como presionar en la vista final en la que se muestran los atributos del usuario el botón “Log out”. Enviará un ticket SAML2 con la petición de cierre de sesión y recibirá la confirmación. Una vez recibida, devolverá al usuario a la primera vista SP.aspx.

El código está en XMLData.cs / saml2LogOut().

```
public static XmlDocument saml2LogOut()
        {
            XmlDocument doc = new XmlDocument();
            doc.PreserveWhitespace = false;
            try
            {
                // logoutrequest/BEGIN
                XmlNode logoutrequest = doc.CreateElement("saml2p", "LogoutRequest",
"urn:oasis:names:tc:SAML:2.0:protocol");


                XmlAttribute attribute0 = doc.CreateAttribute("xmlns", "saml2p",
"http://www.w3.org/2000/xmlns/");
                attribute0.Value = "urn:oasis:names:tc:SAML:2.0:protocol";


                XmlAttribute attribute1 = doc.CreateAttribute("xmlns", "ds",
"http://www.w3.org/2000/xmlns/");
                attribute1.Value = "http://www.w3.org/2000/09/xmldsig#";


                XmlAttribute attribute2 = doc.CreateAttribute("xmlns", "eidas",
"http://www.w3.org/2000/xmlns/");
                attribute2.Value = "http://eidas.europa.eu/saml-extensions";


                XmlAttribute attribute3 = doc.CreateAttribute("xmlns", "saml2",
"http://www.w3.org/2000/xmlns/");
```

```
                attribute3.Value = "urn:oasis:names:tc:SAML:2.0:assertion";

                XmlAttribute Destination = doc.CreateAttribute("Destination");
                Destination.Value = Conf.get_url_eidas();

                XmlAttribute ID = doc.CreateAttribute("ID");
                ID.Value = "_" + Guid.NewGuid().ToString();

                XmlAttribute IssueInstant = doc.CreateAttribute("IssueInstant");
                IssueInstant.Value = Conf.get_valid_until();

                XmlAttribute NotOnOrAfter = doc.CreateAttribute("NotOnOrAfter");
                NotOnOrAfter.Value = Conf.get_valid_until();

                XmlAttribute Reason = doc.CreateAttribute("Reason");
                Reason.Value = "urn:oasis:names:tc:SAML:2.0:logout:user";

                XmlAttribute Version = doc.CreateAttribute("Version");
                Version.Value = "2.0";

                logoutrequest.Attributes.Append(attribute0);
                logoutrequest.Attributes.Append(attribute1);
                logoutrequest.Attributes.Append(attribute2);
                logoutrequest.Attributes.Append(attribute3);
                logoutrequest.Attributes.Append(Destination);
                logoutrequest.Attributes.Append(ID);
                logoutrequest.Attributes.Append(IssueInstant);
                logoutrequest.Attributes.Append(NotOnOrAfter);
                logoutrequest.Attributes.Append(Reason);
                logoutrequest.Attributes.Append(Version);
                //logoutrequest/END

                // issuer/BEGIN
                XmlNode issuer = doc.CreateElement("saml2", "Issuer",
"urn:oasis:names:tc:SAML:2.0:assertion");
                issuer.AppendChild(doc.CreateTextNode(Conf.get_url_return_logout()));
                logoutrequest.AppendChild(issuer);
                //issuer/END
```

```
                // nameid/BEGIN
                XmlNode nameid = doc.CreateElement("saml2", "NameID",
"urn:oasis:names:tc:SAML:2.0:assertion");

                XmlAttribute Format = doc.CreateAttribute("Format");
                Format.Value = "urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified";

                XmlAttribute SPNameQualifier = doc.CreateAttribute("SPNameQualifier");
                SPNameQualifier.Value = Conf.get_url_return_response();

                nameid.Attributes.Append(Format);
                nameid.Attributes.Append(SPNameQualifier);

                nameid.AppendChild(doc.CreateTextNode(Conf.get_provider_name()));

                logoutrequest.AppendChild(nameid);
                // nameid/END

                doc.AppendChild(logoutrequest);
                return doc;
            }
            catch (Exception e)
            {
                //TODO
            }
            return null;
        }
```

9.2.11 Actualización Automática de Certificados

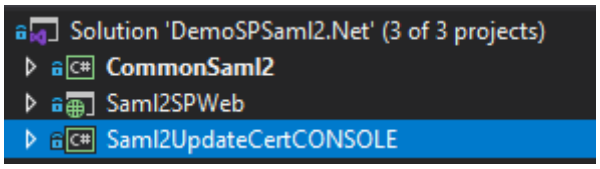


Ilustración 10: Paquete de Integración C# - Aplicación de Consola Saml2UpdateCertCONSOLE3

Saml2UpdateCertCONSOLE es una aplicación de consola con funcionalidad autocontenida (no utiliza rutinas de la biblioteca *CommonSaml2*) que permite descargar los certificados de firma y cifrado de la pasarela Cl@ve y persistirlos en el sistema de ficheros del SP.

9.2.11.1 Configuración

En el fichero de configuración de la aplicación se definen las siguientes entradas:

- *UrlDescargaCertClave*: endpoint del servicio de la pasarela desde donde se descarga el XML que contiene la parte pública de los certificados.

  Para poder establecer una conexión segura mediante SSL con la pasarela es necesario disponer en el *Almacén de Certificados de Windows* (de *Usuario*, en el caso de que la identidad con la que se ejecuta la aplicación sea una cuenta de usuario, o de *Máquina Local*, si la identidad es una cuenta de servicio) del certificado de host de la pasarela y, en su caso, de toda su ruta de certificación. Los certificados han de instalarse en la carpeta *Trusted Root Certification Authorities* del almacén.

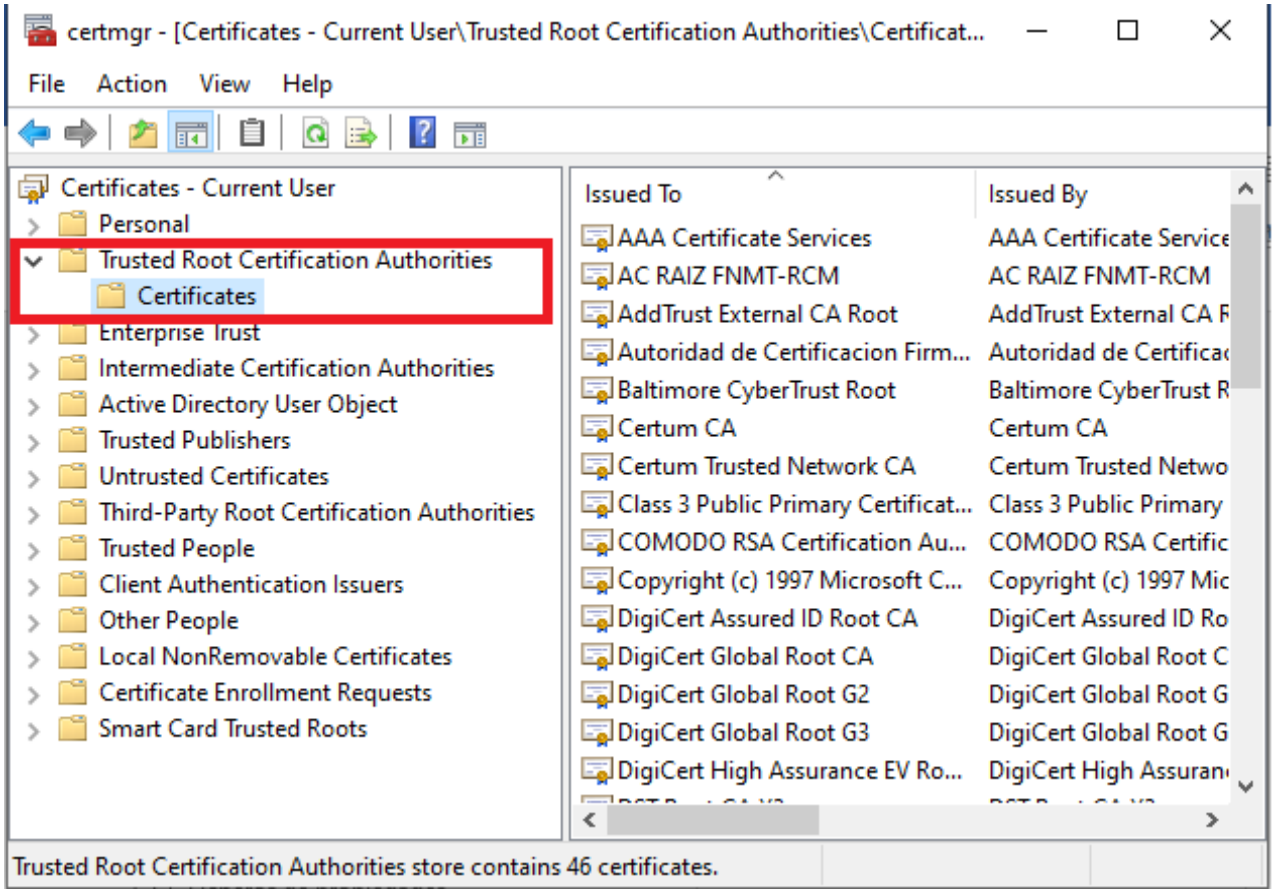


Ilustración 11: Paquete de Integración C# - Aplicación de Consola Saml2UpdateCertCONSOLE (Trusted Root Certification Authorities)

- *RutaCertificados*: ubicación del sistema de ficheros del SP donde se guardarán los certificados con extensión *.cer*.

La identidad con la que se ejecuta la aplicación debe tener permisos de lectura/escritura sobre esta ubicación.

9.2.11.2 Proceso

La clase ***UpdateCertWorker.cs*** implementa el proceso de actualización de certificados llevando a cabo las siguientes acciones:

- Obtener el XML de la pasarela Cl@ve (*GetXmlStrFromUrl*)
- Extraer la información de certificados del XML (*ExtractCertificatesFromXml*)
- Limpiar (o crear, en el caso de que no exista) el directorio donde se han de persistir los certificados (*CleanCertificateDirectory*)
- Para cada certificado definido en el XML:
  - Persistir el certificado en la ubicación definida (*PersistCertificate2FileSystem*)
- Generar el archivo de definición de certificados descargados *certificates.json* (CreateCertificateDefinitionFile)
  Este archivo define en notación JSON una colección de objetos de tipo *ClaveCertificate* con las siguientes propiedades:
  - *Name*: nombre del certificado tal como se especifica en el XML.
  - *Use*: "*Signing*" si el certificado se usa para firma o "*Encryption*" si se usa para cifrado.
  - *Active*: booleano que indica el estado del certificado (si está activo o no) en la pasarela.
  - *DownloadDate*: fecha de descarga del certificado en formato ISO 8601.

  Ej.

```
[
   {
      "Name":"proxy_sign_5731fcb0",
      "Use":"Signing",
      "Active":true,
      "DownloadDate":"2021-05-28T07:46:08.8507331Z"
   },
   {
      "Name":"proxy_cipher_hhhhh",
      "Use":"Encryption",
      "Active":false,
      "DownloadDate":"2021-05-28T07:46:08.8507331Z"
   },
   {
      "Name":"proxy_cipher_5731fcb0",
      "Use":"Encryption",
      "Active":true,
      "DownloadDate":"2021-05-28T07:46:08.8507331Z"
   }
]
```

9.2.11.3 Planificación del Proceso

Si se desea que el proceso de actualización se realice con una determinada periodicidad se puede ejecutar mediante el *Planificador de Tareas de Windows*.

Con el kit se facilita una plantilla de definición de tarea en formato XML (*SAML2UpdateCert_TaskDefinition.xml*) a utilizar como base para la creación de una nueva tarea. Para ello, se importará la plantilla desde el *Planificador* y se modificarán los siguientes parámetros:

- Cuenta de ejecución: identidad con la que ejecutará la aplicación.

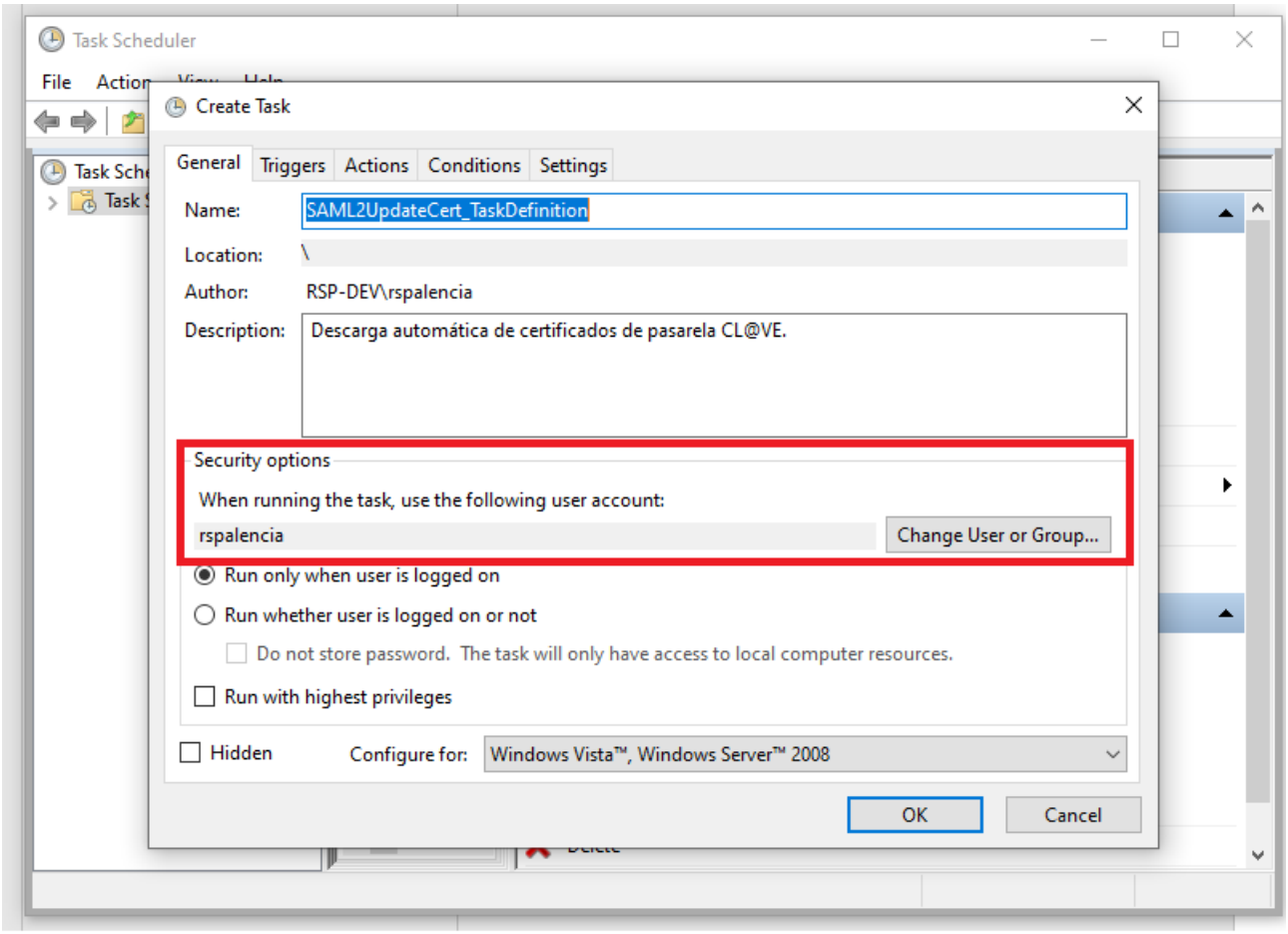


Ilustración 12: Paquete de Integración C# - Aplicación de Consola Saml2UpdateCertCONSOLE (Planificador de Tareas de Windows)

- Programa/script de la acción: ruta al ejecutable de la aplicación.

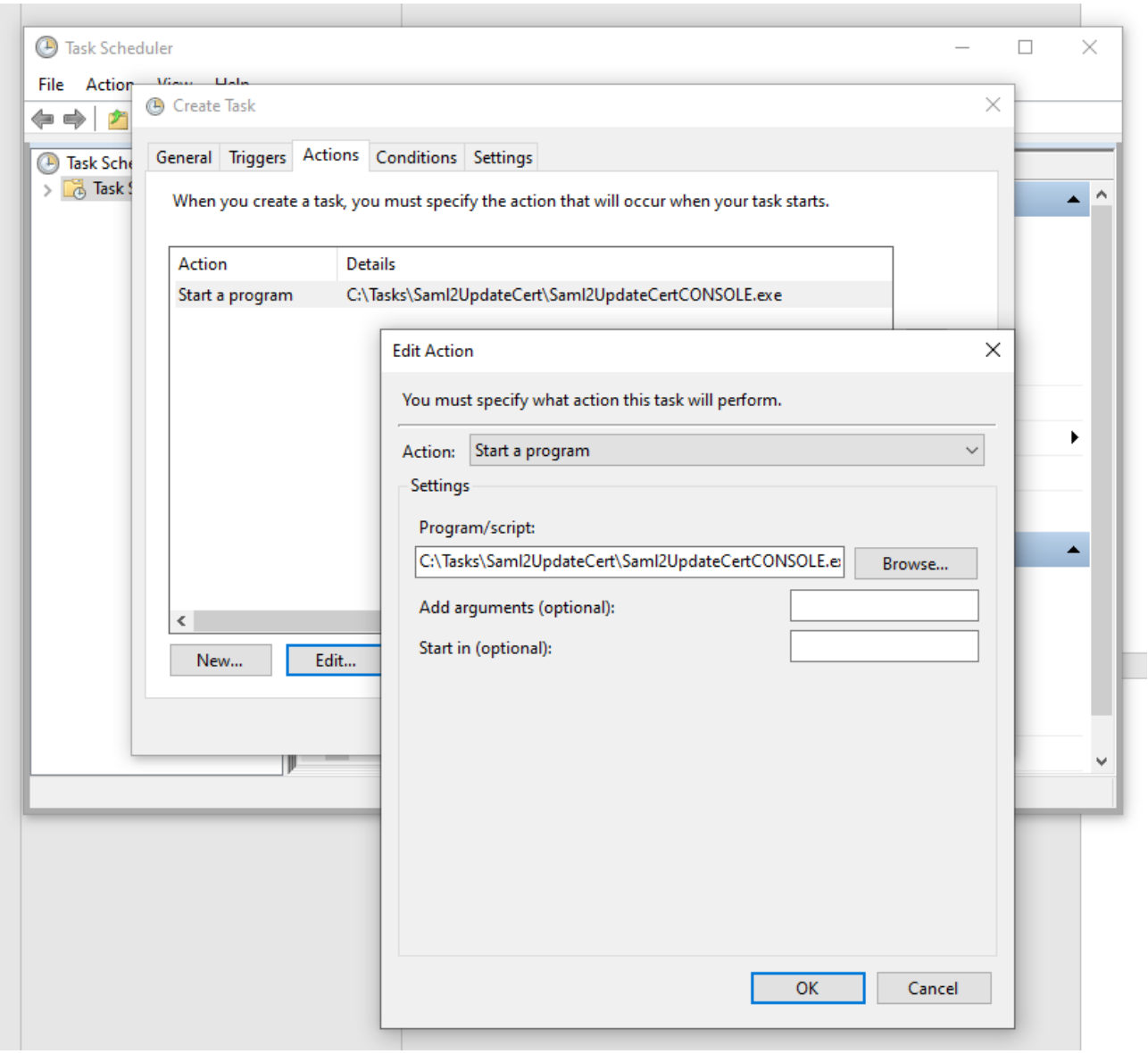


Ilustración 13: Paquete de Integración C# - Aplicación de Consola SamlUpdateCertCONSOLE (Acción del Planificador de Tareas de Windows)

- Programa/script de la acción: ruta al ejecutable de la aplicación.

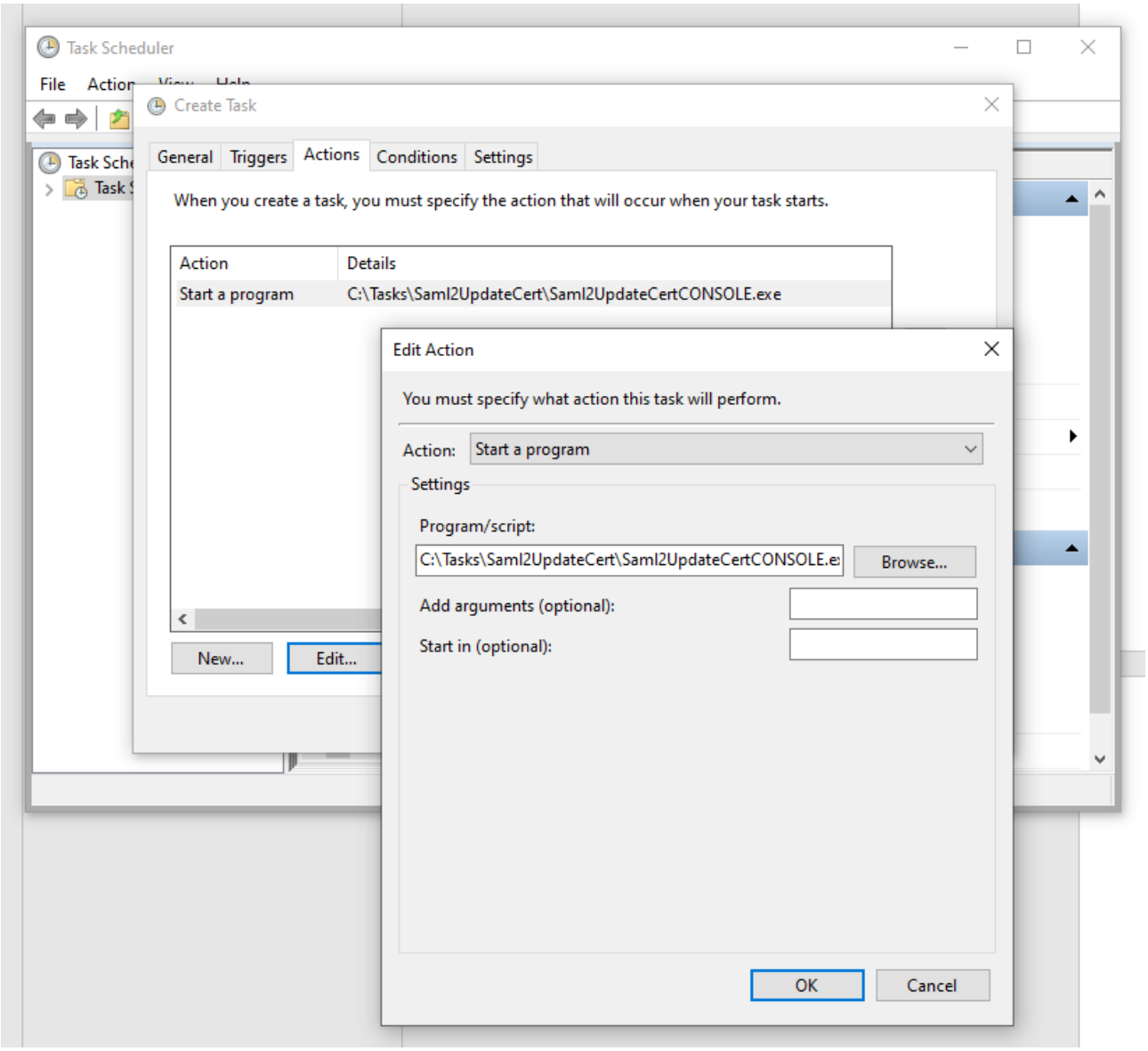


Ilustración 14: Paquete de Integración C# - Aplicación de Consola Saml2UpdateCertCONSOLE (Script de la Acción del Planificador de Tareas de Windows)

- Disparador: configuración de la periodicidad con la que se ejecutará la tarea.

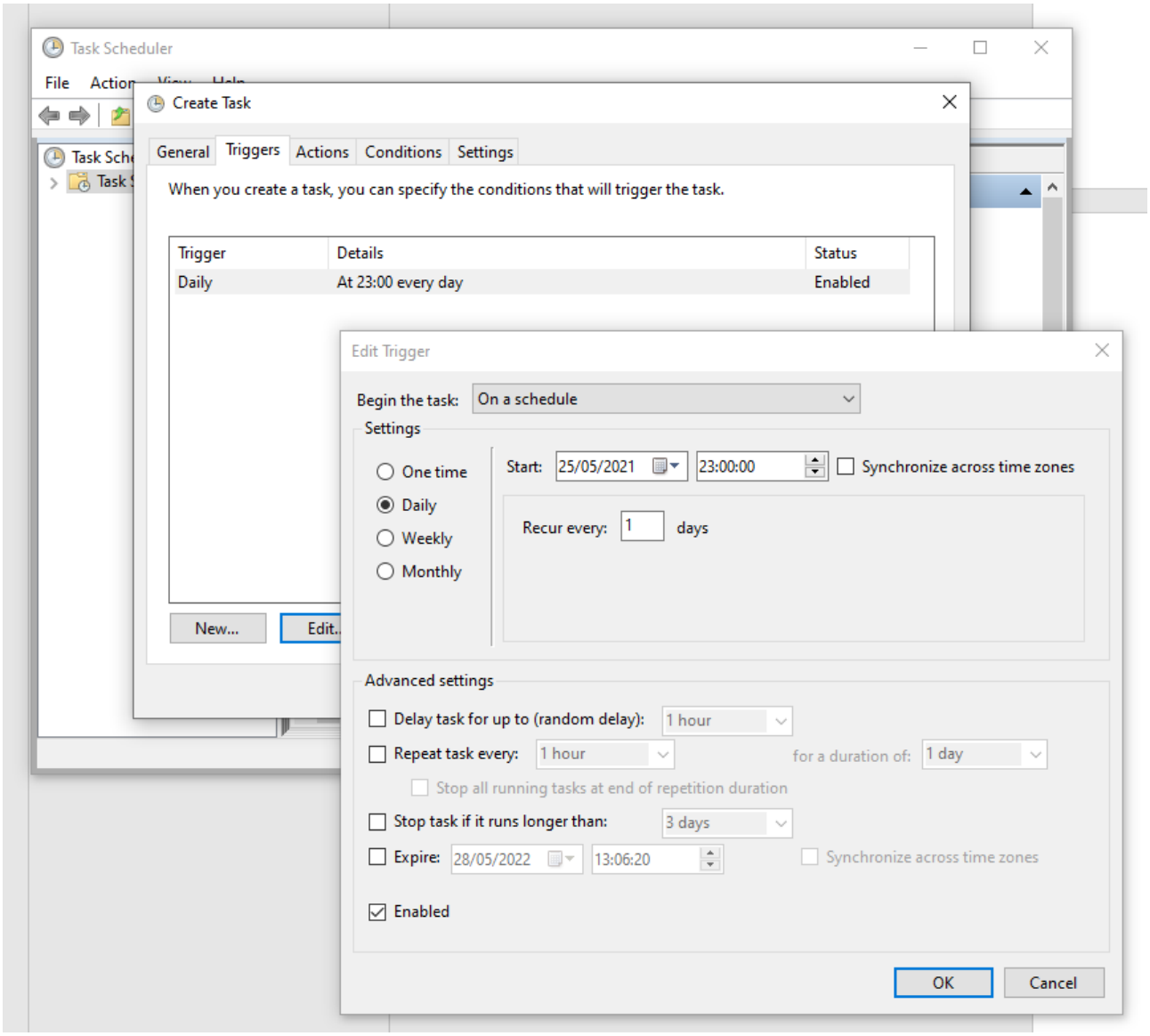


Ilustración 15: Paquete de Integración C# - Aplicación de Consola Saml2UpdateCertCONSOLE (Disparador de la Acción del Planificador de Tareas de Windows)

9.3 Paquete de Integración PHP

Este paquete de integración está destinado a SP's que deseen hacer uso de la tecnología PHP. Para su correcto funcionamiento, el sistema ha de cumplir una serie de requisitos mínimos:

Concepto	Valores testados	Valores soportados
Versión de PHP	8.2.0	5.6.31 o superiores
Servidores de aplicaciones	Apache 2.4.26	Apache Web Server
IDE	Netbeans 8.2	Netbeans 8.2 o superiores

9.3.1 Notas Importantes para Conectar el Kit contra Clave2 Pasarela del entorno de Servicios Estables (SE)

La configuración del presente Kit utiliza un certificado de prueba, a modo de ejemplo, para poder conectarse con Servicios Estables. Dicha configuración debe cambiarse por la que utilice el Organismo en Servicios Estables. Hay que tener en cuenta que todo Organismo que vaya a utilizar Clave debe realizar una solicitud de alta en el servicio, y que el certificado que vaya a utilizar debe de ser emitido para el Organismo. En otro caso podría pasar que varios Organismos utilizaran para sus accesos el certificado de serie añadido en los kits.

9.3.1.1 Configurar Kit PHP Clave para Conectar con el SP '21114293V_E04975701' de Servicios Estables (SE)

Modificar el fichero 'C:\xampp\htdocs\SimpleSamlSP\lib\SAML2\Constants.php':

modificar: const SPID = '21114293V_E04975701';

Añadir al fichero 'C:\xampp\htdocs\SimpleSamlSP\config\authsources.php' el siguiente código:

```
'21114293V_E04975701' => array(
        'saml:SP',
        'certificate' => 'sello_kit_de_pruebas_ac_sector_p_blico_.crt',
        'validate.certificate' => 'sello_entidad_sgad_pruebas.cer',
        'privatekey' => 'sello_kit_de_pruebas_ac_sector_p_blico_.pem',
        'privatekey_pass' => 'changeit',
        'name' => array(
           'en' => 'demo-sp-php',
           'pt' => 'demo-sp-php',
                       'es' => 'demo-sp-php',
        ),
        'attributes.NameFormat' => 'urn:oasis:names:tc:SAML:2.0:attrname-format:uri',
        'sign.authnrequest' => TRUE,
        'sign.logout' => TRUE,
        'entityID' => $entityIdTemp,
        'idp' => 'https://se-pasarela.clave.gob.es/Proxy2/ServiceProvider',
        'idp_binding' => 'urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST',
        'logout.url' => 'http://localhost:81/SP/logoutResponse.php',
        'OrganizationName' => array(
           //     'en' => 'Eidas SP Node',
           'es' => 'SP nodo Eidas',
        ),
        'OrganizationDisplayName' => array (
           //     'en' => 'Eidas SP Node',
```

```
            'es' => 'SP nodo Eidas',
        ),
        'OrganizationURL' => array (
            //  'en' => 'https://se-eidas.redsara.es',
            'es' => 'https://se-eidas.redsara.es',
        ),
    ),
```

9.3.1.2 Contenido de la Carpeta 'C:\xampp\htdocs\SimpleSamlSP\cert'

- **sello_kit_de_pruebas_ac_sector_p_blico_.crt:** certificado para conectar el kit con CLAVE PASARELA, que contiene la parte pública y privada. Para extraer este certificado, hay que abrir el contenedor 'sello_kit_de_pruebas_ac_sector_p_blico_.p12' con el 'KeyStore Explorer' (pwd: changeit), y exportar el propio certificado en formato 'PEM'. El certificado exportado en formato 'PEM' es el que vamos a guardar en un nuevo fichero con el nombre 'sello_kit_de_pruebas_ac_sector_p_blico_.crt'.
- **sello_kit_de_pruebas_ac_sector_p_blico_.pem:** abrimos el contenedor 'sello_kit_de_pruebas_ac_sector_p_blico_.crt' con el 'KeyStore Explorer' (pwd: changeit), y hacer el export 'export private key' (pwd: changeit) con format 'PKCS #8' (importante cifrar la exportación con el pwd 'changeit'). Cambiar la extensión del fichero generado a '.pem', abrirlo, y añadir al final del fichero todo el texto del fichero de antes: 'sello_kit_de_pruebas_ac_sector_p_blico_.crt'.
- **sello_entidad_sgad_pruebas.cer:** certificado público que permite confiar con el certificado que firma las peticiones SAML que vuelven de CLAVE2 PASARELA hacia el Kit. Es responsabilidad del organismo, actualizar este certificado en dicho KeyStore, con el certificado activo correspondiente en el momento de uso de este Kit.
- **sello_kit_de_pruebas_ac_sector_p_blico_.cer:** se trata de la parte pública del contenedor 'sello_kit_de_pruebas_ac_sector_p_blico_.p12', y debe estar importado (a través de WEBPORTAL de SERVICIOS ESTABLES) en la tabla de certificados del SP '21114293V_E04975701' (SP configurado en 'authsources.php'). Una vez cargado el certificado en el SP, hay que esperar a que se ejecute el daemon 'AutoDiscoverDaemon' que sincroniza los certificados des de la BDD hacia los KeyStores. Actualmente dicho SP ya tiene configurado este certificado público.

9.3.2 Ficheros de Configuración

Para poder adaptar el paquete a cada SP es necesario realizar algunas modificaciones en los ficheros de configuración.

En primer lugar, dentro de la carpeta *Config* se encuentra ***Authsources.php***. En este fichero debemos indicar en primer lugar un identificador de nuestro SP y a partir de ahí, definir el valor de una serie de parámetros relativos al mismo:

```
$entityIdTemp = 'http://localhost/SP/return.php';



$config = array(

  // This is a authentication source which handles admin authentication.
  'admin' => array(
    // The default is to use core:AdminPassword, but it can be replaced with
    // any authentication source.

    'core:AdminPassword',
  ),

 '21114293V_E04975701' => array(
                    'saml:SP',
                    'certificate' => 'sello_kit_de_pruebas_ac_sector_p_blico_.crt',
                    'validate.certificate' => 'sello_entidad_sgad_pruebas.cer',
                    'privatekey' => 'sello_kit_de_pruebas_ac_sector_p_blico_.pem',
                    'privatekey_pass' => 'changeit',
                    'name' => array(
                              'en' => 'demo-sp-php',
                              'pt' => 'demo-sp-php',
                 'es' => 'demo-sp-php',
                    ),
                    'attributes.NameFormat' => 'urn:oasis:names:tc:SAML:2.0:attrname-format:uri',
                    'sign.authnrequest' => TRUE,
            'sign.logout' => TRUE,
            'entityID' => $entityIdTemp,
                    'idp' => 'https://se-pasarela.clave.gob.es/Proxy2/ServiceProvider',
            'idp_binding' => 'urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST',
            'logout.url' => 'http://localhost/SP/logoutResponse.php',
            'OrganizationName' => array(
               //    'en' => 'Eidas SP Node',
                   'es' => 'SP nodo Eidas',
```

```
        ),
    'OrganizationDisplayName' => array (
        //    'en' => 'Eidas SP Node',
            'es' => 'SP nodo Eidas',
        ),
    'OrganizationURL' => array (
          //  'en' => 'https://se-eidas.redsara.es',
            'es' => 'https://se-eidas.redsara.es',
        ),
    ),
```

En segundo lugar, se deberán cambiar dos variables dentro del directorio lib/SAML2 en el fichero *Constants.php*:

```
//ID del SP en uso, necesario para recibir credenciales alemanas
const SPID = '21114293V_E04975701';
const SPAPPLICATION = 'SPApp';

//URL del endpoint del SP que gestionará las respuestas de la pasarela Cl@ve.
const ASSERTION_URL = ' http://localhost/SP/return.php’;
```

En el formulario inicial, el usuario podrá seleccionar cuál de los IdP’s deshabilitar. En caso de no seleccionar ninguno, todos los IdP aparecerán habilitados por defecto. En este fichero se declaran los IdP disponibles:

```
public static $idsAttrs = array('AFirmaIdP', 'GISSIdP', 'AEATIdP', 'EIDASIdP');
```

Este array contiene todos los IdP’s soportados. Para agregar un nuevo IdP, se debe añadir una nueva entrada en el mismo y su correspondiente entrada en $*attrs*:

```
public static $attrs = array(
//Nombre del nuevo IdP
            'AFirmaIdP.name' => 'AFirmaIdP',
//URI del nuevo IdP
            'AFirmaIdP.uri' => 'http://es.minhafp.clave/AFirmaIdP',
//Formato del espacio de nombres del nuevo IdP
             'AFirmaIdP.nameFormat' => 'urn:oasis:names:tc:SAML:2.0:attrname-format:uri',
//Valor por defecto del nuevo IdP
             'AFirmaIdP.value' => NULL,
```

9.3.3 Certificados

Como se ha comentado anteriormente, la confianza en la comunicación del módulo SP y la pasarela Cl@ve eIDAS se basa en certificados electrónicos. Por lo tanto, la pasarela tiene que confiar en el certificado del SP y éste tiene que confiar en el certificado de la pasarela. Para poder establecer esta relación de confianza, debemos tener en cuenta lo siguiente;

- En el fichero *config/config.php* se indicará cual será el directorio en el que se almacenarán los diferentes certificados, tanto los utilizados para firmar, como los empleados para validar respuestas (certificado de la pasarela Cl@ve de confianza) y descifrar contenido recibido.

```
$config = array('certdir' => 'cert/',
```

Por lo tanto, en el almacén de certificados */cert* deberán estar:

- 'certificate' => 'Stork.crt' – Certificado que el SP presentará en sus metadatos y que el nodo Cl@ve deberá almacenar para establecer confianza entre ambos.
- 'validate.certificate' => 'certificadoProxy.crt' – Certificado de la pasarela Cl@ve que nuestro Proveedor de Servicios almacenará para verificar posteriormente que las respuestas obtenidas provienen de un nodo Cl@ve de confianza.
- 'privatekey' => 'Stork.key' – Clave privada con la que nuestro Proveedor de Servicios firmará y descifrará las aserciones.

9.3.4 Librerías Utilizadas

En la carpeta */lib* se encuentran todas las librerías utilizadas en este paquete:

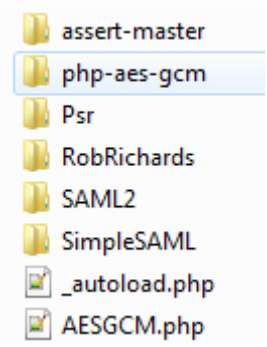


Ilustración 16: Paquete de Integración PHP – Dependencias

En este caso, el paquete de integración PHP se ha basado en el proyecto SimpleSAMLphp de código abierto, por ello se puede ver la existencia de librerías propias del proyecto (SAML2, SimpleSAML).

Adicionalmente se han utilizado librerías externas como:

- *assert-master*, para la extracción y validación de las aserciones

- *php-aes-gcm*, para la integración de criptografía de clave asimétrica con el algoritmo aes-gcm.
- *Psr,* para generar mensajes de log utilizados para la captura y visualización de errores
- *RobRichads*, para el USO de criptografía en ficheros XML.

9.3.5 Recogida de Datos Previa de las Fuentes de Autenticación

- Constants: Clase en la que se encuentran definidas las principales constantes utilizadas en todo el paquete y los métodos para recoger los atributos que serán solicitados en la petición. En primer lugar, será utilizada para obtener el ID de nuestro SP y de nuestra aplicación.
    - genDefaultAttrs – Método utilizado para recoger las opciones seleccionadas por el usuario.
- **SimpleSAML_Auth_Source:** Clase básica encargada de la autenticación del usuario y de la obtención de las fuentes de autenticación del mismo. Mediante el método *getById*, a partir de nuestro ID y de la configuración existente en el fichero *authsources.php*, devuelve un objeto de la clase *sspmod_saml_Auth_Source_SP*.

```
$authSource = SAML2\Constants::SPID;
$as = new SimpleSAML_Auth_Simple($authSource);
$as = SimpleSAML_Auth_Source::getById($authSource);

//Se carga la configuración de nuestro Proveedor de Servicio de config/authsources.php
$localConfig = $as->getLocalConfig();
//Se carga de la configuración la url del IdP Proxy.
$idp[0]= $as->getidp();
$idp[1]= $as->getidp_binding();

//Se carga la información del IdP de la configuración
$idplocalConfig = $as->getIdPfromConfig($idp);
$session = SimpleSAML_Session::getSessionFromRequest();
$session->cleanup();

try {
  $state['saml:sp:AuthId'] = $authSource;

//Cargar extensiones.
      $extensions = Constants::genDefaultAttrs($_POST);
```

9.3.6 Construcción de la Petición de Autenticación

Clases utilizadas en la construcción de la petición de autenticación:

- **eidas_saml_Message**: Subclase que extiende de la clase sspmod_saml_Message que define el código para la construcción de mensajes en formato SAML2 basados en la configuración disponible.

```
    $ar = eidas_saml_Message::buildAuthnRequest($extensions, $forceauthn, $localConfig,
$idplocalConfig);
```

  - *BuildAuthnRequest*. Método que construye la petición de autenticación basándose tanto en la información de la configuración como en los atributos previamente seleccionados. Para ello se basa en objetos de la subclase *SAML2_EidasAuthnRequest* que extienden de la clase *AuthnRequest*. A partir de este objeto comienza a añadir nodos con los parámetros correspondientes

```
public static function buildAuthnRequest($extensions, $forceauthn, SimpleSAML_Configuration
$splocalConfig, SimpleSAML_Configuration $idplocalConfig) {


            $ar = new SAML2\SAML2_EidasAuthnRequest();


            if ($spMetadata->hasValue('NameIDPolicy')) {
                  $nameIdPolicy = $spMetadata->getString('NameIDPolicy', NULL);
            } else {
                  $nameIdPolicy = $spMetadata->getString('NameIDFormat',
SAML2\Constants::NAMEID_TRANSIENT);
            }


            if ($nameIdPolicy !== NULL) {
                  $ar->setNameIdPolicy(array(
                        'Format' => $nameIdPolicy,
                        'AllowCreate' => TRUE,
                  ));
            }
               /**
                * Look for the endpoints in the information retrieved
                */
            $dst = $idplocalConfig->getDefaultEndpoint('SingleSignOnService',
array(SAML2\Constants::BINDING_HTTP_REDIRECT));
            $dst = $dst['Location'];
```

```
                /**
                 * Start building the message for the authentication request
                 */
            $ar->setIssuer($splocalConfig->getString('entityid'));
            $ar->setDestination($dst);
            $ar->setForceAuthn($splocalConfig->getBoolean('ForceAuthn', $forceauthn));
            $ar->setIsPassive($splocalConfig->getBoolean('IsPassive', FALSE));

            $protbind = $splocalConfig->getValueValidate('ProtocolBinding', array(
                        SAML2\Constants::BINDING_HTTP_POST,
                        SAML2\Constants::BINDING_HTTP_ARTIFACT,
                        SAML2\Constants::BINDING_HTTP_REDIRECT,
                  ), SAML2\Constants::BINDING_HTTP_POST);

            /* Setting the appropriate binding based on parameter in sp configuration
defaults to HTTP_POST */
            $ar->setProtocolBinding($protbind);

            if ($spMetadata->hasValue('AuthnContextClassRef')) {
                  $accr = $splocalConfig->getArrayizeString('AuthnContextClassRef');
                  $ar->setRequestedAuthnContext(array('AuthnContextClassRef' => $accr));
            }

           if (!empty($extensions)) {
             $ar->setExtensions($extensions);
           }

            self::addRedirectSign($splocalConfig, $idplocalConfig, $ar);
            return $ar;
      }
```

- *addRedirectSign*. Método que incluye la clave de firma y el certificado del emisor (SP) al mensaje. Sus parámetros de entrada son la configuración del emisor, la configuración existente relativa al destinatario y el mensaje.

```
private static function addRedirectSign(SimpleSAML_Configuration $splocalConfig,
SimpleSAML_Configuration $idplocalConfig, SAML2\SAML2_EidasAuthnRequest $message)
  {
```

```
    if ($message instanceof SAML2_LogoutRequest || $message instanceof SAML2_LogoutResponse)
{
      $signingEnabled = $splocalConfig->getBoolean('sign.logout', NULL);
      if ($signingEnabled === NULL) {
        $signingEnabled = $idplocalConfig->getBoolean('sign.logout', NULL);
      }
    } elseif ($message instanceof SAML2\SAML2_EidasAuthnRequest) {
      $signingEnabled = $splocalConfig->getBoolean('sign.authnrequest', NULL);
      if ($signingEnabled === NULL) {
        $signingEnabled = $idplocalConfig->getBoolean('sign.authnrequest', NULL);
      }
    }
    if ($signingEnabled === NULL) {
      $signingEnabled = $idplocalConfig->getBoolean('redirect.sign', NULL);
      if ($signingEnabled === NULL) {
        $signingEnabled = $splocalConfig->getBoolean('redirect.sign', FALSE);
      }
    }
    if (!$signingEnabled) {
      return;
    }
    self::addSign($splocalConfig, $idplocalConfig, $message);
  }
```

- *addSign*. Método encargado de buscar las claves y certificados indicados en los parámetros del SP y de añadir los correspondientes al mensaje. Se basa en métodos de la clase SimpleSAML_Utilities para el parseo y manipulación del mensaje y en objetos de la clase XMLSecurityKey para la manipulación de las claves.

```
public static function addSign(SimpleSAML_Configuration $splocalConfig,
SimpleSAML_Configuration $idplocalConfig, SAML2\SignedElement $element) {
    $keyArray = SimpleSAML_Utilities::loadPrivateKey($splocalConfig, TRUE);
    $certArray = SimpleSAML_Utilities::loadPublicKey($splocalConfig, FALSE);
    $privateKey = new \XMLSecurityKey(\XMLSecurityKey::RSA_SHA512, array('type' =>
'private'));
    if (array_key_exists('password', $keyArray)) {
      $privateKey->passphrase = $keyArray['password'];
    }
    $privateKey->loadKey($keyArray['PEM'], FALSE);


    $element->setSignatureKey($privateKey);
```

```
    if ($certArray === NULL) {
      /* We don't have a certificate to add. */
      return;
    }
```

9.3.7 Construcción de la Petición de Logout

Para realizar una solicitud de cierre de sesión, se debe generar una petición SAML similar a la de autenticación, pero, en este caso, de tipo logout.

En logout.php, se construye la petición partiendo de la configuración local e indicando el endpoint de la URL que deberá recibir y procesar la respuesta, y la URL del destino:

```
public static function buildLogoutRequest(SimpleSAML_Configuration $srclocalConfig,
SimpleSAML_Configuration $dstlocalConfig) {


            $lr = new SAML2\LogoutRequest();
            $lr->setIssuer($srclocalConfig->getString('logout.url'));
             $lr->setDestination($srclocalConfig->getString('idp'));
            self::addRedirectSignLogout($srclocalConfig, $dstlocalConfig, $lr);


            return $lr;
    }
```

9.3.8 Construcción del ticket SAML

A partir del objeto SAML2_EidasAuthnRequest obtenido previamente, se procede a la construcción del ticket SAML.

- **SAML2_EidasHTTPost** – Clase encargada de implementar el HTTP POST Binding. Para ello utilizará el método sendDefault o send, dependiendo si los atributos a solicitar han sido seleccionados por el usuario o si se han seleccionado por defecto.

```
$b = new SAML2_EidasHTTPPost($_POST['country'], $_POST);
$b->send($ar);
```

  - *Send* y *sendDefault*. Tienen como parámetro de entrada el objeto *SAML2_EidasAuthnRequest*. A partir de la información proporcionada en dicho objeto, convierten el mensaje en un XML inicialmente no firmado mediante el método

toUnsignedXML y posteriormente firmado gracias al método *toSignedXML*. Ambos métodos pertenecen a la clase *SAML2_Message.*

```
public function toUnsignedXML()
    {
        $this->document = DOMDocumentFactory::create();
        $root = $this->document->createElementNS(Constants::NS_SAMLP, 'saml2p:'.$this-
>tagName);
        $this->document->appendChild($root);

        $root->setAttributeNS(Constants::NS_SAML, 'saml2:tmp', 'tmp');
        $root->removeAttributeNS(Constants::NS_SAML, 'tmp');

        $root->setAttributeNS(Constants::NS_EIDAS, 'eidas:tmp', 'tmp');
        $root->removeAttributeNS(Constants::NS_EIDAS, 'tmp');

        $root->setAttribute('ID', $this->id);
        $root->setAttribute('Version', '2.0');
        $root->setAttribute('IssueInstant', gmdate('Y-m-d\TH:i:s\Z', $this->issueInstant));

        if ($this->destination !== null) {
            $root->setAttribute('Destination', $this->destination);
        }

        $root->setAttribute('ForceAuthn', 'true');

        if ($this->consent !== null && $this->consent !== Constants::CONSENT_UNSPECIFIED) {
            $root->setAttribute('Consent', $this->consent);
        }

        if ($this->issuer !== null) {
            if (is_string($this->issuer)) {
                Utils::addString($root, Constants::NS_SAML, 'saml2:Issuer', $this->issuer);
            } elseif ($this->issuer instanceof XML\saml\Issuer) {
                $this->issuer->toXML($root);
            }
        }
```

```
        if (!empty($this->extensions)) {
            Extensions::addList($root, $this->extensions);
        }
        return $root;
    }
```

```
public function toSignedXML()
    {

         $root = $this->toUnsignedXML();

        if ($this->signatureKey === null) {
            /* We don't have a key to sign it with. */

            return $root;
        }

        /* Find the position we should insert the signature node at. */
        if ($this->issuer !== null) {
            /*
             * We have an issuer node. The signature node should come
             * after the issuer node.
             */
            $issuerNode = $root->firstChild;
            $insertBefore = $issuerNode->nextSibling;
        } else {
            /* No issuer node - the signature element should be the first element. */
            $insertBefore = $root->firstChild;
        }

        Utils::insertSignature($this->signatureKey, $this->certificates, $root,
$insertBefore);

        return $root;
    }
```

- *insertSignature*. Es el método utilizados para insertar el nodo Signature en el XML y pertenece a la clase *Utils*. Sus parámetros de entrada son; la clave utilizada para firmar el

mensaje, los certificados que deben ser añadidos al nodo Signature, el nodo XML a firmar y el elemento XML que se tomara como referencia para colocar este nodo.

```
public static function insertSignature(
        \XMLSecurityKey $key,
        array $certificates,
        \DOMElement $root,
        \DOMNode $insertBefore = null
    ) {
        $objXMLSecDSig = new \XMLSecurityDSig();
        $objXMLSecDSig->setCanonicalMethod(\XMLSecurityDSig::EXC_C14N);

        switch ($key->type) {
            case \XMLSecurityKey::RSA_SHA256:
                $type = \XMLSecurityDSig::SHA256;
                break;
            case \XMLSecurityKey::RSA_SHA384:
                $type = \XMLSecurityDSig::SHA384;
                break;
            case \XMLSecurityKey::RSA_SHA512:
                $type = \XMLSecurityDSig::SHA512;
                break;
            default:
                $type = \XMLSecurityDSig::SHA1;
        }
        $objXMLSecDSig->addReferenceList(
            array($root),
            $type,
            array('http://www.w3.org/2000/09/xmldsig#enveloped-signature',
\XMLSecurityDSig::EXC_C14N),
            array('id_name' => 'ID', 'overwrite' => false)
        );

        $objXMLSecDSig->sign($key);

        foreach ($certificates as $certificate) {
            $objXMLSecDSig->add509Cert($certificate, true);
        }
```

```
        $objXMLSecDSig->insertSignature($root, $insertBefore);
    }
```

- ***XMLSecurityDSig, XMLSecurityKey y XMLSecEnc***. Clases utilizadas para el trabajo con claves.

Una vez generado el ticket SAML se produce el envío de la petición:

```
    public function sendDefault(SAML2\SAML2_EidasAuthnRequest $message) {

              if ($this->destination === NULL) {
                    $destination = $message->getDestination();
              } else {
                    $destination = $this->destination;
              }

            if ($this->country !== NULL && $this->country !== "") {
              $destination = $destination /*. '?country=' . $this->country*/;
            }

              $relayState = $message->getRelayState();

                //This method sign the resulting XML document if the private key for the
signature is set.
              $msgStr = $message->toSignedXML();
              $msgStr = $msgStr->ownerDocument->saveXML($msgStr);

              SimpleSAML_Utilities::debugMessage($msgStr, 'out');

              $msgStr = base64_encode($msgStr);
              $msgStr = htmlspecialchars($msgStr);

              if ($message instanceof SAML2\SAML2_EidasAuthnRequest) {
                    $msgType = 'SAMLRequest';
              } else {
                    $msgType = 'SAMLResponse';
              }

              $destination = htmlspecialchars($destination);
```

```
            if ($relayState !== NULL) {
                  $relayState = '<input type="hidden" name="RelayState" value="' .
htmlspecialchars($relayState) . '">';
            } else {
                  $relayState = '';
            }
               /**
                * The output is the final SAML Authentication Request
                */
            $out = <<<END
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>POST data</title>
</head>
<body onload="document.forms[0].submit()">
<noscript>
<p><strong>Note:</strong> Since your browser does not support JavaScript, you must press the
button below once to proceed.</p>
</noscript>
<form method="post" action="$destination">
<input type="hidden" name="$msgType" value="$msgStr" />

<input type="hidden" name="country" value="ES"/>

$relayState
<noscript><input type="submit" value="Submit" /></noscript>
</form>
</body>
</html>
END;
            echo($out);
            exit(0);
      }
```

9.3.9 Validación de una Respuesta

Para la validación de una SAML Response, se utilizan las siguientes clases:

- **HTTPPost**. Subclase que extiende de *Binding*.
  - *receive*. Recibe un mensaje SAML 2 enviado una petición HTTP-POST binding.

```
public function receive()
    {
        if (array_key_exists('SAMLRequest', $_POST)) {
            $msg = $_POST['SAMLRequest'];
        } elseif (array_key_exists('SAMLResponse', $_POST)) {
            $msg = $_POST['SAMLResponse'];
        } else {
            throw new \Exception('Missing SAMLRequest or SAMLResponse parameter.');
        }

        $msg = base64_decode($msg);

        Utils::getContainer()->debugMessage($msg, 'in');

        $document = DOMDocumentFactory::fromString($msg);
        $xml = $document->firstChild;

        $msg = SAML2_Message::fromXML($xml);

        if (array_key_exists('RelayState', $_POST)) {
            $msg->setRelayState($_POST['RelayState']);
        }

        return $msg;
    }
```

- **eidas_saml_Message**: Se vuelve a utilizar un método de esta clase anteriormente definida, processResponse
  - *processResponse. Método encargado de procesar la respuesta.*

```
public static function processResponse(SimpleSAML_Configuration $splocalConfig,
SimpleSAML_Configuration $idplocalConfig, SAML2\Response $response) {
            if (!$response->isSuccess()) {
```

```
                    throw self::getResponseError($response);
                }
                /* Validate Response-element destination. */
                $currentURL = \SimpleSAML\Utils\HTTP::getSelfURLNoQuery();
                   if (VAR_DESA == 1) {
                        $cfg = \SimpleSAML_Configuration::getInstance();
                        $baseDir = $cfg->getBaseDir();
                         $cur_path = realpath($_SERVER['SCRIPT_FILENAME']);
                         // find the path to the current script relative to the www/ directory of
 SimpleSAMLphp
                         $rel_path     =     str_replace($baseDir.'www'.DIRECTORY_SEPARATOR,
                                       '',
 $cur_path);
                         // convert that relative path to an HTTP query
                         $url_path = str_replace(DIRECTORY_SEPARATOR, '/', $rel_path);
                         $currentURL = \SimpleSAML\Utils\HTTP::getSelfURLHost().'/'.$url_path;
                     }

                $msgDestination = $response->getDestination();
                if ($msgDestination !== NULL && $msgDestination !== $currentURL) {
                         throw new Exception('Destination in response doesn\'t match the current
 URL. Destination is "' .
                                 $msgDestination . '", current URL is "' . $currentURL . '".');
                }

    //  $responseSigned = self::checkSign($idplocalConfig, $response);

$responseSigned = self::checkSign($splocalConfig, $response);

                /*
    *   When we get this far, the response itself is valid.
    *   We only need to check signatures and conditions of the response. */

                $assertion = $response->getAssertions();
                if (empty($assertion)) {
                        throw  new  SimpleSAML_Error_Exception('No  assertions  found  in
                        response
```

```
 from IdP.');
                }

                $ret = array();

                foreach ($assertion as $a) {

                        $ret[] = self::processAssertion($splocalConfig, $idplocalConfig,
 $response, $a, $responseSigned);

                }

                        return $ret;
        }
```

- *checkSign*. Método encargado de la validación de la firma.

```
public      static      function checkSign(SimpleSAML_Configuration          $srclocalConfig,

SAML2\SignedElement $element) {

              /* Find the public key that should verify signatures by this entity. */

              $keys = $srclocalConfig->getPublicKeys('', false, 'validate.');

              if ($keys !== NULL) {

                     $pemKeys = array();

                     foreach ($keys as $key) {

                            switch ($key['type']) {

                            case 'X509Certificate':

                                   $pemKeys[] = "-----BEGIN CERTIFICATE-----\n" .

                                          chunk_split($key['X509Certificate'], 64) .

                                         "-----END CERTIFICATE-----\n";

                                  break;

                           default:

                                  SimpleSAML_Logger::debug('Skipping  unknown  key  type:  '  .
$key['type']);
```

```
                        }

                }

        } else {

                throw new SimpleSAML_Error_Exception(

                        'Missing certificate for ' .

                        var_export($srclocalConfig->getString('entityid'), TRUE));

        }


        SimpleSAML_Logger::debug('Has ' . count($pemKeys) . ' candidate keys for
validation.');


        $lastException = NULL;

        foreach ($pemKeys as $i => $pem) {

                $key = new XMLSecurityKey(XMLSecurityKey::RSA_SHA512,
array('type'=>'public'));
                $key->loadKey($pem);


                try {
                        /*

                         * Make sure that we have a valid signature on either the
response
    *   or the assertion. */

                        $res = $element->validate($key);

                        if ($res) {

                                SimpleSAML_Logger::debug('Validation with key #' . $i . '
succeeded.');


                                return TRUE;

                        }

                        SimpleSAML_Logger::debug('Validation with key #' . $i . ' failed
without exception.');
                } catch (Exception $e) {

                        SimpleSAML_Logger::debug('Validation with key #' . $i . ' failed
```

```
with exception: ' . $e->getMessage());
                            $lastException = $e;

                    }

            }


            /* We were unable to validate the signature with any of our keys. */ if
            ($lastException !== NULL) {

                    throw $lastException;

            } else {

                    return FALSE;

            }

}
```

- *processAssertion. Método privado encargado de validar la respuesta.*

```
private    static    function    processAssertion(SimpleSAML_Configuration
           $splocalConfig,

SimpleSAML_Configuration $idplocalConfig, SAML2\Response $response, $assertion,
$responseSigned) {


                assert('$assertion instanceof SAML2_Assertion || $assertion instanceof
SAML2_EncryptedAssertion');

             assert('is_bool($responseSigned)');


               if (VAR_DESA == 0) {


                   $assertion   =   self::decryptAssertion($idplocalConfig,
                                    $splocalConfig,
$assertion);


                   if (!self::checkSign($idplocalConfig,
                           $assertion)) { if (!$responseSigned) {

                                       throw new SimpleSAML_Error_Exception('Neither the
assertion nor the response was signed.');

                            }
```

```
                    }

                }


            /* At least one valid signature found. */


            $currentURL = \SimpleSAML\Utils\HTTP::getSelfURLNoQuery();


                if (VAR_DESA == 1){

                    $cfg =
                    \SimpleSAML_Configuration::getInstance();
                    $baseDir = $cfg->getBaseDir();

                    $cur_path = realpath($_SERVER['SCRIPT_FILENAME']);

                    // find the path to the current script relative to the www/ directory
                    of
SimpleSAMLphp

                    $rel_path    =    str_replace($baseDir.'www'.DIRECTORY_SEPARATOR,
                                 '',

$cur_path);

                    // convert that relative path to an HTTP query
                    $url_path = str_replace(DIRECTORY_SEPARATOR, '/', $rel_path);

                    $currentURL = \SimpleSAML\Utils\HTTP::getSelfURLHost().'/'.$url_path;

                }


            /* Check various properties of the assertion. */


            $notBefore = $assertion->getNotBefore();

            if ($notBefore !== NULL && $notBefore > time() + 60) {

                   throw new SimpleSAML_Error_Exception('Received an assertion that is
valid in the future. Check clock synchronization on IdP and SP.');
            }


            $notOnOrAfter = $assertion->getNotOnOrAfter();
            if ($notOnOrAfter !== NULL && $notOnOrAfter <= time() - 60) {

                   throw new SimpleSAML_Error_Exception('Received an assertion that has
expired. Check clock synchronization on IdP and SP.');
            }
```

```
            $sessionNotOnOrAfter = $assertion->getSessionNotOnOrAfter();

            if ($sessionNotOnOrAfter !== NULL && $sessionNotOnOrAfter <= time() - 60) {
                  throw new SimpleSAML_Error_Exception('Received an assertion with a

session that has expired. Check clock synchronization on IdP and SP.');

}
            $validAudiences = $assertion->getValidAudiences();
            if ($validAudiences !== NULL) {

                  $spEntityId = $splocalConfig->getString('entityid');
                  if (!in_array($spEntityId, $validAudiences, TRUE)) {
                        $candidates = '[' . implode('], [', $validAudiences) . ']';

                        throw new SimpleSAML_Error_Exception('This SP [' . $spEntityId
. '] is not a valid audience for the assertion. Candidates were: ' . $candidates);
                  }

            }


              $found = FALSE;
            $lastError = 'No SubjectConfirmation element in Subject.';

            $validSCMethods = array(SAML2\Constants::CM_BEARER, SAML2\Constants::CM_HOK,
SAML2\Constants::CM_VOUCHES);

            foreach ($assertion->getSubjectConfirmation() as
                  $sc) { if (!in_array($sc->Method,
                  $validSCMethods)) {

                        $lastError = 'Invalid Method on SubjectConfirmation: ' .
var_export($sc->Method, TRUE);

                        continue;

                  }


                  /* Is SSO with HoK enabled? IdP remote metadata overwrites SP
                  metadata

configuration. */

                  $hok = $idplocalConfig->getBoolean('saml20.hok.assertion',
                  NULL); if ($hok === NULL) {

                        $hok = $splocalConfig->getBoolean('saml20.hok.assertion',
                        FALSE);

                  }
                  if ($sc->Method === SAML2\Constants::CM_BEARER && $hok) {
```

```
                        $lastError = 'Bearer SubjectConfirmation received, but Holder-
of-Key SubjectConfirmation needed';
                        continue;

                }

                if ($sc->Method === SAML2\Constants::CM_HOK && !$hok) {

                        $lastError = 'Holder-of-Key SubjectConfirmation received, but
the Holder-of-Key profile is not enabled.';

                        continue;

                }


                $scd = $sc->SubjectConfirmationData;

                if ($sc->Method === SAML2\Constants::CM_HOK) {
                        /* Check HoK Assertion */

                        if (\SimpleSAML\Utils\HTTP::isHTTPS() === FALSE) {

                                $lastError = 'No HTTPS connection, but required
                                        for

Holder-of-Key SSO';

                                continue;

                        }

                        if (isset($_SERVER['SSL_CLIENT_CERT']) &&
empty($_SERVER['SSL_CLIENT_CERT'])) {

                                $lastError = 'No client certificate provided during TLS

Handshake with SP';

                                continue;

                        }

                        /* Extract certificate data (if this is a certificate).
                        */ $clientCert = $_SERVER['SSL_CLIENT_CERT'];

                        $pattern = '/^-----BEGIN CERTIFICATE-----([^-]*)^-----
                        END

CERTIFICATE-----/m';

if (!preg_match($pattern, $clientCert, $matches)) {

                                $lastError = 'Error while looking for client
                                certificate

during TLS handshake with SP, the client certificate does not '
```

```
                                                    . 'have the expected structure';

                                    continue;
                            }

                            /* We have a valid client certificate from the browser. */

                            $clientCert   =   str_replace(array("\r",   "\n",   "   "),
                                              '',
$matches[1]);


                            foreach ($scd->info as $thing) {

                                    if($thing instanceof SAML2_XML_ds_KeyInfo) {

                                            $keyInfo[]=$thing;

                                    }

                            }
                            if (count($keyInfo)!=1) {

$lastError                                          =       'Error validating    Holder-of-Key
                                                    assertion:

Only one <ds:KeyInfo> element in <SubjectConfirmationData>
                                                    allowed'; continue;

                            }


foreach ($keyInfo[0]->info as $thing) {
                                    if($thing instanceof SAML2_XML_ds_X509Data) {

                                            $x509data[]=$thing;

                                    }
                            }

                            if (count($x509data)!=1) {

$lastError                                          =       'Error validating    Holder-of-Key
                                                    assertion:

Only one <ds:X509Data> element in <ds:KeyInfo> within <SubjectConfirmationData>
                                                    allowed'; continue;

                            }


foreach ($x509data[0]->data as $thing) {
```

```
                                if($thing instanceof
                                        SAML2_XML_ds_X509Certificate) {
                                        $x509cert[]=$thing;

                                }

                        }

                        if (count($x509cert)!=1) {

                                $lastError = 'Error validating Holder-of-Key assertion:
Only one <ds:X509Certificate> element in <ds:X509Data> within <SubjectConfirmationData>
allowed';

                                continue;
                        }


                        $HoKCertificate = $x509cert[0]->certificate;
                        if ($HoKCertificate !== $clientCert) {

                                $lastError = 'Provided client certificate does not match
the certificate bound to the Holder-of-Key assertion';

                                continue;

                        }

                }


   // if no SubjectConfirmationData then don't do anything. if ($scd ===
      null) {

                        $lastError = 'No SubjectConfirmationData
                        provided'; continue;

}
                if ($scd->NotBefore && $scd->NotBefore > time() + 60) {
                        $lastError = 'NotBefore in SubjectConfirmationData is in the
future: ' . $scd->NotBefore;

                        continue;

                }
                if ($scd->NotOnOrAfter && $scd->NotOnOrAfter <= time() - 60) {

                        $lastError = 'NotOnOrAfter in SubjectConfirmationData is in the

past: ' . $scd->NotOnOrAfter;

                        continue;

                }

                if ($scd->Recipient !== NULL && $scd->Recipient !== $currentURL) {
```

```
                        $lastError = 'Recipient in SubjectConfirmationData does not
                        match
the current URL. Recipient is ' .

                                var_export($scd->Recipient, TRUE) . ', current URL is '
. var_export($currentURL, TRUE) . '.';
                        continue;

                }

                if ($scd->InResponseTo !== NULL && $response->getInResponseTo() !==
NULL && $scd->InResponseTo !== $response->getInResponseTo()) {

                        $lastError = 'InResponseTo in SubjectConfirmationData does not
match the Response. Response has ' .

                                var_export($response->getInResponseTo(),    TRUE)    .
                                                                                ',

SubjectConfirmationData has ' . var_export($scd->InResponseTo, TRUE) .
                        '.'; continue;

                }
                $found = TRUE;

                break;

        }
        if (!$found) {

                throw new SimpleSAML_Error_Exception('Error validating
SubjectConfirmation in Assertion: ' . $lastError);
        }


        /* As far as we can tell, the assertion is valid. */


        /* Maybe we need to base64 decode the attributes in the assertion?
        */ if ($idplocalConfig->getBoolean('base64attributes', FALSE)) {

                $attributes = $assertion->getAttributes();
                $newAttributes = array();

                foreach ($attributes as $name => $values) {

                        $newAttributes[$name] = array();
                        foreach ($values as $value) {

                                foreach(explode('_', $value) AS $v) {

                                        $newAttributes[$name][] = base64_decode($v);
```

```
                                }
                        }

                }

                $assertion->setAttributes($newAttributes);
        }



        /* Decrypt the NameID element if it is encrypted.
        */ if ($assertion->isNameIdEncrypted()) {

                try {
                        $keys = self::getDecryptionKeys($idplocalConfig,
                        $splocalConfig);

                } catch (Exception $e) {

                        throw new SimpleSAML_Error_Exception('Error decrypting NameID:
                        '

. $e->getMessage());

}
```

- **Crypto.** Clase que describe algunos de los métodos relacionados con la criptografía.
- *loadPrivateKey.* Carga una clave privada de los metadatos.
- **XMLSecurityKey.** Clase perteneciente a la librería publica xmlseclibs anteriormente descrita y utilizada en el tratamiento de claves de seguridad y XML. En este caso se crea un nuevo objeto de esta clase para intervenir en la parte de descifrado del mensaje.
    - *RSA_OAEP_MGF1P*. Constante que indica el algoritmo a usar para descifrar el mensaje.
    - *LoadKey.* Método que carga la clave que cumpla con los parámetros de entrada indicados.
- **EncryptedAssertion.** Clase encargada de manejar las respuestas encriptadas.
    - *getAssertion.* Método para extraer la respuesta.
- **Utils**. Clase previamente descrita.
    - *decryptElement* y *doDecryptElement*. Métodos utilizados para descifrar el mensaje.

```
//Obtener las assertions en caso de qe sean un objeto EncryptedAsserton, es decir, descifrar
las assertions
        if ($assertions[0] instanceof SAML2\EncryptedAssertion) {
            try {


                                            $keyArray =
SimpleSAML\Utils\Crypto::loadPrivateKey($metadata, TRUE);
```

```
                                        assert('isset($keyArray["PEM"])');

                                        $key = new
XMLSecurityKey(XMLSecurityKey::RSA_OAEP_MGF1P, array('type'=>'private'));
                                        if (array_key_exists('password', $keyArray)) {
                                                $key->passphrase =
$keyArray['password'];
                                        }
                                        $key->loadKey($keyArray['PEM']);

                                        $assertions[0] = $assertions[0]-
>getAssertion($key, array());


                                    }
```

Por último, el método que se encarga en última instancia de descifrar el mensaje es *doDecryptElement* cuyos parámetros de entrada son, los datos encriptados, la clave necesaria y un parámetro opcional que permite introducir una lista negra de algoritmos de encriptación.

```
//Obtener las assertions en caso de qe sean un objeto EncryptedAsserton, es decir, descifrar
las assertions


private static function doDecryptElement(\DOMElement $encryptedData, \XMLSecurityKey
$inputKey, array &$blacklist)
    {
        $enc = new \XMLSecEnc();


        $enc->setNode($encryptedData);
        $enc->type = $encryptedData->getAttribute("Type");
       //  $encryptedNode = $encryptedData->firstChild;
        $symmetricKey = $enc->locateKey($encryptedData);


        if (!$symmetricKey) {
            throw new \Exception('Could not locate key algorithm in encrypted data.');
        }


        $symmetricKeyInfo = $enc->locateKeyInfo($symmetricKey);
        if (!$symmetricKeyInfo) {
            throw new \Exception('Could not locate <dsig:KeyInfo> for the encrypted key.');
        }
```

```
        $inputKeyAlgo = $inputKey->getAlgorith();
        if ($symmetricKeyInfo->isEncrypted) {
            $symKeyInfoAlgo = $symmetricKeyInfo->getAlgorith();

            if (in_array($symKeyInfoAlgo, $blacklist, true)) {
                throw new \Exception('Algorithm disabled: ' . var_export($symKeyInfoAlgo,
true));
            }

            if ($symKeyInfoAlgo === \XMLSecurityKey::RSA_OAEP_MGF1P && $inputKeyAlgo ===
\XMLSecurityKey::RSA_1_5) {
                /*
                 * The RSA key formats are equal, so loading an RSA_1_5 key
                 * into an RSA_OAEP_MGF1P key can be done without problems.
                 * We therefore pretend that the input key is an
                 * RSA_OAEP_MGF1P key.
                 */
                $inputKeyAlgo = \XMLSecurityKey::RSA_OAEP_MGF1P;
            }

            /* Make sure that the input key format is the same as the one used to encrypt
the key. */
            if ($inputKeyAlgo !== $symKeyInfoAlgo) {
                throw new \Exception(
                    'Algorithm mismatch between input key and key used to encrypt ' .
                    ' the symmetric key for the message. Key was: ' .
                    var_export($inputKeyAlgo, true) . '; message was: ' .
                    var_export($symKeyInfoAlgo, true)
                );
            }

            /** @var XMLSecEnc $encKey */
            $encKey = $symmetricKeyInfo->encryptedCtx;
            $symmetricKeyInfo->key = $inputKey->key;

            $keySize = $symmetricKey->getSymmetricKeySize();
            if ($keySize === null) {
                /* To protect against "key oracle" attacks, we need to be able to create a
symmetric key, and for that we need to know the key size.
```

```
                 */
                throw new \Exception('Unknown key size for encryption algorithm: ' .
var_export($symmetricKey->type, true));
            }


            try {
                $key = $encKey->decryptKey($symmetricKeyInfo);
                if (strlen($key) != $keySize) {
                    throw new \Exception(
                        'Unexpected key size (' . strlen($key) * 8 . 'bits) for encryption
algorithm: ' .
                        var_export($symmetricKey->type, true)
                    );
                }
            } catch (\Exception $e) {
                /* We failed to decrypt this key. Log it, and substitute a "random" key. */
                Utils::getContainer()->getLogger()->error('Failed to decrypt symmetric key:
' . $e->getMessage());
                /* Create a replacement key, so that it looks like we fail in the same way
as if the key was correctly padded. */


                /* We base the symmetric key on the encrypted key and private key, so that
we always behave the same way for a given input key.
                 */
                $encryptedKey = $encKey->getCipherValue();
                $pkey = openssl_pkey_get_details($symmetricKeyInfo->key);
                $pkey = sha1(serialize($pkey), true);
                $key = sha1($encryptedKey . $pkey, true);


                /* Make sure that the key has the correct length. */
                if (strlen($key) > $keySize) {
                    $key = substr($key, 0, $keySize);
                } elseif (strlen($key) < $keySize) {
                    $key = str_pad($key, $keySize);
                }
            }
            $symmetricKey->key=$key;
        } else {
            $symKeyAlgo = $symmetricKey->getAlgorith();
            /* Make sure that the input key has the correct format. */
```

```
            if ($inputKeyAlgo !== $symKeyAlgo) {
                throw new \Exception(
                    'Algorithm mismatch between input key and key in message. ' .
                    'Key was: ' . var_export($inputKeyAlgo, true) . '; message was: ' .
                    var_export($symKeyAlgo, true)
                );
            }
            $symmetricKey = $inputKey;
        }

        $algorithm = $symmetricKey->getAlgorith();
        if (in_array($algorithm, $blacklist, true)) {
            throw new \Exception('Algorithm disabled: ' . var_export($algorithm, true));
        }

        /** @var string $decrypted */
        $decrypted = $enc->decryptNode($symmetricKey, false);

        /*
         * This is a workaround for the case where only a subset of the XML
         * tree was serialized for encryption. In that case, we may miss the
         * namespaces needed to parse the XML.
         */
        $xml = '<root xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" '.
                     'xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">' .
            $decrypted .
            '</root>';

        try {
            $newDoc = DOMDocumentFactory::fromString($xml);
        } catch (RuntimeException $e) {
            throw new \Exception('Failed to parse decrypted XML. Maybe the wrong sharedkey
was used?', 0, $e);
        }

        $decryptedElement = $newDoc->firstChild->firstChild;
        if ($decryptedElement === null) {
            throw new \Exception('Missing encrypted element.');
        }
```

```
        if (!($decryptedElement instanceof \DOMElement)) {
            throw new \Exception('Decrypted element was not actually a \DOMElement.');
        }

        return $decryptedElement;
```

9.3.9.1 *Validación del Período de Validez del Ticket SAML*

El código proporcionado valida el período de validez del tiquet SAML si está informado. Es decir, valida que la fecha actual este entre los atributos NotBefore y NotOnOrAfter del tiquet SAML.

Para que esta validación sea efectiva el servidor del SP y el servidor de la plataforma Cl@ve tiene que estar sincronizados.

El código proporcionado aplica un 1 minuto de clockskew para evitar desfases entre las horas de los servidores. Ver las clases NotBefore y NotOnOrAfter del namespace SAML2\Assertion\Validation\ConstraintValidator para mas detalle.

9.3.10 Validación de una Respuesta de Logout

Una vez realizada una petición de logout, el endpoint recibirá una respuesta y en **logoutResponse.php** se procesará la misma.

Como se ve a continuación, se validará la firma de manera similar a una SAML Reponse y se devolverá error en caso de que la respuesta sea inválida, o se redirigirá al usuario a la página principal para que pueda iniciar sesión de nuevo:

```
          try {
        $retVal = eidas_saml_Message::checkSign($localConfig, $response);

        if (!$response->isSuccess() || !$retVal) {
                $statsData['error'] = $response->getStatus();
                throw Exception("LogoutResponse is Fail or invalid");
        }
        $relayState = $response->getRelayState();
        header ('location: /SP');
    } catch (Exception $e) { // TODO: look for a specific exception
            echo "<h2>Se ha producido un error</h2><br>Por favor, haga clic <a
href=\"/SP/\">aqu&iacute;</a> para volver a la p&aacute;gina principal.";
            throw $e; // do not ignore other exceptions!
    }
```

9.3.11 Actualización Automática de Certificados

Saml2UpdateCert es un proceso PHP que permite descargar los certificados de firma y cifrado de la pasarela Cl@ve y persistirlos en el sistema de ficheros del SP.

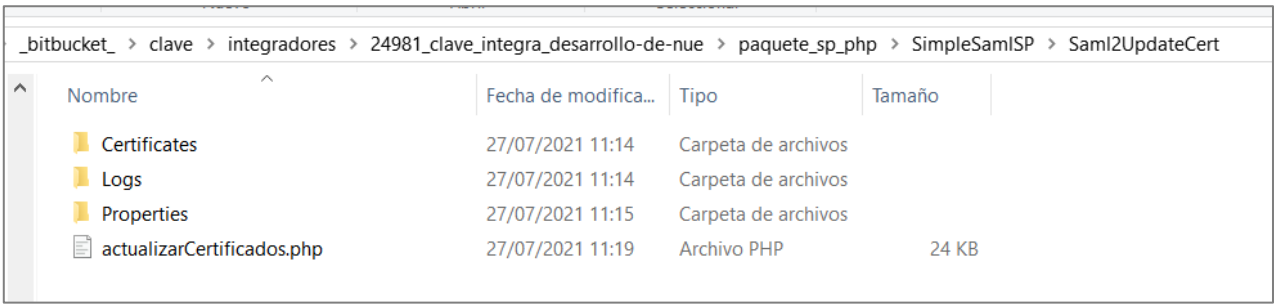


Ilustración 17: Paquete de Integración PHP – Proceso Saml2UpdateCert

9.3.11.1 Configuración

En el fichero \Saml2UpdateCert**actualizarCertificados.php**, en la variable constante **PROPERTIES_PATH** se india la ruta del fichero de configuración **certproxy.properties**.

```
// ------------------------------------------------------------------------
// Estas dos constantes hacen referencia a la ruta y al archivo desde el que se
// va a leer el fichero de configuración que estblecerá el resto de los datos.
// ------------------------------------------------------------------------
const PROPERTIES_PATH = "/SimpleSamlSP/Saml2UpdateCert/Properties";
const PROPERTIES_FILE = "certproxy.properties";
// ------------------------------------------------------------------------
```

Ilustración 18: Paquete de Integración PHP – Configuración Fichero certproxy.properties

En el fichero de configuración **certproxy.properties** de la aplicación se definen las siguientes entradas:

- *certproxy2_processActivated:* Activar o desactivar el proceso.
- *certproxy2_endpoint*: endpoint del servicio de la pasarela desde donde se descarga el XML que contiene la parte pública de los certificados.
- *certproxy2_certificatesPath*: ubicación del sistema de ficheros del SP donde se guardarán los certificados con extensión *.cer*.

  La identidad con la que se ejecuta la aplicación debe tener permisos de lectura/escritura sobre esta ubicación.

- *certproxy2_logsPath*: ubicación de los ficheros de log.

  La identidad con la que se ejecuta la aplicación debe tener permisos de lectura/escritura sobre esta ubicación.

9.3.11.2 Proceso

El proceso **actualizarCertificados.php** implementa el proceso de actualización de certificados llevando a cabo las siguientes acciones:

- Obtener el XML de la pasarela Cl@ve
- Extraer la información de certificados del XML
- Crear, en el caso de que no exista, el directorio donde se han de persistir los certificados
- Leer el fichero de control de certificados persistidos (propiedad certproxy2_certificatesFile)
- Para cada certificado definido en el XML que no este persitido en local o que el estado sea diferente:
    - Persistir el certificado en la ubicación definida
- Actualizar el fichero de control de certificados persistidos. De manera que la siguiente vez que se ejecute solo actualiza los certificados nuevos o que han cambiado de estado.
  Este archivo almacena, para cada certificado, la siguiente información:
    - *Name*: nombre del certificado tal como se especifica en el XML.
    - *Use*: "*Signing*" si el certificado se usa para firma o "*Encryption*" si se usa para cifrado.
    - *Active*: booleano que indica el estado del certificado (si está activo o no) en la pasarela.
    - *DownloadDate*: fecha de descarga del certificado en formato ISO 8601.

9.3.11.3 Planificación del Proceso

Si se desea que el proceso de actualización se realice con una determinada periodicidad se puede ejecutar mediante un lanzador de procesos, como por ejemplo el *Planificador de Tareas de Windows*.

La sentencia de ejecución (que se debería guardar en un archivo .bat para ejecutarse desde una tarea programada del sistema) es la siguiente:

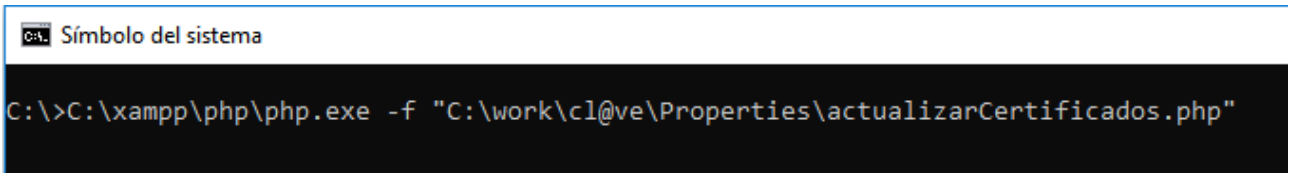


Ilustración 19: Paquete de Integración PHP – Comando Ejecución Proceso Saml2UpdateCert

donde:

- `C:\xampp\php\php.exe`: Es la ruta donde se encuentra el ejecutable de PHP.
- `C:\work\cl@ve\Properties\actualizarCertificados.php`: Es la ruta donde se encuentra el fichero PHP que realiza la tarea de actualización de los certificados.

10 Pasos Durante la Ejecución de una Prueba

Al acceder al SP de demo desde el navegador se le muestra al usuario una ventana con un botón: “Iniciar sesión”.

Ilustración 20: Pruebas – Interfaz de Inicio de Sesión (Demo SP)

En caso de que se pulse sobre la opción “Iniciar sesión”, el programa iniciará automáticamente el proceso de enviar una petición SAML por defecto sin que sea necesario intervenir. Esta petición solicitará los atributos requeridos como atributos obligatorios y el resto como opcionales, excluyendo los atributos de representación.

Una vez que el usuario selecciona los atributos deseados, habrá que presionar el botón “Iniciar sesión”, que llevará al usuario hacia la pasarela Clave.

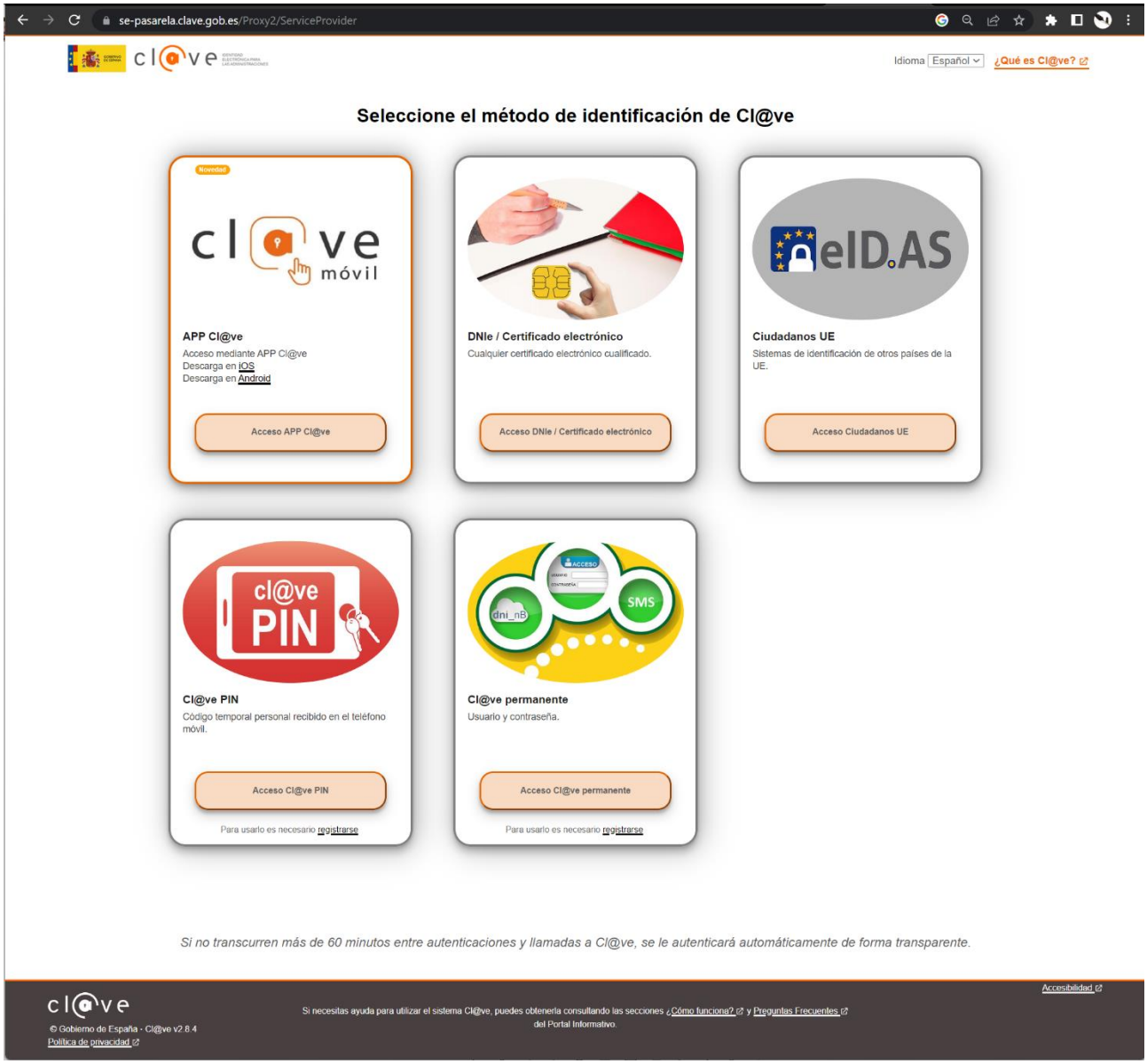


Ilustración 21: Pruebas – Pasarela (Selección de IdP)

Tras realizar el proceso de identificación, y tras recibir la respuesta, se muestra una página en la que, tras validar el ticket recibido, se muestra la página final del ciclo, donde se debería continuar con el flujo de negocio normal de una aplicación. En el ejemplo los valores recabados.

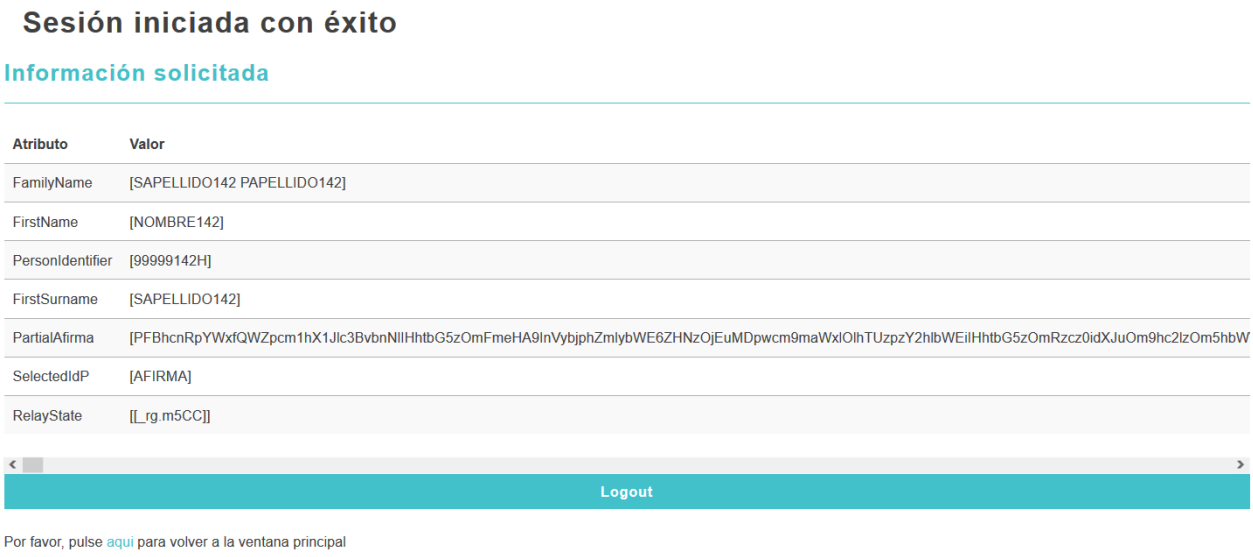


Ilustración 22: Pruebas – Interfaz de Inicio de Sesión Correcto (Demo SP)

En caso de error se mostrará una página descriptiva del problema recibido.

Ilustración 23: Pruebas – Interfaz de Inicio de Sesión Incorrecto (Demo SP)

10.1 Validar Petición

En caso de haber recibido una solicitud, se sigue el siguiente flujo lógico.

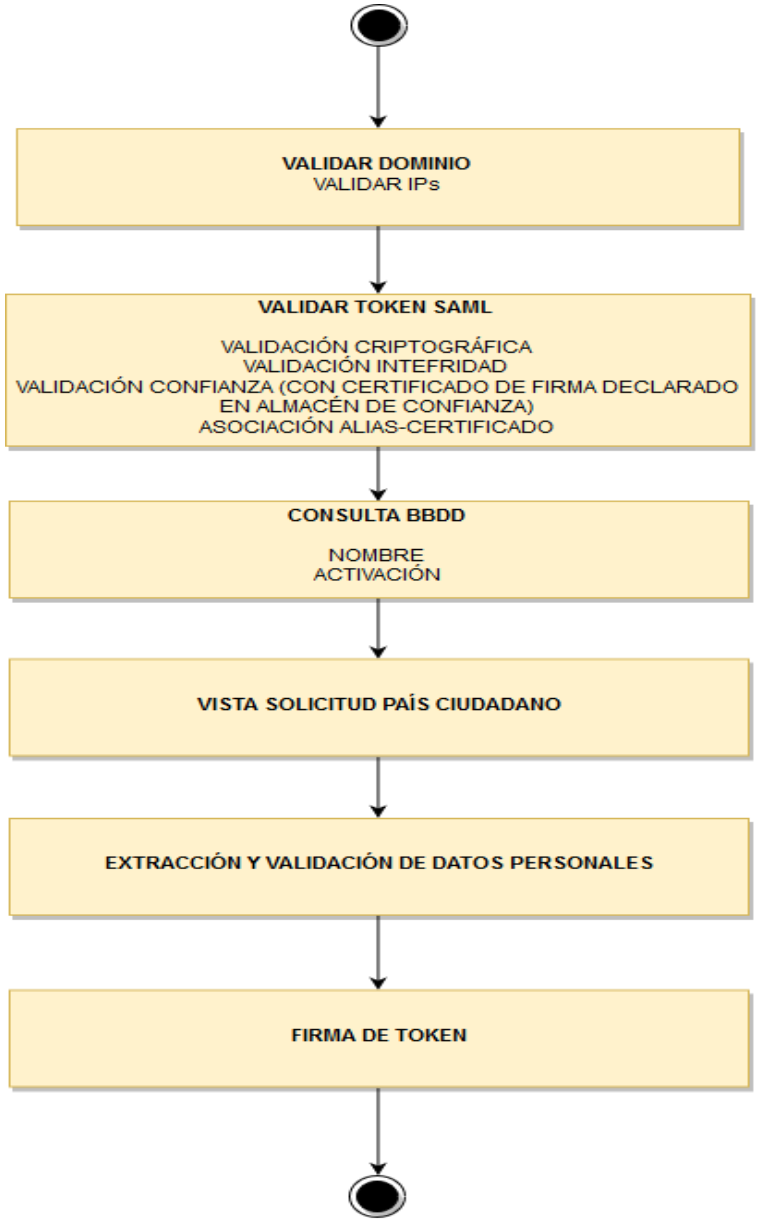


Ilustración 24: Cl@ve 2 - Procesamiento de una Petición SAML Emitida por un SP

10.2 Validar Respuesta

En caso de haber recibido una respuesta SAML, se procesa del modo descrito a continuación:

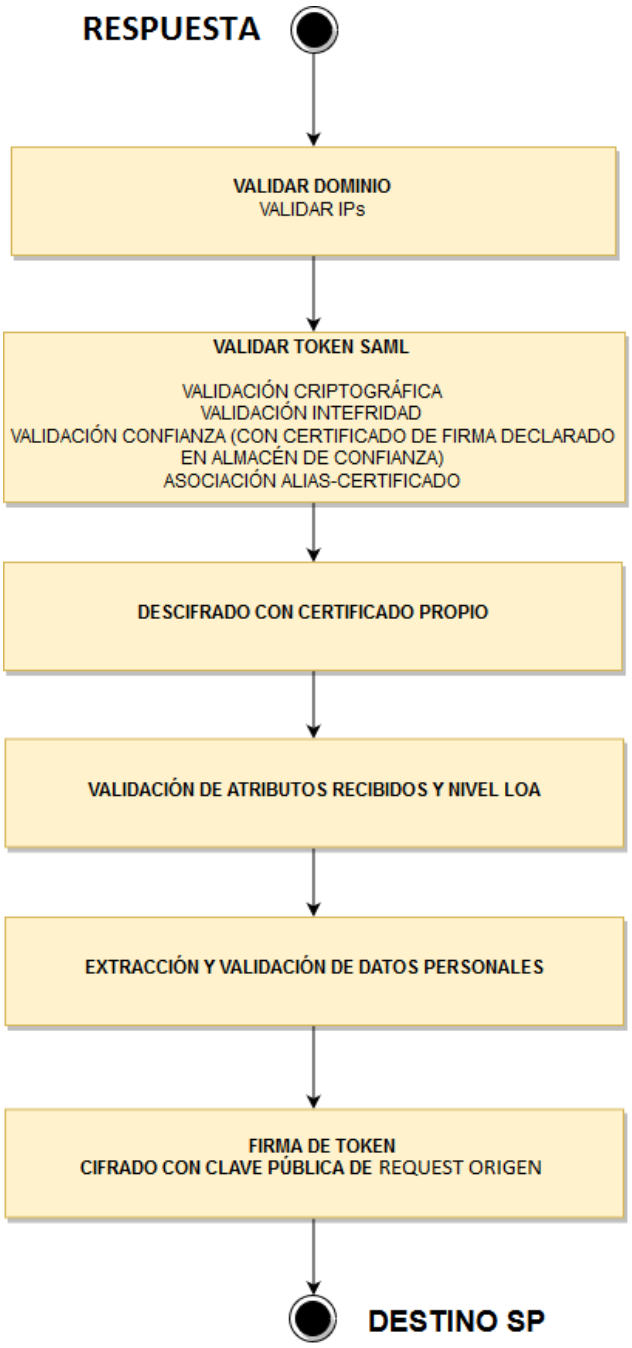


Ilustración 25: Cl@ve - Procesamiento de una Respuesta SAML Emitida por un IdP

11 Información de Conexión a la Pasarela Clave

11.1 Servicios Estables (SE)

Servicios Estables (también conocido como SE) se considera un entorno de pruebas conectado a Internet. La URL de servicio de la Pasarela Cl@ve es

https://se-pasarela.clave.gob.es/Proxy2/ServiceProvider

11.2 Producción (PRO)

La URL de servicio de la Pasarela Cl@ve es

https://pasarela.clave.gob.es/Proxy2/ServiceProvider

12 Anexo

En este apartado se describirán los procesos de firma y cifrado de ficheros XML, así como de metadatos, con el protocolo SAML2.

Para la firma de un XML, se utiliza la clave privada del mismo que los crea, lo que incluye un elemento en el XML llamado *Signature* que contiene una serie de elementos como es el *DigestMethod*, que contiene un atributo *Algorithm* el cual contiene el método que se ha utilizado para firmar (por ejemplo, el atributo podría ser: *Algorithm =http://www.w3.org/2001/04/xmlenc#sha512*, lo cual nos dice que se ha firmado con sha512).

Si se diese el caso de que alguien modificase el XML después de realizar la firma, ésta ya no sería válida, ya que los datos sobre los que se firmó son diferentes a los que hay una vez modificado el XML.

```
− <md:EntityDescriptor entityID="https://se-eidas2.redsara.es/EidasNode/ConnectorResponderMetadata" validUntil="2017-09-29T07:52:42.214Z">
  − <ds:Signature>
    − <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha512"/>
      − <ds:Reference URI="">
        + <ds:Transforms></ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>
        − <ds:DigestValue>
            DGu9FLTmvkrHFUXrYV3OAD2hfFFa8UFB3tdbiHlxNGItSzyiK/SljuhMSb9mnAKnWcK/M+zynUPR8qEbi6pbxg==
          </ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>
    + <ds:SignatureValue></ds:SignatureValue>
    + <ds:KeyInfo></ds:KeyInfo>
    </ds:Signature>
  + <md:Extensions></md:Extensions>
  + <md:IDPSSODescriptor WantAuthnRequestsSigned="true" protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol"></md:IDPSSODescriptor>
  + <md:Organization></md:Organization>
  + <md:ContactPerson contactType="support"></md:ContactPerson>
  + <md:ContactPerson contactType="technical"></md:ContactPerson>
  </md:EntityDescriptor>
```

El proceso para comprobar la firma será el siguiente:

- Ver el algoritmo que se utiliza, y si usa el mismo un extremo que el otro se continúa.
- Se recoge el nodo X509Certificate, que es la clave pública del extremo firmante. Si no está incluida en el almacén de claves de CA's de confianza, no se continúa.
- Con ella, se comprueba la firma de la parte del XML firmada.
- El siguiente paso es comprobar que soporta todos los atributos obligatorios para personas físicas y jurídicas.

Una vez llegados a este punto con todas las validaciones, se puede dar por terminado el método de comprobar una firma.

El proceso para cifrar un XML, se puede elegir cifrar sólo una parte del mismo o todo él dejando en el atributo *URI* del nodo *Reference*. Si se deja vacío, se cifrará todo el XML.

Se coge la clave pública del otro extremo y se cifra el XML.

Una vez cifrado, no se pueden realizar cambios sobre el mismo, ya que el cifrado no valdría.

El proceso para firmar un XML es el siguiente:

- Se crea el XML con el protocolo SAML2.
- Se cifra el XML (completo, o sólo una parte del mismo) con la clave pública del otro extremo.
- Se firma ese XML firmado con la propia clave privada.
- Una vez hecho esto, se puede enviar al otro extremo.
- El proceso si el XML está firmado y cifrado, primero se comprobará la firma con los pasos ya explicados y después se descifrará. Es decir, el proceso completamente contrario. Si primero si cifra y después se firma, aquí se comprueba la firma y después se descifra el contenido para obtener el XML en claro.

12.1 Algoritmos Criptográficos Empleados

En este apartado se va a hablar de cómo se firman y cifran XML con el protocolo SAML2 y metadatos, explicando los algoritmos elegidos y cómo funcionan.

12.1.1 Cálculo de Resumen (Digest)

Un algoritmo de digest consiste en una función matemática que obtiene un resumen de los datos procesados. El resultado se llama “digest‟, “hash” o huella digital. El valor calculado debe ser único, es decir, no debe existir una entrada diferente que dé como resultado el mismo digest. Se utiliza para obtener un identificador único de tamaño fijo de un recurso y para evidenciar si se ha producido un cambio en los datos.

- Se emplean estándares definidos por NIST.
- Se asocia a la integridad de los datos.
- Algoritmos y familias frecuentes:
    - MD-5, SHA-1, SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512) y el nuevo SHA-3.
    - Existen ataques conocidos para MD-5 y SHA-1 que hacen que su uso no esté recomendado.
- **Cl@ve utiliza el algoritmo SHA-512**.

12.1.2 Cifrado Simétrico

En el cifrado simétrico se aplican una serie de transformaciones a los datos a proteger. Estas transformaciones consisten en permutaciones y transposiciones realizadas secuencialmente en función de una condición previa, es decir, de una clave, de tal forma que el proceso sea irreversible sin conocer la clave utilizada. Para cifrar y descifrar la información se utiliza la misma clave.

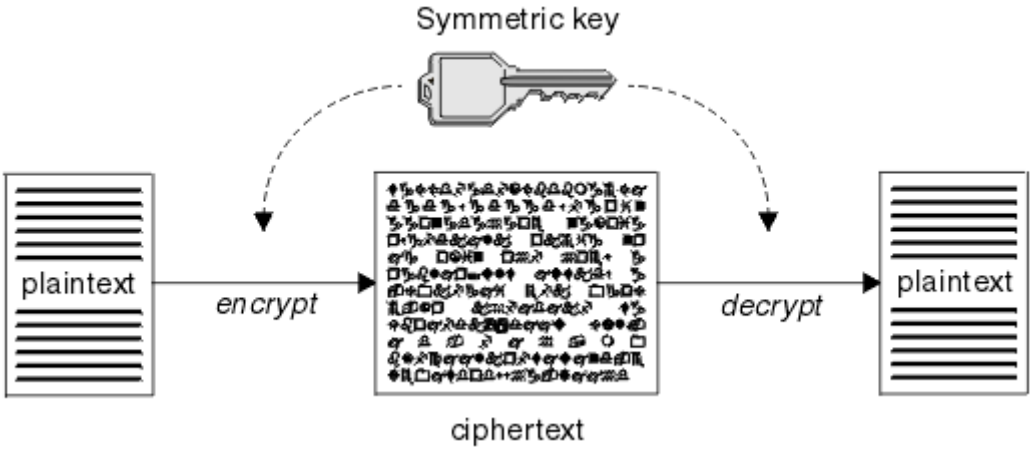


Ilustración 26; Cifrado Simétrico

12.1.3 Cifrado Simétrico de Tipo AES (Rijndael)

Se trata del estándar de cifrado adoptado por el protocolo eIDAS. Soporta longitudes de clave de 128, 192 y 256 bits. Es el sustituto de Data Encryption Standard (DES). Y está ampliamente estudiado y analizado por la comunidad internacional. Uno de sus puntos fuertes para su elección fue el rendimiento. Su definición se encuentra en el FIPS-197 y actualmente se considera el algoritmo de criptografía simétrica más extendido. Principales características:

- Procesa la información en bloques de 128 bits.
- Basado en permutaciones y substituciones utilizando un esquema matricial de 4x4.
- La clave define las operaciones de permutación y sustitución que se realizan durante el cifrado.
- La longitud de clave está ligada al número de operaciones intermedias que se realizan
    - AES128 → 10 ciclos de repetición
    - AES196 → 12 ciclos de repetición
    - AES256 → 14 ciclos de repetición

Es remarcable llegados a este punto que, en contra de lo que podría suponerse, aplicar un cifrado simétrico fuerte no es suficiente, ya que la información cifrada puede contener patrones que un atacante podría explotar para romper el cifrado. Este punto se puede apreciar en la siguiente imagen, que compara un cifrado crudo y un cifrado bien inicializado, alimentado de ruido, etc.

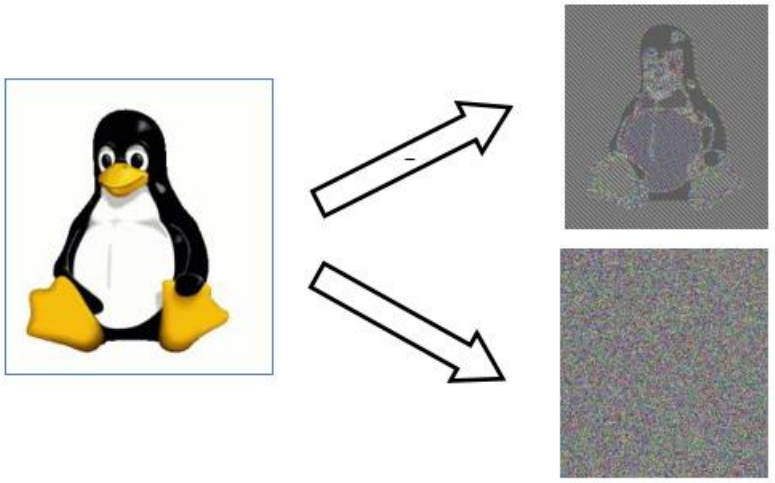


Ilustración 27: Cifrado Simétrico AES – Inicialización de Cifrado

Para inicializar correctamente el cifrado se introducen los siguientes conceptos asociados:

- Vector de inicialización (IV): estado inicial del cifrador. Es un valor aleatorio.
- Existen requisitos sobre los IVs en función del modo de cifrado.
  - Los IVs deben ser conocidos por ambas partes para poder descifrar la información
  - Los IVs no necesariamente tienen que ser secretos. En muchas ocasiones forman parte de las cabeceras de los protocolos.
- Padding: relleno para completar el tamaño de bloque. Hay que ser cuidadoso al aplicar el padding porque puede introducir ciertos patrones en los datos.
- Nonce: valor único para cada ejecución.

12.1.4 Modo de Cifrado GCM (Galois Counter Mode)

Este es el módulo de cifrado utilizado por Cl@ve para cifrar aserciones. En este modo de cifrado, además de cifrar con AES y RSA, se calcula una etiqueta de autenticación de los datos que permite validar la autenticidad.

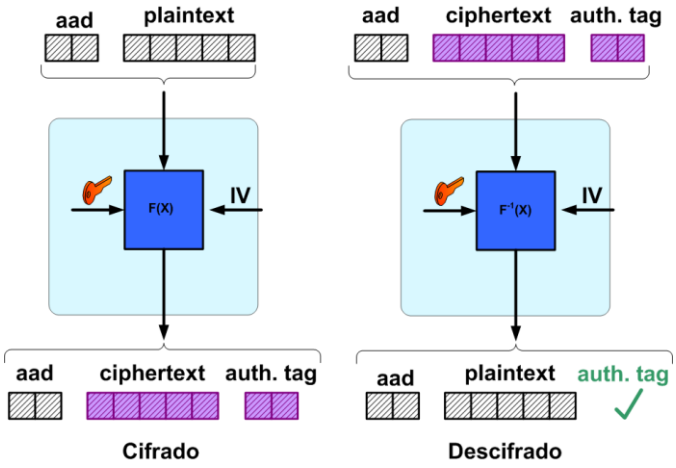


Ilustración 28: Cifrado GCM

Características:

- Procesado por paquetes.
- Soporta distintas longitudes para la etiqueta de autenticación.
- Necesario un IV por paquete.
- Longitud recomendada IV: 96 bits
- IVs no pueden repetirse para una misma clave.

12.1.5 Cifrado Asimétrico

El cifrado asimétrico tiene un objetivo similar al cifrado simétrico, pero divide la clave utilizada en dos partes. Lo que se cifra con una parte, solo puede ser descifrado con la otra parte, y viceversa. Su implementación se basa en problemas de la Teoría de Números y su fortaleza reside en la dificultad a la hora de factorizar grandes números enteros. Dicho de otro modo:

- Dados dos números primos grandes M y N es fácil hallar su producto P=MxN.
- Pero dado P es muy difícil hallar sus factores M y N. Se consideran problemas computacionalmente inabordables incluso con gran cantidad de recursos de potencia de cálculo y memoria.
- Actualmente un número de 1024 bits (≈310 dígitos) no puede ser factorizado:
    - Se estima que se necesitaría una máquina específicamente diseñada, con un coste >10M $, que tardaría un año en realizar la factorización de un solo número.

Esta clave en dos partes se utiliza para definir dos tipos de clave:

- Clave privada: conocida y custodiada únicamente por la entidad propietaria.

- Clave pública: publicada por la entidad propietaria y que cualquiera puede utilizar.

La algoritmia del cifrado asimétrico es mucho más costosa computacionalmente que la criptografía simétrica, por lo que no se suele utilizar para cifrar grandes cantidades de información.

Generalmente, las claves se asocian con el uso de certificados electrónicos. Los certificados electrónicos consisten en una serie de datos personales que, junto a la clave pública, son firmadas electrónicamente por una autoridad de certificación, dando lugar al ecosistema PKI (Pubic Key Infrastructure).

Una PKI permite formar cadenas de autenticación mediante cadenas de firmas, con las cuales, partiendo de la premisa de que se confía en la autoridad de certificación, es posible trasladar dicha confianza hasta la clave privada utilizada en un mensaje firmado o cifrado. El certificado electrónico asociado a dicha clave privada permite además asegurar que la persona o elemento referenciado es quien dice ser.

12.1.6 Cifrado RSA

Es un algoritmo de cifrado asimétrico desarrollado en la década de los 70 generalmente utilizado para cifrar mensajes pequeños y para realizar firma digital.

Para realizar un cifrado RSA, el emisor utiliza la clave pública del destinatario, que, para descifrarlo, utiliza su clave privada. Nadie más puede leer el mensaje porque la clave privada solo la tiene el destinatario, mientras que cualquiera puede cifrar un mensaje para él. Dicho de otro modo, para realizar un cifrado asimétrico

- Una entidad A posee un par clave privada/clave pública (K / p)
- Si una entidad B desea enviar confidencialmente un mensaje M a la entidad A:
    - B conoce la clave pública de A p
    - B cifra el mensaje M con la clave p:
        - Z = CIFRARp(M)
    - B transmite Z a.
- Sólo A puede acceder al contenido en claro del mansaje, ya que sólo A conoce su propia clave privada K
- A descifra con la clave K el mensaje Z recibido:
    - M = DESCIFRARK(Z).

Para realizar el descifrado asimétrico:

- Una entidad A posee un par clave privada/clave pública (K / p)
- Si A desea enviar un mensaje M, puede generar una firma F, de manera que cualquier otra entidad B pueda verificar que M fue emitido por A
    - A conoce su clave pública K
    - A genera la firma del mensaje M con la clave K:
    - F = FIRMAK(M)
    - A transmite el mensaje y la firma (M, F) a B

- B recibe el mensaje M y verifica la firma de A con la clave p:
- VERIFICARp(M, F)
- Sólo A puede haber generado la firma F del mansaje M, ya que sólo A conoce su propia clave privada K
- Por eficiencia en realidad no se firma el mensaje, sino el digest obtenido de aplicarle una función HASH.

Para realizar una firma electrónica, el emisor de la firma utiliza su clave privada para cifrar los datos a firmar. Compone un mensaje que incluye el dato original, su clave pública y el valor de la firma. El validador utiliza la clave pública para descifrar el valor de firma y compararlo con el hash de los datos originales. Si coinciden, tiene garantías sobre la procedencia del mensaje firmado. De esta manera se comprueba lo siguiente:

- Integridad: Tiene por objetivo detectar alteraciones, ya sean o no intencionadas.
- Autenticidad: Tiene por objetivo verificar la identidad/autoría.
- No repudio: Tiene por objetivo evitar la denegación de la autoría de la información o de una acción.

12.1.7 SSL / TLS

Es en primera instancia, la primera capa de seguridad que se aplica en el protocolo eIDAS, y que protege el canal utilizado para transmitir el token SAML y los metadatos. La protección consiste en el cifrado que aporta el uso del esquema “https” en detrimento de “http”, dentro del navegador del usuario. Los sistemas de navegación web soportan una serie de algoritmos que les permita negociar con servidores. Estos algoritmos se agrupan en protocolos, entre los que se encuentran:

- SSL
  - Desarrollado por Netscape
  - Última versión 3.0 (1996)
- TLS:
  - Basado en SSL 3.0
  - Estándar del IETF (Internet Engineering Task Force)
  - Última versión 1.2 (RFC 5246)

El objetivo de estos protocolos consiste en proporcionar seguridad en las comunicaciones a través de una red. Se ejecuta en una capa entre el protocolo de aplicación (HTTP, SMTP, IMAP, etc.) y el protocolo de transporte. Combina varios algoritmos criptográficos:

- Criptografía asimétrica para establecimiento de claves simétricas.
- Criptografía simétrica para confidencialidad de los datos.
- Algoritmos MAC para la integridad de los datos.

Para establecer un canal de comunicaciones, un cliente y un servidor deciden establecer una conexión de tipo TLS, ya que ambos lo soportan. Comienza el *handshake* para negociar una clave simétrica compartida:

- El cliente manda un mensaje con sus capacidades criptográficas.
- El servidor manda un mensaje con los parámetros criptográficos, que se eligen basándose en las capacidades del cliente, junto con su certificado.
- Si el servidor requiere autenticación del cliente le solicita su certificado y verifica que es confiable para él.
- El cliente verifica que el certificado del servidor es confiable para él.
- Cliente y servidor negocian una clave simétrica (máster) utilizando, por ejemplo, el algoritmo DH (Diffie-Hellman) con firma (ahora sobre un canal autenticado).
- El cliente y el servidor derivan una clave de sesión a partir de la clave master.
- El cliente y el servidor utilizan la clave de sesión para cifrar y autenticar los mensajes durante el resto de la conexión.

12.1.8 Firma y Cifrado en el Token SAML2

Para la firma de un XML, se incluye un elemento en el XML a firmar llamado *Signature.* Este elemento define su propio espacio de nombres y contiene una serie de sub-elementos que componen la firma. Entre ellos, están:

- El elemento *SignatureValue*, donde reside el valor de la firma.
- El elemento *SignedInfo*, donde residen las referencias hacia los datos firmados.
- El elemento el *SignatureMethod* y *DigestMethod*, que contiene un atributo *Algorithm* el cual contiene el método que se ha utilizado para firmar y calcular digest.
- El elemento *CanonicalizationMethod*, que especifica la manera de leer los datos.
- El elemento *KeyInfo*, que especifica la clave utilizada para firmar.

El proceso para firmar un XML es el siguiente:

- Se crea el XML según el estándar SAML2, perfil eIDAS, donde se indican:
  - Petición:
    - Una lista de atributos a recabar referidos al usuario.
    - Configuración de la autenticación (IdP's disponibles, nivel LoA mínimo, etc.).
    - Un identificador único del token SAML.
  - Respuesta:
    - Una lista de atributos obtenidos tras autenticar al usuario.
    - Datos propios de la autenticación. (IdP utilizado, nivel LoA logrado, etc.).
    - Opcionalmente, se cifra la aserción del token SAML, dentro del XML, con la clave pública del otro extremo, obtenida mediante consulta previa, y validación, de sus metadatos.
      - Se emplea el estándar https://www.w3.org/TR/xmlenc-core1/

- Se cifran los datos con una clave simétrica pre calculada y se adjuntan al token. En Cl@ve se utiliza el algoritmo:

  "http://www.w3.org/2009/xmlenc11#aes256-gcm"

- La clave anterior se cifra simétricamente usando la clave pública del destinatario y se adjunta al token. En Cl@ve se utiliza:

  “http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p”

- Se incluye una referencia al destinatario de los datos cifrados.

  - Un atributo XML donde se hace referencia al ID único de la petición a la que se responde.

- Ambos casos:
  - Datos identificativos del emisor (URL de metadatos, ProviderName, etc.).
  - Fecha de emisión, datos identificativos del receptor, un identificador único del token SAML, entre otros.

- Se firma ese XML. Se utiliza la clave privada propia del emisor, de forma que cualquiera pueda comprobar, mediante la clave pública, la autenticidad. Para realizar la firma se utiliza el estándar XMLDSig para generar una firma de tipo enveloped, donde la firma está dentro de lo firmado. Se adjunta el certificado X509 asociado a la clave utilizada, codificado en Base64, para facilitar el proceso de validación y reconstrucción de la confianza.
- Una vez hecho esto, se procede con el envío al otro extremo. Opcionalmente, como paso previo, se puede presentar al usuario la información a transmitir solicitando su consentimiento.
- A continuación, se muestra un ejemplo de XML firmado:

```
<saml2p:Response xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:eidas="http://eidas.europa.eu/attributes/naturalperson" xmlns:saml2="urn:oasis:names:tc:SAML:2
    <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">https://se-eidas.redsara.es/EidasNode/ConnectorResponderMetadata</saml2:Issuer>
    <ds:Signature>
        <ds:SignedInfo>
        <ds:SignatureValue>gFGy6n1mB1CAruEm4Ntkna4ioUozHqEG1yosBPJ+0oB7FKYT48JyQ6A/ECu4d4SrcytnYkP99EDuL7ocJQmFHINnax3FFih5nIrLI4K3en2liY5W4fz9NBXS1RXDPVwiyM7dIP1z
        <ds:KeyInfo>
    </ds:Signature>
    <saml2p:Status>
        <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
        <saml2p:StatusMessage>urn:oasis:names:tc:SAML:2.0:status:Success</saml2p:StatusMessage>
    </saml2p:Status>
    <saml2:EncryptedAssertion>
        <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="_0883e026958e22461bf09e193dbe366d" Type="http://www.w3.org/2001/04/xmlenc#Element">
            <xenc:EncryptionMethod Algorithm="http://www.w3.org/2009/xmlenc11#aes256-gcm"/>
            <ds:KeyInfo>
                <xenc:EncryptedKey Id="_d8f0de86c78a5579e5636fa668e708aa">
                    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
                        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                    </xenc:EncryptionMethod>
                    <ds:KeyInfo>
                    <xenc:CipherData>
                        <xenc:CipherValue>EVc6eD3eU+cQBssXBKWnprQpDUzGpdF/79HrDC9bYnkPLR2mE7J6DDApeFuNQLGJzyOJ6Gfo9iyViU0G+lviUnfObI0w9NL4tAZybwHvs/eXNc7QiilEtwnx+
                    </xenc:CipherData>
                </xenc:EncryptedKey>
            </ds:KeyInfo>
            <xenc:CipherData>
                <xenc:CipherValue>nyzL9u8RCga9whQF+vlyQHQQ9FIPPnxAcvfIqE7S94VhwvdFBKgkKED2KWOIP/13S5qK56V/M2kmXJ3m791cI7wtf5CP0MK0H1SwxmZyhQFhfOYIaPA/8kB5sr+C2FWel
            </xenc:CipherData>
        </xenc:EncryptedData>
    </saml2:EncryptedAssertion>
</saml2p:Response>
```

Ilustración 29: Respuesta SAML - Assertion Cifrado

El proceso para comprobar la firma es el siguiente:

- El receptor abre el token recibido, valida los metadatos del emisor y procede a la validación de la firma electrónica.
- Se carga una instancia del algoritmo declarado en la propia firma.

- Se recoge el nodo XML llamado “X509Certificate”, donde reside la clave pública y el certificado del emisor. Este certificado, o su emisor, debe estar incluido en el almacén de confianza.
- Se recogen las referencias a los datos firmados y se calculan los valores de hash que les corresponden. Se comprueba la integridad comparando el valor indicado en la firma con el valor calculado durante la validación.
- Se utiliza la clave pública para descifrar el nodo XML llamado “SignatureValue”. Se calcula el digest del nodo “SignedInfo”. Se comprueba la autenticidad del mensaje comparando el valor digest descifrado con el valor calculado durante la validación.
- Si todo va bien, y en caso de que la respuesta venga cifrada, procede a descifrar el contenido para obtener la aserción XML en claro, donde se encuentran los datos personales del usuario. Para ello:
    - Se carga una instancia del algoritmo indicado en la aserción cifrada.
    - Se descifra asimétricamente la clave simétrica indicada utilizando la clave privada del receptor.
    - Se descifran los datos cifrados utilizando la clave simétrica obtenida del paso anterior.

- La validación del token incluye la validación de los atributos y del esquema eIDAS. Se comprueba que los datos obligatorios están presentes, etc. En la siguiente imagen se muestran algunos nodos que contienen esa información sobre los atributos, publicadas como metadatos de un módulo:

```
<saml2:Attribute FriendlyName="D-2012-17-EUIdentifier" Name="http://eidas.europa.eu/attributes/legalperson/D-2012-17-EUIdentifier"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>
<saml2:Attribute FriendlyName="EORI" Name="http://eidas.europa.eu/attributes/legalperson/EORI" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>
<saml2:Attribute FriendlyName="LEI" Name="http://eidas.europa.eu/attributes/legalperson/LEI" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>
<saml2:Attribute FriendlyName="LegalAddress" Name="http://eidas.europa.eu/attributes/legalperson/LegalAddress"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>
<saml2:Attribute FriendlyName="LegalName" Name="http://eidas.europa.eu/attributes/legalperson/LegalName" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:uri"/>
<saml2:Attribute FriendlyName="LegalPersonIdentifier" Name="http://eidas.europa.eu/attributes/legalperson/LegalPersonIdentifier"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>
<saml2:Attribute FriendlyName="SEED" Name="http://eidas.europa.eu/attributes/legalperson/SEED" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>
<saml2:Attribute FriendlyName="SIC" Name="http://eidas.europa.eu/attributes/legalperson/SIC" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>
<saml2:Attribute FriendlyName="TaxReference" Name="http://eidas.europa.eu/attributes/legalperson/TaxReference"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>
<saml2:Attribute FriendlyName="VATRegistration" Name="http://eidas.europa.eu/attributes/legalperson/VATRegistration"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>
<saml2:Attribute FriendlyName="RepresentativeD-2012-17-EUIdentifier" Name="http://eidas.europa.eu/attributes/legalperson/representative/D-2012-17-EUIdentifier"
```

Ilustración 30: Atributos SAML del Esquema eIDAS

- Finalmente, el receptor ya dispone de los datos del usuario y continúa con el flujo de negocio que corresponda según sus propias necesidades.

12.1.9 Algoritmos de Firma Soportados por Cl@ve[4]

- http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1
- http://www.w3.org/2007/05/xmldsig-more#sha384-rsa-MGF1

[4] Esta lista indica los algoritmos de firma soportados por defecto (a nivel de código) por Cl@ve. En cualquier caso, es posible restringir, a nivel de configuración, dichos algoritmos de firma soportados, por lo que podría ocurrir que, en un determinado entorno, el sistema Cl@ve tenga deshabilitados alguno/s de lo/s algoritmos aquí indicados.

- http://www.w3.org/2007/05/xmldsig-more#sha512-rsa-MGF1
- http://www.w3.org/2001/04/xmldsig-more#rsa-sha256
- http://www.w3.org/2001/04/xmldsig-more#rsa-sha384
- http://www.w3.org/2001/04/xmldsig-more#rsa-sha512
- http://www.w3.org/2001/04/xmldsig-more#rsa-ripemd160
- http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256
- http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha384
- http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512

12.1.10 Algoritmos de Cifrado de Datos Utilizados por Cl@ve

- http://www.w3.org/2009/xmlenc11#aes128-gcm
- http://www.w3.org/2009/xmlenc11#aes256-gcm
- http://www.w3.org/2009/xmlenc11#aes192-gcm

12.1.11 Algoritmos de Cifrado de Clave Utilizados por Cl@ve

- http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p

12.1.12 Algoritmos de Resumen Soportados por Cl@ve

- http://www.w3.org/2001/04/xmlenc#sha256
- http://www.w3.org/2001/04/xmldsig-more#sha384
- http://www.w3.org/2001/04/xmlenc#sha512

12.2 Filtros Web y Configuración de Seguridad

La gestión de seguridad web de los servlets se implementa en el siguiente paquete Java:

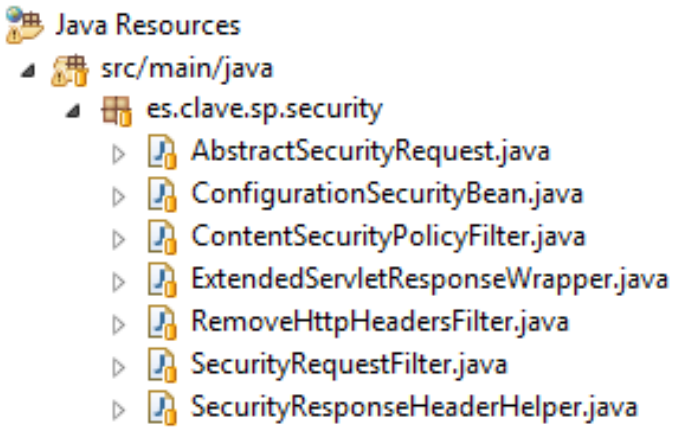


Ilustración 31: Seguridad - Filtros Web

Contiene la lógica que permite filtrar el tráfico frente a solicitudes abusivas o maliciosas. Sus elementos de configuración aparecen en la siguiente página:

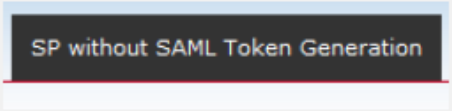
Key	Description
`security.header.CSP.enabled`	Enable/disable sending the Content Security Policy (CSP) header. CSP protects against the injection of foreign content (refer to the *eIDAS-Node Security Considerations* guide for more information about the security features).
`security.header.CSP. includeMozillaDirectives`	In the CSP, this additional directive can be added for backward compatibility with old Mozilla browsers (refer to the *eIDAS-Node Security Considerations* guide for more information about the security features).
`security.header. XXssProtection.block`	This header enables the cross-site-scripting (XSS) filter built into most recent web browsers (refer to the *eIDAS-Node Security Considerations* guide for more information about the security features).
`security.header. XContentTypeOptions.noSniff`	The only defined value 'nosniff' prevents Internet Explorer and Google Chrome from 'MIME-sniffing' by inspecting the content of a response (refer to the *eIDAS-Node Security Considerations* guide for more information about the security features).
`security.header. XFrameOptions.sameOrigin`	Prevents the application from being propagated in a frame or iframe, which in turns protects against key logging, clickjacking and similar attacks. Setting this option to **true** will prevent the eIDAS-Node from being framed in another application. If the SP needs to frame the eIDAS-Node, the option has to be set to 'false' (such as on the second tab of the SP Demo where the SAML request is generated by the eIDAS-Node).  (refer to the *eIDAS-Node Security Considerations* guide for more information about the security features).
`security.header. HSTS.includeSubDomains`	HTTP Strict-Transport-Security (HSTS) instructs browsers to prefer secure connections to the server (HTTP over SSL/TLS) over insecure ones (refer to the *eIDAS-Node Security Considerations* guide for more information about the security features).
`security.header.CSP.fallback CheckMode`	If enabled, CSP fallback check mode includes an enforced CSP violation in JSP pages in order to check browser CSP feature. The included script displays a warning message in client browsers if CSP is not supported. However with CSP enabled browsers it may result in flood of warning messages logged by CSP report servlet. Disabled by default.

Ilustración 32: Seguridad - Elementos de Configuración para Filtros Web

12.2.1 Gestión de Ataques de Denegación de Servicio

En el fichero de configuración interno eidas-commons/src/main/resources/eidasParameters.properties se utiliza para activar o desactivar la validación de los siguientes parámetros de configuración.

```
max.spUrl.size=150
max.attrList.size=20000
max.invalidAttributeList.size=20000
max.attrName.size=100
max.callback.size=300
max.idp.url.size=300
max.attrValue.size=65535
max.attrType.size=25
max.spId.size=40
max.providerName.size=128
max.spName.size=25
max.country.size=150
max.qaaLevel.size=1
max.spQaaLevel.size=1
max.errorCode.size=5
max.errorMessage.size=300
max.username.size=30
max.connectorRedirectUrl.size=300
max.connector.redirectUrl.size=300
max.service.redirectUrl.size=300
max.serviceRedirectUrl.size=300
max.connectorAssertionUrl.size=300
max.SAMLRequest.size=131072
max.SAMLResponse.size=131072
max.RelayState.size=80
max.remoteAddr.size=300
max.remoteHost.size=300
max.localAddr.size=300
max.localName.size=300
```

Esta configuración limita en tamaño los datos que se intercambian y por lo tanto se definen en este documento, ya que pueden ocasionar posibles problemas de denegaciones de servicio ante una incorrecta configuración.

Se pueden destacar especialmente los campos max.SAMLResponse.size si se van a incorporar datos personales de considerable tamaño (este valor es heredado de Stork donde existe el antecedente de haber superado el tamaño descrito al incluir el suplemento al diploma académico, generando una incidencia general).

Este problema puede ser repetible con las URL ajenas cuando se incluyan parámetros en la URL o al hacer HTTP Redirect.

12.2.2 Ataques CSRF

En un ataque de CSRF, un eventual atacante aprovecha una sesión de autenticación previa que ha conseguido interceptar para impersonar a un usuario. Para preservar y transmitir un mensaje de solicitud como paso inicial de un protocolo SAML, se imponen ciertos requisitos para transmitir la respuesta. Concretamente**,** si un mensaje de solicitud de SAML va acompañado de un dato adicional "RelayState", el módulo debe devolver con su respuesta el mismo RelayState que recibió con la solicitud. Este campo ni se procesa ni se valida, solo se responde en modo eco.

Por ello, el formulario de solicitud incluye este campo junto a la solicitud SAML en Base64.

Al recibir la respuesta del IdP, se comprueba la integridad de su valor comparándolo con el valor original. Lo mismo debe hacer el SP.

12.2.3 Ataques en el XML

SQL Injection → Se comprueba que no aparezca el carácter "'" en el XML.

Cross Site Scripting → Se comprueba que no aparezca la palabra "script" en el XML.

Ataque DTD → Existe un repositorio interno, nunca se descargan esquemas externos.

Ataque XML Entity Expansion → No se descargan esquemas.

Billion Laughs Attack → Se define un tamaño máximo para los ENTITIES.

Se comprueba que los esquemas recibidos son los que deben ser y que no hay esquemas adicionales declarados para prevenir envenenamientos.

Se valida el esquema del XML recibido. Esto implica evitar ataques de solapamiento en los que se agrega información al XML con la intención de confundir las referencias de la firma.

Para ello, durante el pre-parseo se lee el XML con los siguientes parámetros:

- factory.setNamespaceAware(true)
- factory.setValidating(true)
- Se declaran los esquemas para no descargarlos:

    - factory.setAttribute("http://java.sun.com/xml/jaxp/properties/schemaLanguage", XMLConstants.W3C_XML_SCHEMA_NS_URI);
    - factory.setAttribute("http://java.sun.com/xml/jaxp/properties/schemaSource", new InputSource(schemaFile));
- factory.setAttribute(XMLConstants.ACCESS_EXTERNAL_DTD, "file,jar"); factory.setAttribute(XMLConstants.ACCESS_EXTERNAL_SCHEMA, "file,jar");

Para prevenir ataques DOS: factory.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, true);

12.3 Códigos de Error

La siguiente tabla muestra los códigos de error que podrían generar los componentes junto con una descripción del error, el comportamiento específico y, cuando sea relevante, las posibles acciones del operador para remediar el error.

Código	Mensaje	Descripción	Comportamiento	Acción del operador
003002	Error de autenticación	Falló la autenticación del ciudadano	urn: oasis: nombres: tc: SAML: 2.0: estado:AuthnFailed	No es necesario tomar ninguna acción.
203014	Invalid signature certificate	Si el parámetro check_certificate se establece en true en eidas.xml, el certificado no se puede autofirmar	El mensaje de error se mostrará en los registros. El error se propagará en el nivel superior	No se permiten certificados autofirmados en el entorno de producción
200009	Estado de retransmisión no válido.	El tamaño del parámetro Estado de retransmisión del SP es mayor de lo permitido.	Respuesta SAML generada con urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue El mensaje de error se mostrará en los registros.	El SP proporciona un estado de retransmisión no válido. No es necesario realizar ninguna acción a nivel de nodo eIDAS
200006	El token de solicitud SAML falta o no es válido.	Falta la solicitud SAML del SP o no es válida.	No se genera ninguna respuesta SAML. El mensaje de error se mostrará en los registros.	La firma de solicitud SAML no es válida (podría haberse falsificado).
200007	El servidor no pudo identificar al proveedor de servicios	El SP no fue identificado con éxito utilizando la información enviada por él.	No se genera ninguna respuesta SAML El mensaje de error se mostrará en los registros.	El nombre del SP no es reconocido por el nodo eIDAS Configuración del conector
203001	Error al crear el mensaje SAML	No se puede crear un mensaje SAML válido	Este es el mensaje de error predeterminado para el mensaje de generación SAML.	Consulte el error anterior en el registro para encontrar la causa
202005	Lista de atributos no válidos	La lista de atributos de la solicitud SAML está vacía o el tamaño es mayor de lo permitido.	Respuesta SAML generada con error El mensaje de error se mostrará en los registros.	El servicio proxy de nodo eIDAS recibe una solicitud SAML con una lista de atributos no válidos. No se requiere ninguna acción.
202002	Token de solicitud SAML no válido	La solicitud SAML recibida por el servicio de proxy de nodo eIDAS no es válida.	No se generará ninguna respuesta SAML.	El token enviado por el conector no es válido al comprobar por su integridad en el Servicio de Proxy.
202003	Token de respuesta SAML no válido	La respuesta SAML enviada por el servicio de proxy de nodo eIDAS no es válida.	Respuesta SAML generada con error El mensaje de error se mostrará en los registros.	El token enviado por el servicio de proxy no es válido al comprobar su integridad en el conector
202012	no.consent.val.mand.attr			
202010	Falta el atributo obligatorio	Falta el atributo requerido	La respuesta SAML generada con el mensaje errorError se mostrará en los registros. (por ejemplo: faltan atributos: LegalPersonIdentifier, LegalName)	Falta un atributo obligatorio en la respuesta. No se requiere ninguna acción.
200003	Falta el dominio sp o no se encuentra en la lista de SP	El dominio sp falta o no se encuentra en el servicio de confianza	No se genera ninguna respuesta SAML El mensaje de error se mostrará en los registros.	La configuración del conector de nodo eIDAS no reconoce el dominio SP.

203001	Error al crear el mensaje SAML	No se puede crear un mensaje SAML válido	Este es el mensaje de error predeterminado para el mensaje de generación SAML.	Consulte el error anterior en el registro para encontrar la causa
203001	Error al crear el mensaje SAML	No se puede crear un mensaje SAML válido	Este es el mensaje de error predeterminado para el mensaje de generación SAML.	Consulte el error anterior en el registro para encontrar la causa
203006	Nivel de QAA no válido	El nivel de QAA máximo de eIDAS-Node Proxy Service/(VIdP) es inferior al solicitado.	El mensaje de error se mostrará en los registros. El error se propagará en el nivel superior	Comprobación de validación de negocio típica, compruebe si la configuración es correcta
000004	El número de solicitud es mayor o igual que el número máximo permitido.	El número de solicitudes realizadas por el ciudadano es superior al permitido.	No se generará ninguna respuesta SAML. El mensaje de error se mostrará en los registros.	
203012	Sesión no válida	La sesión no es válida	El mensaje de error se mostrará en los registros. El error se propagará en el nivel superior (202004)	La sesión (entre la solicitud SAML y la respuesta SAML) no se puede conciliar. Consulte el contenido del registro para obtener más detalles.
203014	Certificado de firma no válido	Si el parámetro check_certificate se establece en true en eidas.xml, El certificado no se puede autofirma	El mensaje de error se mostrará en los registros. El error se propagará en el nivel superior	No se permiten certificados autofirmados en el entorno de producción
203015	Algoritmo de firma no válido	El algoritmo de firma debe validarse con una lista blanca en eidas.xml	El mensaje de error se mostrará en los registros. El error se propagará en el nivel superior	La lista de algoritmos debe ajustarse a la regulación
003003	Enlace de protocolo no válido	El enlace de protocolo actual no es válido o no es compatible con el método http actual	No se genera ninguna respuesta SAML, se 201002 el código de error o se muestra 200006 al usuario	Business exception, no action needs to be taken.
003004	Aserción firmada no válida	La aserción de atributo en la respuesta está firmada y su firma no se valida	No se genera ninguna respuesta SAML	La integridad del mensaje SAML se ve comprometida, no es necesario realizar ninguna acción.
203020	Algoritmo de cifrado no válido	El algoritmo de cifrado no está en la lista de algoritmos aceptados	No se genera ninguna respuesta SAML	Comprobación de validación empresarial típica, compruebe si la configuración es correcta.
203005	Problema de configuración	La configuración del motor SAML no es válida	El mensaje de error se mostrará en los registros. El error se propagará en el nivel superior (202004)	Comprobar la configuración
203007	Validación de mensajes no válidos	La configuración del motor SAML no es válida	El mensaje de error se mostrará en los registros. El error se propagará en el nivel superior	Comprobación de validación empresarial típica, consulte el contenido del registro para obtener más información.
203016	No se puede acceder al almacén de claves	El archivo de almacén de claves configurado para SAMLEngine no se puede leer.	No se genera ninguna respuesta SAML	Compruebe la vía de acceso del almacén de claves.

203017	Certificado no encontrado	El certificado no se encuentra en el archivo de almacén de claves configurado para SAMLEngine.	No se genera ninguna respuesta SAML	Comprobación de validación empresarial típica, compruebe si la configuración es correcta. También se utiliza al extraer un certificado de metadatos en la validación no da como resultado una certificación válida o adecuada para el país al que se hace referencia.
203018	Certificado que no es de confianza	El certificado no es de confianza	No se genera ninguna respuesta SAML	Comprobación de validación empresarial típica, compruebe si la configuración es correcta.
003006	Invalid metadata	El contenido de los metadatos no es válido (ha caducado)	No se genera ninguna respuesta SAML	Excepción de negocio, no es necesario realizar ninguna acción
203022	Metadatos no válidos	El cifrado de respuesta es obligatorio, pero se ha producido un error durante el cifrado (por ejemplo, credencial de cifrado desconocida)	No se genera ninguna respuesta SAML	Compruebe si la configuración es correcta (por ejemplo, si los metadatos del destinatario de la respuesta contienen una credencial de cifrado válida)
203023	Error al descifrar la respuesta	El cifrado de respuesta es obligatorio, pero el mensaje de respuesta recibida no está cifrado	No se genera ninguna respuesta SAML	Comprobar la configuración con el remitente del mensaje de respuesta
203021	Conjunto de atributos no válido	El conjunto de atributos no incluye atributos obligatorios. Para una persona física, el conjunto obligatorio es FamilyName, FirstName y PersonIdentifier.	No se genera ninguna respuesta SAML	Comprobación de validación empresarial típica, compruebe si la configuración es correcta.
003008	Origen no válido para los metadatos	El origen de metadatos no es válido (es decir, solo debe recuperarse de una URL HTTPS)	No se genera ninguna respuesta SAML	Excepción de negocio, no es necesario realizar ninguna acción
003009	No se puede cargar la información de metadatos	El origen de metadatos es incorrecto o no está disponible, o el formato de los metadatos es incorrecto.	No se genera ninguna respuesta SAML	Excepción de negocio, es necesario tomar medidas
202015	Valor no válido para el nivel de garantía	El servicio de proxy de nodo eIDAS no puede validar el nivel de garantía solicitado	Respuesta SAML generada con error El mensaje de error se mostrará en los registros.	La respuesta SAML generada con el mensaje errorError se mostrará en los registros.
202016	SPType no es coherente	El servicio de proxy de nodo eIDAS recibió una solicitud con SPType establecido, mientras que los metadatos del solicitante también tienen sptype establecido	El mensaje de error se mostrará en los registros.	SPType debe establecerse en la solicitud o en los metadatos

202017	SPType obligatorio no encontrado	El SPType no se establece ni en los metadatos ni en la solicitud	El mensaje de error se mostrará en los registros.	SPType debe establecerse en la solicitud o en los metadatos
003007	Nivel de garantía no válido	La solicitud o la respuesta contiene un valor no válido para nivel de garantía. Los valores válidos son http://eidas.europa.eu/LoA/low, http://eidas.europa.eu/LoA/substantial, http://eidas.europa.eu/LoA/high	No se genera ninguna respuesta SAML	Excepción de negocio, no es necesario realizar ninguna acción
203022	Error al cifrar la respuesta	El cifrado de respuesta es obligatorio, pero se ha producido un error durante el cifrado (por ejemplo, credencial de cifrado desconocida)	No se genera ninguna respuesta SAML	Compruebe si la configuración es correcta (por ejemplo, si los metadatos del destinatario de la respuesta contienen una credencial de cifrado válida)
000005	El país seleccionado no es válido o no es de confianza	El país del ciudadano seleccionado no es válido o no se encontró	El SP recibe una respuesta SAML withurn:oasis:names:tc:SAML: 2.0:status:RequestDeniedError mensaje se mostrará en los registros.	En el SP, cambie el país del SP.