Documentación técnica de la cédula de identidad con chip

Autor
Área de Seguridad de la información de Agesic

Fecha de creación
11/05/2023

Tipo de publicación
Guía técnica

Resumen

Documentación técnica de la cédula de identidad con chip

Introducción

- *Gestión de riesgos*. Supervisá, analizá y respaldá los procesos de documentación, validación, evaluación y autorización necesarios para garantizar que los sistemas de TI existentes y nuevos cumplan con los requisitos de seguridad y ciberseguridad de la organización.
- *Desarrollo de software*. Desarrollá y escribí / codificá aplicaciones informáticas, software o programas de utilidad especializados nuevos o ya existentes siguiendo las mejores prácticas de garantía de software.
- *Arquitectura de sistemas*. Desarrollá conceptos de sistema y trabajá en las fases de capacidades del ciclo de vida de desarrollo de sistemas; traduce la tecnología y las condiciones ambientales (por ejemplo, leyes y regulaciones) en diseños y procesos de sistemas y seguridad.
- *Desarrollo de sistemas*. Trabajá en las fases del ciclo de vida de desarrollo de sistemas.
- *Planificación de requisitos de sistemas*. Consultá con los clientes para reunir y evaluar los requisitos funcionales, traducí estos requisitos en soluciones técnicas y brinda orientación a los clientes sobre la aplicabilidad de los sistemas de información para satisfacer las necesidades comerciales.
- *I + D tecnológica*. Realizá evaluaciones de tecnología y procesos de integración; proporcioná y respaldá una capacidad de prototipo y / o evaluá su utilidad.
- *Prueba y evaluación*. Desarrollá y realizá pruebas de sistemas para evaluar el cumplimiento de las especificaciones y requisitos mediante la aplicación de principios y métodos para una planificación, evaluación, verificación y validación rentables de características técnicas, funcionales y de rendimiento (incluida la interoperabilidad) de sistemas o elementos de sistemas que incorporan TI.

Servicio de firma con cédula de identidad digital

Con la emisión del documento de identidad digital surge la oportunidad de usarlo para dar soporte a los servicios digitales actuales, así como también generar nuevos servicios.

El documento contiene 2 chips, el primero es uno no visible que contiene los mismos datos impresos en el documento, con excepción de la huella dactilar. El segundo es un chip de contacto que almacena una clave privada y un certificado electrónico para cada persona, esto permite el uso de la firma electrónica dentro del marco de la PKI Uruguay.

En la actualidad gran parte de las personas cuenta con un dcumento , y considerando que el documento nacional de identidad es obligatorio para los ciudadanos mayores de 18 años, se estima que en los próximos 2 años la mayoría de los ciudadanos uruguayos contarán con uno. Esto abre una puerta para incorporar el servicio de firma digital a los servicios que se prestan actualmente para potenciarlos, o incluso generar nuevos servicios que ahora se hacen viables gracias a esta nueva realidad tecnológica.

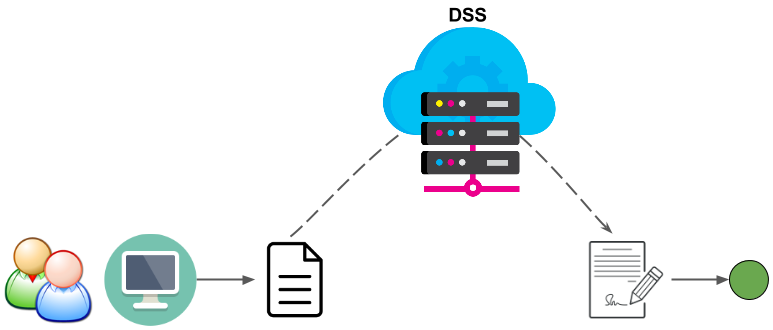
Como parte de las actividades para fomentar el uso de estas tecnologías, Agesic cuenta con una plataforma de Firma Digital como servicio para integrar a aplicaciones de terceros.

Descripción general

El DSS es un servicio ofrecido por Agesic que permite firmar digitalmente documentos utilizando un DNIe.

Esto proporciona nuevas formas de relacionamiento con los usuarios finales, ya que cualquier operación que se realiza de forma presencial por requerir el consentimiento mediante firma, puede comenzar a realizarse digitalmente y en forma remota.

Éste servicio fue pensado para ser fácil de usar e implementar. Cuando la persona realiza un trámite en línea mediante una aplicación web siempre sigue un flujo previamente definido en la misma. En el momento en que se requiera la firma de un documento para validar el trámite, la aplicación web se encarga de enviarlo hacia el DSS. Aquí entra en juego la firma digital por parte del servidor. El procedimiento no tiene ninguna complejidad adicional para el usuario, solamente se le solicita conectar su DNIe a un lector de tarjetas inteligentes e insertar su PIN de seguridad.



Beneficios

El principal beneficio de la integración es la comodidad y flexibilidad que se le brinda al ciudadano para realizar trámites. De esta forma se habilita a la organización a relacionarse digitalmente con las personas, en un entorno seguro y de alta disponibilidad. Esto representa un modelo novedoso en el contexto de servicios digitales y plantea la posibilidad de desarrollar nuevos escenarios de trabajo.

Siguiendo con la perspectiva de los usuarios, el servicio resulta muy amigable en términos de usabilidad, ya que el proceso de firma se da dentro del navegador web, requiriendo únicamente la instalación de un plugin para su uso.

Finalmente, pasando a un ámbito más técnico, el DSS está basado en un estándar internacional, por lo cual integrarse a él no supone un esfuerzo significativo por parte de las instituciones interesadas.

Requerimientos

Los requerimientos se pueden dividir en dos partes, por un lado los del proveedor del servicio (instituciones interesadas) y por otro lado los del cliente.

Desde la perspectiva del proveedor de servicio se necesita realizar la integración con la plataforma, que como ya se mencionó, es un procedimiento relativamente sencillo al basarse en el internacionalmente conocido protocolo DSS (Digital Signature Service).

Desde el punto de vista del usuario, además del DNIe y un lector, es necesaria la instalación de un plugin en su navegador web. Su instalación resulta muy intuitiva dado que el sistema provee ayuda en pantalla en este proceso.

Contenidos relacionados:

- Integración con Servicio de Firma Digital

Autenticación y Firma Avanzada a través de PKCS#11

PKCS#11 es una interfaz estándar para el acceso a dispositivos criptográficos y tanto estos como la Cédula de Identidad uruguaya cuentan con drivers para funcionar como tal. La gran ventaja que ofrece este método es que, al tratarse de una interfaz estándar, existen muchas bibliotecas para facilitar la integración en múltiples plataformas. Para este fin, es necesario, en primera instancia, contar con los drivers correspondientes instalados según el SO en el que se va a trabajar. Dichos drivers pueden ser encontrados en el sitio de Agesic destinado a tal fin.

En general, se recomienda utilizar este mecanismo para autenticar o firmar siempre que se trate de una aplicación *standalone* o con un cliente web y se esté en un sistema operativo para el cual haya drivers soportados (esencialmente, Windows, Ubuntu y OS X). A nivel del software cliente, es común encontrar bibliotecas especiales (*PKCS#11 Providers*), que se instancian, se vinculan a la biblioteca del SO que implementa la interfaz PKCS#11; y la firma o la autenticación con estos dispositivos, que son tecnológicamente equivalentes, se realiza a través de los propios métodos del *provider*, que oficia como *Wrapper*.

Existe un caso de autenticación y dos casos de firma que vale la pena comentar en mayor profundidad, que son el de ***Autenticación con TLS***, ***Firma de PDF en Adobe*** y el del ***Uso de componente externo para firma***.

- ***Autenticación con TLS***. Un uso muy común de autenticación con certificados es aquel en que se emplea TLS con la exigencia de certificado cliente. En este modo de uso de TLS, además de contar el servidor con un certificado para autenticarse ante el cliente (por ejemplo, para evitar phishing), la negociación tiene un paso adicional, en el que el servidor exige al cliente que presente un certificado para autenticarlo. En términos prácticos, una vez que el browser detecta que el servidor web le solicita que presente certificado del cliente, este busca entre los certificados cargados en su *Store* uno que sirva. Cualquier certificado de Firma Avanzada en su dispositivo criptográfico, incluyendo la Cédula de Identidad Digital, sirve para realizar autenticación de este modo. Se requiere cargar previamente el certificado en el *Store* del navegador e ingresar el pin para que cargue la identidad del usuario correctamente. Para esta carga, se necesita especificar el driver PKCS#11 del dispositivo, que debe estar previamente instalado, debido a que los navegadores utilizan esta interfaz para acceder a los dispositivos criptográficos que portan los pares de llaves y certificados.

- ***Firma de PDF en Adobe***. Adobe Reader implementa la Firma Digital de PDF de forma muy sencilla y soporta la carga de un dispositivo a través de interfaz PKCS#11. Es así que es posible cargar la Cédula de Identidad Digital u otro dispositivo criptográfico en Adobe y realizar la Firma Digital Avanzada de un documento PDF en forma simple. Es una alternativa muy sencilla para intercambiar documentación firmada entre partes, ya que no requiere implementación de software alguna. Firmar PDF con Firma Avanzada en Adobe

- ***Uso de componente externo para firma***. Es común que se utilicen componentes que factorizan la implementación de la firma, de forma de simplificar su implementación. Si bien puede haber de muchos tipos, la mayoría basa su implementación en PKCS#11 para soportar una amplia gama de dispositivos criptográficos. Agesic desarrolló un componente de firma como software público. Esto implica que puede descargarse y evolucionarse libremente, ya que su código es abierto, pero también que **no tiene soporte**. Esto último implica que si bien su uso es completamente libre, quien lo incorpore a sus sistemas deberá ser capaz de darle soporte sobre como parte de su solución. **Agesic no asume responsabilidad alguna por su soporte, mantenimiento y evolución**. Se encuentra publicado en el Catálogo de Software Público Uruguayo. Plataformas como Genexus cuentan con componentes ya implementados para facilitar la integración aplicaciones desarrolladas en su tecnología.

Sin perjuicio de estas soluciones, es posible implementar el acceso a las claves y certificados por PKCS#11 desde cualquier tipo de sistema. Como se tiene que acceder a hardware, las implementaciones en arquitecturas *standalone* o cliente-servidor son directas; en arquitecturas web es necesario desarrollar algún componente que corra del lado cliente y realice la firma en nombre del sistema, dado que los browsers no cuentan con interfaces estándar para la realización de firmas digitales desde, por ejemplo, Javascript. La motivación del componente de firma desarrollado por Agesic como Software Público es, precisamente, facilitar la integración de la firma para aplicaciones web.

Descripción técnica de Cédula de Identidad Digital

Desde mayo de 2015, la DNIC emite la nueva Cédula de Identidad, que además de contar con mayores medidas de seguridad física, incorpora muchos elementos digitales para soportar la identificación de las personas en el mundo digital.

Descripción general

La nueva cédula está construida sobre policarbonato y es impresa con grabado láser, garantizando así la durabilidad deseada de 10 años, así como también imposibilitando el borrado o sustitución de datos en ella, dado que, en esencia, el policarbonato es quemado por el láser. Por otra parte, cuenta con medidas de seguridad de primer nivel, que son las visibles al ojo humano (por ejemplo, tramas guilloche y tinta OVI); medidas de segundo nivel, que son las visibles con instrumental específico como luz negra o microscopio (por ejemplo, microtexto y hologramas UV); y medidas de tercer nivel, que son las que requieren instrumental específico y que son solo conocidas por la DNIC.

A nivel digital, la cédula de los mayores de edad posee dos chips: uno visible y de contacto y uno no visible y de uso sin contacto. El chip sin contacto contiene el documento de viaje electrónico, conformante con la normativa ICAO para documentos de viaje electrónicos. Si bien no sustituye el pasaporte por las propias regulaciones de ICAO, sí provee una funcionalidad análoga al chip sin contacto de los nuevos pasaportes electrónicos, y permite realizar control migratorio automático en las estaciones que así lo permitan en los países a los que se permite viajar con la cédula (Sudamérica), como lo es en los eGates del Aeropuerto de Carrasco. El chip con contacto contiene aplicaciones destinadas a realizar identificación electrónica de las personas, pensadas para su uso en contextos de servicios electrónicos, tanto públicos como privados. En este sentido, cuenta con una funcionalidad para leer digitalmente todos los datos del titular (identificación), una para confrontar electrónicamente la huella de una persona contra la almacenada para ver si efectivamente es la dueña de la cédula (Match-On-Card) y un par de llaves y un Certificado Digital para realizar autenticación y firma electrónica avanzada de personas.

Frente de la cédula

Se puede observar debajo de la cara el logo que indica que el documento está habilitado como documento de viaje digital.

Dorso de cédula

En el dorso de la cédula se puede ver el chip de contacto.

Frente de cédula sin chip (menores)

Al no contar con chip sin contacto, la cédula de los menores no constituye un documento de viaje digital, por lo que no cuenta con el logo indicativo. En su lugar se puede observar la silueta de un ñandú como parte del arte del diseño.

Aplicaciones y usos del chip sin contacto

El chip sin contacto de la Cédula de Identidad Digital contiene la aplicación de documento de viaje digital. Dicha aplicación es completamente compatible con la especificación establecida en el documento ICAO 9303 para documentos de viaje digitales y con información biométrica, por lo que cualquier sistema que implemente la lectura de este tipo de documentos, incluyendo los eGates, podrá leer y validar la información de la persona de su Cédula de Identidad Digital de forma inalámbrica. Su uso está pensado para controles migratorios automáticos, en los que el sistema de control puede leer digitalmente los datos del viajero y mediante reconocimiento facial verificar que, efectivamente, es la persona que está queriendo pasar, autenticándolo completamente. Debido a que también lee información identificatoria de la persona, puede realizar cualquier control adicional que considere necesario, como verificación contra listas de habilitación para viajar por cualquier causa legal que aplique. Si bien el documento de viaje por excelencia es el pasaporte, que también es digital y posee la misma tecnología, la cédula puede ser usada como documento de viaje en Sudamérica y, por lo tanto, puede ser usada digitalmente también. Sin perjuicio de todo esto, se puede utilizar esta aplicación de documento de viaje para el fin que se desee, leyendo la información oportunamente con un lector sin contacto en cualquier sistema.

Para prevenir la lectura no autorizada de los datos, por ejemplo, por parte de un lector oculto en la calle, el chip cuenta con protección de tipo *BAC (Basic Access Control)*. Dicho mecanismo consiste en que, para leer los datos, se debe presentar un secreto compartido. Según se indica en la propia especificación de ICAO, dicho secreto puede ser derivado de forma sencilla a partir de información que se encuentra impresa en el documento e incluso forma parte de la MRZ (*Machine Readable Zone*) y puede ser leída ópticamente por una máquina. Esto provoca que cuando se quiere voluntariamente leer digitalmente el documento, por ejemplo, en un control migratorio, el procedimiento es muy sencillo, pero hace imposible la lectura de los datos de forma oculta evitando así abusos sobre la privacidad de la persona. Adicionalmente, el chip cuenta con protección mediante *Active Authentication*, que básicamente consiste en que los datos estén firmados digitalmente por la autoridad emisora, en este caso DNIC, pero que además el chip cuente con una clave privada mediante la cual es posible autenticarlo y determinar así que no fue clonado.

Todas las funcionalidades y mecanismos de seguridad implementados en el chip sin contacto son completamente conformantes con ICAO 9303, por lo que su uso es estándar y pueden encontrarse recursos en abundancia para hacerlo. No obstante, en el siguiente link se encontrará documentación para hacerlo.

Aplicaciones y usos del chip con contacto

Se trata del chip visible de la cédula, el cual está conforme con la normativa ISO/IEC 7816 en todas sus secciones. Para acceder a él, es necesario un lector de tarjetas inteligentes estándar, que se puede conseguir en plaza, en terminales POS estándar (es la misma interfaz física que las tarjetas EMV) e incluso viene embebido en algunas computadoras personales.

Este cuenta con varias funcionalidades, que pueden ser agrupadas conceptualmente en tres aplicaciones:

- ***Identificación***: Obtención de datos visibles en el plástico (menos la imagen de la huella e imagen de la firma) en formato digital, permitiendo así su lectura automática por parte de un sistema informático, evitando digitaciones y simplificando así procesos de registro.
- ***Match on Card***: Confrontación biométrica de una huella capturada contra las huellas de la persona, almacenadas en el documento. Permite implementar la verificación fehaciente de que una persona es efectivamente la dueña de un documento de identidad determinaod, evitando así la validación humana tradicional de comparar la persona contra la foto impresa y reduciendo así notablemente el riesgo de suplantación de identidad.
- ***Autenticación y Firma Digital Avanzada***: La cédula contiene un par de llaves y un certificado de Firma Digital Avanzada de Persona Física. Esta aplicación permite utilizarlo para realizar la Firma Electrónica Avanzada de un documento o realizar una autenticación fuerte utilizando ese par de llaves y el certificado. Su activación requiere el ingreso de un PIN, que es elegido por el usuario en el momento en que le entregan por primera vez el documento.

De acuerdo a la especificación ISO 7816, sección 4, la Cédula de Identidad Digital es utilizada desde una sistema al cual está conectada mediante el intercambio de comandos y respuestas APDU (*Application Protocol Data Unit*), por lo que cualquier uso de ella puede ser hecho intercambiando los comandos apropiados. La ventaja que ofrecen estos comandos es su flexibilidad, dado que se puede acceder directamente a cada función de la Cédula de Identidad a mucho más bajo nivel que las tres aplicaciones anteriormente mencionadas. La desventaja radica en que el nivel de complejidad es alto, dado que los comandos y respuestas son en esencia secuencias de Bytes con una estructura mínima, además de que necesario acceder al hardware directamente; por eso, se debe estar ejecutando en la máquina a la que está directamente conectada la CIe (máquina cliente en una arquitectura web o cliente-servidor). Cualquier otro tipo de driver, SDK, biblioteca, plugin o servicio que se mencione de aquí en lo sucesivo es implementado necesariamente en base a este conjunto de primitivas.

Identificación y match-on-card

Para las aplicaciones de Identificación y Match-on-Card, no se cuenta con ninguna SDK, biblioteca o servicio de más alto nivel al momento de la escritura del presente artículo (setiembre de 2016), por lo que su uso tiene que realizarse exclusivamente a través de APDU. Afortunadamente, la guía de uso y los ejemplos vinculados a continuación abarcan el uso de ambas funcionalidades.

- Uso de Cédula de Identidad Digital a través de APDU - Documentación de comandos APDU para el uso de la Cédula de Identidad Digital.
- eIDuy en github - Iniciativa abierta con ejemplos de uso de la CIe a través de APDU.

Autenticación y Firma Digital

La autenticación y la firma con la Cédula de Identidad Digital son procesos muy similares, debido a que se utilizan los mismos mecanismos tecnológicos subyacentes: criptografía asimétrica e infraestructura de claves públicas (PKI). Las funcionalidades de autenticación con Certificado Digital y Firma Electrónica Avanzada de documentos pueden ser realizadas a través de APDU (los instructivos anteriores cubren también estos casos), pero para estos casos se cuenta con varias alternativas más para la implementación en varios contextos, que hacen que sea notoriamente más sencilla.

La Firma Electrónica Avanzada es un mecanismo de firma de documentos digitales reconocido por la Ley Nº 18.600, del 21 de setiembre de 2009, como equivalente a la firma manuscrita tradicional, por lo que otorga las más altas garantías jurídicas para las transacciones electrónicas. Por más información general, ver Firma Electrónica Avanzada.

Como ya fue mencionado, la Cédula de Identidad Digital cuenta con un par de llaves y un certificado digital emitido por el Ministerio del Interior, que por ser una Autoridad Certificadora acreditada por la UCE constituye un certificado capaz de usarse para realizar Firmas Electrónicas Avanzadas. El proceso tecnológico de la firma consiste en realizar un hash del documento y firmarlo con la clave privada que se almacena en la Cédula de Identidad Digital, dejando el resultado de ese proceso asociado al documento como información utilizada para verificar su integridad y la identidad del firmante. En un nivel un poco más bajo, este proceso de firma buscará siempre hacer lo siguiente:

- Obtener el certificado de persona física de la tarjeta.
- Presentar el PIN del usuario a la tarjeta para activarla.
- Firmar digitalmente un hash del documento.
- Incluir el resultado de dicha firma, el certificado de la persona y, posiblemente, información adicional como parte del documento, de forma de posibilitar la posterior validación por quien lo reciba.

Si el receptor del documento logra validar su firma con el certificado, entonces puede tener la confianza de que el documento no fue modificado desde su firma y que el firmante es la persona cuyos datos figuran en el certificado.

La "autenticación" es el proceso mediante el cual un sistema se asegura de que el usuario con el que se está comunicando es, efectivamente, quien dice ser. No debe ser confundida con la "autorización", que es el proceso mediante el cual se determina qué cosas un usuario *ya autenticado* puede hacer en un determinado sistema (permisos). En este contexto, la autenticación con la Cédula de Identidad Digital refiere a los mecanismos que pueden ser implementados para que un usuario demuestre su identidad ante un sistema simplemente mediante la presentación de su cédula y su PIN. Esta se implementa esencialmente realizando una firma digital de un mensaje elegido aleatoriamente por el sistema que quiere realizar la autenticación; al validar que el usuario fue capaz de firmarlo, puede tener la confianza de que quien está del otro lado de la comunicación es la persona cuyos datos figuran en el certificado, logrando efectivamente autenticarla. La diferencia que tiene con la Firma Digital es que en la firma el contenido del documento firmado es relevante desde un punto de vista de negocio, mientras que en la autenticación lo que se firma es un *challenge* relevante solo en esa transacción y que es descartado.

En cualquiera de los casos descritos a continuación es necesario un lector de tarjetas inteligentes estándar; su eventual instalación queda fuera de alcance, debido a que esta es la manera en que el sistema se comunica con la tarjeta y a que la inmensa mayoría de lectores son *Plug & Play* en los sistemas operativos más usados.

APDU

Al igual que cualquiera de los mecanismos que implementa la cédula, la autenticación puede ser realizada accediendo directamente a las funciones de bajo nivel de la Cédula de Identidad Digital, enviando los comandos APDU correctos. Esta implementación es muy flexible, pero de complejidad alta, por lo que se recomienda su uso solo cuando ninguna de las otras alternativas es aplicable. En entornos de propósito específico, por ejemplo en dispositivos embebidos, cajeros automáticos o terminales de punto de venta (POS), este es el método preferido, ya que no se está en una red pública o entorno web para utilizar otros servicios, ni tampoco se cuenta con drivers o bibliotecas de más alto nivel. También se recomienda este uso en PC tradicionales cuando no existen drivers PKCS#11 para el sistema operativo en uso.

Ver Uso de Cédula Digital a través de APDU.

PKCS#11

PKCS#11 es una interfaz estándar para el acceso a dispositivos criptográficos y, por lo tanto, es una alternativa natural para implementar Firma Electrónica Avanzada con cualquier token o con la Cédula de Identidad Digital, ya que esta cuenta con drivers para funcionar como dispositivo PKCS#11. Por mayor información para su implementación, se recomienda ver Autenticación y Firma Avanzada a través de PKCS#11, considerando la Cédula de Identidad Digital como un dispositivo criptográfico cualquiera.

Servicio de autenticación con Cédula de Identidad Digital

Agesic cuenta con una plataforma para autenticación con Cédula de Identidad Digital como servicio, que ofrece facilidad de integración con aplicaciones por utilizar protocolos estándar y maduros para la integración y, al basarse en plugins y componentes de browser y ser específicamente diseñada para usar la Cédula de Identidad uruguaya, provee una experiencia de usuario de gran calidad.

Su funcionamiento esencial es sencillo: cuando una aplicación web quiere realizar la autenticación de una persona, en lugar de realizarla localmente la redirige a esta plataforma, delegando la autenticación efectiva del sujeto en ella. El protocolo de integración provee las garantías de integridad necesarias para que la aplicación en cuestión pueda estar segura que la persona fue efectivamente autenticada y hacerse de sus datos identificatorios básicos.

Las restricciones para su uso son que aplica solamente a aplicaciones web, y que los plugins son solo para versiones desktop de los navegadores, así que si bien se tiene un alto nivel de compatibilidad en PC y navegadores tradicionales, de momento no soporta los entornos Mobile. Las ventajas que ofrece usar esta plataforma son la simplicidad en la implementación y la experiencia del usuario final, por lo que es la opción recomendada siempre que aplique tecnológicamente.

Su documentación puede encontrarse en la página dedicada al Servicio de Autenticación con la Cédula de Identidad Digital.

Servicio de firma con Cédula de Identidad Digital

Análogamente a la autenticación, Agesic cuenta con una plataforma que ofrece firma con Cédula de identidad digital como servicio. Esta también comparte las propiedades del servicio de autenticación (facilidad de implementación y experiencia de usuario), y su funcionamiento es análogo: se delega la firma en este servicio. La aplicación recibe el documento ya firmado y puede efectuar su validación para asegurarse de que todo está correcto.

Las restricciones y ventajas que ofrece son las mismas que para la autenticación, por lo que se recomienda su uso salvo cuando el entorno de los usuarios no sea compatible o no se quiera incurrir en una dependencia de un servicio externo. En esos casos, se recomienda implementar una solución propia, para lo cual se puede elegir alguna de las soluciones alternativas ya mencionadas.

Su documentación puede encontrarse en la página dedicada al Servicio de firma con la Cédula de Identidad Digital.

Componente de firma para Firma Digital Avanzada

Este documento describe la arquitectura del Componente de Firma y cómo se integra a un sistema web para la firma digital de documentos PDF, XML y génericos mediante Hash.

Alguna de las ventajas de utilizar la Firma Digital Avanzada son:

- Autenticación: La Firma Digital es equivalente a la firma física del documento, autenticando la identidad de quien lo firma.
- Integridad: Asegura que el contenido original del documento no ha sido alterado.

Arquitectura

Esquema general de la solución

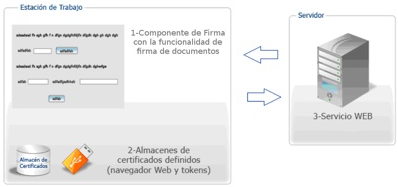
La solución está desarrollada con tecnología Java Applet y/o Java Web Start y es compatible con jre 1.6 y 1.7.

La interacción con el sistema web existente se produce a través de dos servicios Web SOAP que publicará el sistema existente y el Componente de Firma podrá consumir:

- En el primer servicio el Componente de Firma podrá obtener la colección de documentos a firmar.
- En el segundo servicio el Componente de Firma podrá entregar al sistema existente los documentos ya firmados por el usuario.

Ambos servicios se autenticarán con un identificador de transacción alfanumérico, que será generado aleatoriamente por la aplicación a la cual se le está incorporando Firma Digital y será obtenido por el Componente de Firma como un parámetro de carga. El mismo ID de transacción se utiliza para autenticar el consumo de los dos servicios; y cuando se consume el segundo, este deberá ser invalidado en forma interna por el sistema existente.

La siguiente figura muestra la arquitectura general de la solución.



1. Componente de Firma, conteniendo las funcionalidades necesarias para visualizar y firmar un conjunto de documentos, el mismo se ejecuta en el contexto del navegador Web que el usuario está utilizando o como una aplicación Java Web Start.

2. Token criptográfico con el certificado de firma y su clave privada

3. Servicio Web de Integración (AgesicFirmaWS), encargados de enviar los documentos a firmar, recibir los documentos firmados y generar el ID de transacción.

Como se muestra en la figura, los componentes 1 y 2 residen en la estación de trabajo del usuario que está firmando los documentos y el componente 3 reside en el sistema existente a la cual se le está incorporando la solución de firma electrónica de documentos.

Para firmar los documentos, el Componente de Firma listará los certificados almacenados en los siguientes repositorios:

- Tokens USB o Smartcards conectadas y con drivers instalados (Firma tipo PKCS#11).

- Archivos de tipo P12 o PFX que el usuario manualmente seleccionara (Firma tipo P12). Las firmas de este tipo NO son firma digital avanzada como la define la ley Nª 18.600.

Los pasos necesarios para firmar digitalmente un conjunto de documentos son los siguientes:

1. El Componente de Firma accede a la colección de documentos a firmar, pasando como parámetro el ID de transacción al Servicio Web de integración.
2. En caso de estar configurada la visualización de documentos, el Componente de Firma despliega los documentos a firmar, según su tipo.
3. El Componente de Firma accede a los almacenes de certificados para listar al usuario todos los certificados presentes o da la opción de que manualmente seleccione un archivo de tipo P12 o PFX.
4. El usuario selecciona el certificado con el cual firmará los documentos.
5. El usuario selecciona la opción de firmar los documentos.
6. El Componente de Firma firma los documentos con el certificado seleccionado en el punto 2.
7. El Componente de Firma envía los documentos firmados (pasando como parámetro el ID de la transacción)
8. A nivel de estándares de firma, se soporta la firma de documentos PDF, con el estándar PDFSignature, XML con los estándares XMLDSig y XADES, y el cifrado directo de hashes soportando SHA-1 y SHA-256.

Componente de firma

Integración

Para incluir el Componente de Firma en el sistema existente se deben realizar los siguientes pasos:

1. Instalación de los Servicios Web de Integración (AgesicFirmaWS) entregado como parte de la solución.
2. Configurar el Componente de Firma según las necesidades de firma electrónica del sistema existente.
3. Incorporación del Componente de Firma en la aplicación Web del sistema existente.

Para más detalle sobre la integración se recomienda la lectura del documento Manual Técnico Web Service Firma-Electrónica-Avanzada.doc

Configurar el componente

Para configurar el Componente de Firma se debe de realizar las siguientes configuraciones:

- En el caso de ejecutar el Componente de Firma como Applet se debe de configurar configuration.properties que se encuentra en el archivo “AgesicFirmaApplet-AgesicFirmaApplet.jar”. En el caso de ejecutarlo como Java Web Start se debe de agregar los parámetros en el archivo jnlp que ejecute dicho Componente de Firma.
- messages_LOCALE.properties donde LOCALE toma el valor del LOCALE definido. Contiene todos los mensajes y etiquetas que el Componente de Firma despliega al usuario.*

Nombre	Descripción
AGESIC_FIRMA_WS	La dirección del WSDL de los servicios Web de Integración
URL_OK_POST	Url de tipo http o https donde se envía el ID de transacción una vez que se haya realizado la firma en forma correcta.
SHOW_CONTINUE	true si visualiza el botón de continuar una vez que la firma se efectuó de forma correcta, false en otro caso
SHOW_CLOSE	true si visualiza el botón de cerrar una vez que se carga el Componente de Firma.
PDFSIGNATURE_APPEARANCE_ENABLED	true Si se habilita que la firma se vea en el PDF como parte del documento
WS_REQUEST_TIMEOUT WS_CONNECT_TIMEOUT	El timeout para esperar respuesta y conexión en la invocación de los servicios Web de integración en mili segundos
IMAGE_AGESIC	En caso de configurar un valor para esta propiedad, indicar la imagen correspondiente en base64. Esta imagen será agregada al Componente de Firma en la esquina inferior izquierda.

IMAGE_ORG	En caso de configurar un valor para esta propiedad, indicar la imagen en base64. Esta imagen será agregada al Componente de Firma en la esquina inferior derecha.
IMAGE_AGESIC_ORG_W	Ancho para ambas imágenes
IMAGE_AGESIC_ORG_H	Alto para ambas imágenes
ALADDIN_WIN / ALADDIN_LIN / ALADDIN_MAC	Rutas candidatas de la librería para el eToken Aladdin por sistema operativo.
EPASS2003_WIN / EPASS2003_LIN / EPASS2003_MAC	Rutas candidatas de la librería para el eToken ePass3003 Auto por sistema operativo.
LIB_ALADDIN_WIN / LIB_ALADDIN_LIN / LIB_ALADDIN_MAC	En caso que el usuario deba indicar dónde se encuentra la librería del eToken Aladdin, se le mostrará, para su sistema operativo, el nombre de la librería esperada, indicada en la propiedad correspondiente. Por ejemplo: LIB_ALADDIN_WIN= eToken.dll
LIB_EPASS2003_WIN/LIB_EPASS2003_LIN / LIB_EPASS2003_MAC	En caso que el usuario deba indicar dónde se encuentra la librería del eToken ePass2003, se le mostrará, para su sistema operativo, el nombre de la librería esperada, indicada en la propiedad correspondiente.
LIB_EPASS3003_WIN/LIB_EPASS3003_LIN/LIB_EPASS3003_MAC	En caso que el usuario deba indicar dónde se encuentra la librería del eToken ePass3003 Auto, se le mostrará, para su sistema operativo, el nombre de la librería esperada, indicada en la propiedad correspondiente.

Una vez modificado el archivo de configuración es necesario realizar la firma de los jars que componen el Componente de Firma con el certificado del organismo.

Nota: los jars sólo deben tener una firma.

Incorporación del Componente de Firma como Java Web Start

Para incorporar el Componente de Firma en la aplicación Web como una aplicación JavaWeb Start, se debe:

- Crear desde la aplicación Web existente el archivo JNLP

Agesic

Agesic Firma
Agesic Firma

-ID_TRANSACCION=$ID_TRANSACCION$
-TIPO_DOCUMENTO=$TIPO_DOCUMENTO$
-AGESIC_FIRMA_WS=$AGESIC_FIRMA_WS$
-URL_OK_POST=$-URL_OK_POST$

- El Componente de Firma debe recibir como parámetro los siguientes elementos:

Parametro	**Descripción**
ID_TRANSACCION	El identificador de la transacción el cual se obtiene al invocar a los servicios Web de integración por parte del sistema donde se está incluyendo el Componente de Firma. El identificador de la transacción es el utilizado por el Componente de Firma para invocar los servicios Web de integración.
TIPO_DOCUMENTO	El tipo de los documentos que deberá firmar el Componente de Firma, los tipos permitidos son: xml , pdf o hash

AGESIC_FIRMA_WS	La dirección del WSDL de los servicios Web de Integración
URL_OK_POST	Url de tipo http o https donde se envía el ID de transacción una vez que se haya realizado la firma en forma correcta.
CODEBASE	La URL de descarga de los archivos jar indicados en resources.
DOCUMENTO_VISIBLE	Si el Componente de Firma debe visualizar los documentos que serán firmados. Los valores posibles son true o false
LOCALE	El locale del archivo de mensajes
USUARIO_DOCUMENTO	true si el documento a firmar debe de ser seleccionado por el usuario y no descargado desde los servicios Web de integración
USUARIO_DOCUMENTO_FIRMADO	true si los documentos firmados serán almacenados por el usuario en su PC y NO deben de ser enviados al sistema existente por medio de los servicios Web de Integración.

En caso que en el archivo html que incluye el Componente de Firma, se configuren parámetros incluidos en el archivo configuration.properties de “AgesicFirmaApplet-AgesicFirmaApplet.jar”, la aplicación tomará los indicados en el archivo JNLP.

Integración del Componente de Firma como un Applet

Advertencia: el uso del componente como Applet Java es desaconsejado, debido a que los principales navegadores ya no dan soporte para dicha tecnologia

Para incorporar el Componente de Firma en la aplicación Web como un Applet, se debe incluir el siguiente código HTML en la página correspondiente:

- Código JavaScript

La función cerrarApplet y continuarApplet debe de ser implementado según las necesidades de la aplicación Web.

La función cerrarApplet es invocada cuando se realiza clic en el botón cerrar del Componente de Firma y la función continuarApplet es invocada cuando la firma se realizó de forma correcta y se realiza clic en el botón continuar.

Un ejemplo de implementación de cerrarApplet puede ser navegar a una URL como se muestra en el siguiente ejemplo:

```
function getApplet(name){
if (window.document[name]) {
return window.document[name];
}
if (navigator.appName.indexOf("Microsoft Internet")==-1) {
if (document.embeds && document.embeds[name])
return document.embeds[name];
} else {
return document.getElementById(name);
}
}
function cerrarAppletImp(){
try{
getApplet('signApplet').navegarApplet("http://www.sofis-solutions.com");
}catch(e){
}
}

function cerrarApplet() {
cerrarAppletImp();
}
```

- Código HTML que incluye el Componente de Firma

El Componente de Firma debe recibir como parámetro los siguientes elementos:

Parametro	**Descripción**
ID_TRANSACCION	El identificador de la transacción el cual debe generar el sistema donde se está incluyendo el Componente de Firma. El identificador de la transacción es el utilizado por el Componente de Firma para invocar los servicios Web de integración.
TIPO_DOCUMENTO	El tipo de los documentos que deberá firmar el Componente de Firma, los tipos permitidos son: xml , pdf o hash
AGESIC_FIRMA_WS	La dirección del WSDL de los servicios Web de Integración
URL_OK_POST	Url de tipo http o https donde se envía el ID de transacción una vez que se haya realizado la firma en forma correcta.
DOCUMENTO_VISIBLE	Si el Componente de Firma debe visualizar los documentos que serán firmados. Los valores posibles son true o false
CODEBASE	La URL de descarga de los archivos jar indicados en el campo “archive”.
CODE	Especifica el nombre del archivo principal para ejecutar el Componente de Firma.
ARCHIVE	Los jars utilizados por el Componente de Firma. El nombre de los jars se componen del nombre del mismo y de la versión utilizada: nombre-x.0.jar, donde x se corresponde al número de versión, por ejemplo, para el jar icepdf-core-4.3.2 y versión 1.0 el nombre completo del jar sería icepdf-core-4.3.2-1.0.jar
LOCALE	El locale del archivo de mensajes
USUARIO_DOCUMENTO	true si el documento a firmar debe de ser seleccionado por el usuario y no descargado desde los servicios Web de integración
USUARIO_DOCUMENTO_FIRMADO	true si los documentos firmados serán almacenados por el usuario en su PC y NO deben de ser enviados al sistema existente por medio de los servicios Web de Integración.

En caso que en el archivo html que incluye el Componente de Firma, se configuren parámetros incluidos en el archivo configuration.properties de “AgesicFirmaApplet-

AgesicFirmaApplet.jar", la aplicación tomará los indicados en el archivo html.

FIRMASERVER

Firma Server tiene como rol actuar de intermediario, publicando los servicios para consumo por parte del Componente de Firma, y publicando otros servicios para consumo de la aplicación Web integrada al Componente de Firma. A continuación se presenta un diagrama de arquitectura que ilustra este diseño.

Como se aprecia en la imagen anterior el Componente de Firma el cual está ejecutando en la PC del cliente que accede a la aplicación Web consume servicios Web que expone el componente Firma Server.

Adicionalmente la aplicación Web también consume servicios que expone Firma Server, por ejemplo para obtener el identificador de la transacción que define los documentos a firmar por el Componente de Firma.

Integración

El Componente de Firma internamente consume las siguientes operaciones del servicio Web de nombre AgesicFirmaWS que expone FirmaServer:

• obtenerDocumentos: recibe como parámetro el id de la transacción y retorna el conjunto de documentos a desplegar por el Componente de Firma para ser firmados por el usuario.

• comunicarDocumentos: recibe como parámetro el id de la transacción , el conjunto de documentos con la firma electrónica incorporada y el certificado (clave pública) para validación de la firma. Realiza las validaciones correspondientes y retorna el valor 0 en caso que la operación se haya efectuado en forma correcta, o el valor 1 en otro caso, retornando un mensaje genérico de error al usuario.

La aplicación Web que integra el Componente de Firma podrá consumir las siguientes operaciones del servicio Web de nombre AgesicFirmaServerWS que expone Firma Server:

• firmarDocumentos: recibe como parámetro un conjunto de documentos para firmar, junto con el tipo de documento del que se trata (tipo de firma a realizar), los almacena en la tabla de transacciones internas que mantiene Firma Server asociados a un ID de Transacción aleatorio que generará en el momento, y devuelve dicho ID a la aplicación Web para que le quede como referencia.

• obtenerDocumentosFirmados: recibe como parámetro el id de la transacción y devuelve el conjunto de Documentos firmados, junto con la información de la validación de la firma y certificado correspondiente, eliminando la solicitud de la tabla interna en el proceso.

Configuración

Para configurar el servicio web, se debe configurar en el archivo configuration.properties, ubicado en \AgesicFirmaWS\src\java\org\agesic\firma\config, las siguientes propiedades:

Nombre	Descripción
SIGN_ALG	Algoritmo de firma SHA1withRSA o SHA256withRSA. Debe coincidir con el configurado en el Componente de Firma
TIME_RUN_CLEAN_BD_MS	Cada cuánto se verifican los datos para eliminar aquellos caducados, en milisegundos. Cada 1 día por defecto
CLEAN_EXPIRED_AFTER	Tiempo que se considera que un registro a caducado, en milisegundos. Configurado en 1 hr por defecto.

Instalación

Firma Server es un WAR de nombre AgesicFirmaWS.war el cual puede ser instalado en Tomcat o en Jboss 7.

Firma Server mantiene una tabla de transacciones, para lo cual debe acceder a una Base de datos la cual debe ser configurada previa a la instalación.

El nombre de la base de datos es AGESIC_FIRMA, las tablas necesarias por Firma Server son creadas por este, pero si debe existir previamente la base de datos creada.

Para instalar Firma Server en Jboss 7 debe:

• Utilizar la versión del componente AgesicFirmaWS.war para Jboss

• Configurar en el archivo de configuración de Jboss (por ejemplo standalone.xml o domain.xml según el nodo utilizado) la configuración para Web Service agregando al subsistema urn:jboss:domain:webservices

true

jbossws.undefined.host

• Agregar en el Jboss el módulo con el driver de la base de datos a utilizar.

o Acceda a la carpeta {JBOSS_HOME}/modules y dentro de la carpeta modules se debe crear la siguiente estructura de directorios según el motor de base de datos a utilizar, por ejemplo para postgresql: org\postgresql\main

o Luego, en la carpeta main, se deben copiar el archivo con el driver JDBC del motor de la base de datos y crear el archivo module.xml, por ejemplo para el caso de postgresql el archivo module.xml es de la siguiente forma:

Donde postgresql-9.1-902.jdbc4.jar es el jar copiado a la carpeta

org\postgresql\main

• Configurar en el Jboss el Data Source necesario

o Configurar en el archivo de configuración de Jboss (por ejemplo standalone.xml o domain.xml según el nodo utilizado) el Data Source agregando al subsistema urn:jboss:domain:datasources un nuevo Data Source

jdbc:{MOTOR_BD}://{host}:{port}/AGESIC_FIRMA agesic_firma_d

{user}

{password}

Remplazar "{host}" y "{port}" por la dirección IP y número de puerto donde escucha el motor de bases de datos.

Remplazar "{user}" y "{password}" por el nombre de usuario y la contraseña del servidor de base de datos.

Remplazar "{MOTOR_BD}" por el correcto dependiendo del motor de la base de datos por ejemplo para postgresql reemplazar con el valor postgresql

A continuación, en el mismo archivo de configuración del jboss y aún dentro del mismo subsistema, ir hasta la sección y copiar el siguiente texto:

{CLASS_DRIVER}

Remplazar "{ MODULE_NAME}" por el nombre definido en el archivo module.xml, para el ejemplo postgresql indicado es el valor org.postgresql

Remplazar "{CLASS_DRIVER }" por el nombre de la clase correcta del driver JBDC del motor, para el caso de postresql el valor: org.postgresql.Driver

• Configurar AgesicFirmaWS según el capitulo anterior

• Deployar el War en el Jboss.

Para instalar Firma Server en Tomcat debe:

• Utilizar la versión del componente AgesicFirmaWS.war para Tomcat

• Incluir el driver de la base de datos utilizada en el WAR en la carpeta WEB-INF/lib

• Configurar el acceso a la base de datos editando el archivo persistence.xml incluido en AgesicFirmaWS.war

o Editar el archivo de nombre persistence.xml que se encuentra en WEB-INF\classes\META-INF y modificar los valores de las siguientes propiedades por los correctos.

• hibernate.connection.url: La URL de conexión a la base de datos, depende del manejador de base de datos seleccionado

• hibernate.connection.username: El usuario para acceder a la base de datos AGESIC_FIRMA.

• hibernate.connection.password: La contraseña para acceder a la base de datos AGESIC_FIRMA.

• hibernate.connection.driver_class: El driver JDBC del manejador

• hibernate.dialect: el dialecto de la base de datos a utilizar, la lista de dialectos validos se encuentra en la documentación de Hibernate.

• Configurar AgesicFirmaWS según el capitulo anterior

• Deployar el war en el Tomcat.

JAVA: implementación de referencia con servicio de firma con Cédula de Identidad Digital

Acceder al código fuente de la implementación de referencia.

Requisitos para la maqueta:

1. Instalar maven.
2. Instalar BouncyCastle Provider.

- Bajar la última versión del Provider (v1.57, actualmente).
- Editar archivo $JAVA_HOME/jre/lib/security/java.security; en la lista de security providers, agregar bouncycastle con la siguiente linea: 1.security.provider.N=org.bouncycastle.jce.provider.BouncyCastleProvider (siendo N el último número de la lista +1)
- Copiar la librería BouncyCastle a la carpeta $JAVA_HOME/jre/lib/ext

Clonar el repositorio del git y ejecutar el comando "mvn build".

Se descargarán todas las dependencias necesarias para que el proyecto funcione.

Luego, se deberán editar los archivos de configuración: application.properties y application-{ambientes}.properties

application.properties

Se pueden configurar el puerto y el contexto en el cual correrá la aplicación.

También se puede cambiar el nivel de logging y el tamaño de archivos que se permiten subir para firmar.

application-{ambiente}.properties

Estos archivos están pensados para que cada ambiente tenga su configuración particular. Para el caso de la maqueta, editar únicamente el ambiente **dev,** ya que es el que corre por defecto (esto está definido en el archivo pom.xml).

Integración con DSS de Agesic

Para poder utilizar el DSS, es necesario darse de alta como RP (Relaying Party). Para ello, hay que completar un formulario completando, fundamentalmente, lo siguiente:

- Nombre de servicio: Este campo se relaciona con el valor "dss.service.name" en el archivo application.properties.
- URL de respuesta: Este campo se relaciona con el valor "dss.response.url" en el archivo applicacion.properties.
- También se debe adjuntar un certificado público del firmante para poder validar el request que se envía al DSS para firmar; por otro lado, se recibirá el certificado público del DSS de Agesic para poder validar el response del DSS firmado.

Se deben editar las rutas donde está el keystore para firmar el pedido al DSS y el keystore que valida la respuesta del DSS. Para crear un keystore, se puede utilizar el comando "openssl" o usar alguna aplicación como, por ejemplo, "Keystore Explorer".

Kit de desarrollo de Cédula de Identidad Digital

Se adjunta el contrato de adhesión para solicitar kits de desarrollo de Cédula de Identidad Digital.

El kit contiene una cédula de test y un lector de cédula. También se adjunta el procedimiento para completar el Contrato de Adhesión y los componentes exactos del kit.

El Contrato de Adhesión debe ser firmado por un representante legal de la empresa y se debe presentar certificado notarial que acredite tal calidad. Quedan exentas de presentar el certificado notarial aquellas empresas cuyos representantes legales se encuentren en RUPE como tal.

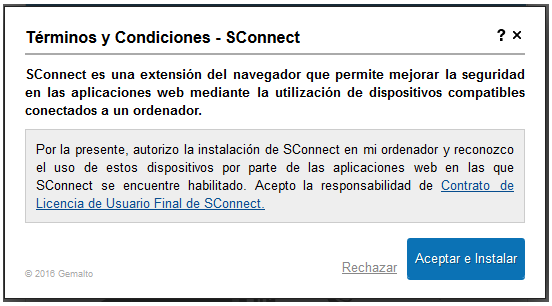
- Procedimiento: kits de desarrollo para terceros
- Contrato de adhesión Cédulas espécimen Terceros

Tutorial de instalación de plugins para uso de cédula en navegador

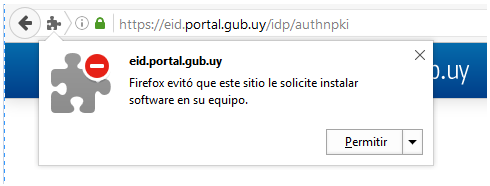
Instalar sconnect en Mozilla Firefox

Al seleccionar ingreso con Cédula de Identidad Digital, deberás insertar el dispositivo en el lector de tarjetas inteligentes y conectarlo a tu computadora.

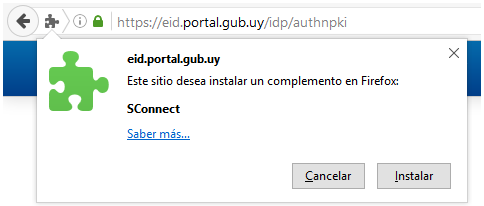
Luego se disparará automáticamente el asistente para la instalación del Plugin SConnect. En primer lugar, se desplegarán los términos y condiciones del plugin, los cuales deberán ser aceptados.



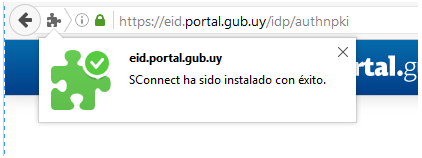
A continuación, deberás hacer clic en "Permitir" para que tu navegador web autorice la descarga de SConnect.



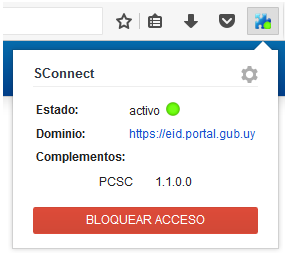
Una vez descargado, deberás instalar el plugin haciendo clic en "Instalar".



Cuando el SConnect se haya instalado correctamente, se mostrará el siguiente mensaje.



También aparecerá un nuevo ícono en la esquina superior derecha de tu navegador web, como se muestra en la siguiente imagen.


SConnect
Estado:
activo
Dominio:
https://eid.portal.gub.uy
Complementos:
PCSC
1.1.0.0
BLOQUEAR ACCESO

Guía de uso de Cédula de Identidad con chip a través de APDU

En el presente capítulo se presentan las dos versiones existentes del Macth on Card Y NFC para los dos tipos de chip existentes en las cédulas electrónicas expedidas por estado Uruguayo

Para diferenciar las versiones del chip se puede aplicar Los comandos a continuación

Lo siguiente son los ejemplos de respuesta esperadas de las dos versiones.

IAS CLASSIC v4:

Software Version

The command returns the software version in TLV format as follows:

Tag	Length	Meaning	Display Value	ASCII Value
COh	OEh	Applet Label	"LAS Classic v4"	49 41 53 20 43 6C 61 73 73 69 63 20 76 34h
C1h	07h	Applet Version	4.0.x.y	34 2E 30 2E x 2E yh

The DO with tag COh contains the product reference.
The DO with tag C1h contains the product version.
The software version is an example and will evolve as the product evolves.

IAS CLASSIC v5:

The command returns the software version in the TLV format as follows:

Tag	Length	Meaning	Display Value	ASCII Value
coh	OEh	Applet Label	IAS Classic	49h 41h 53h 20h 43h 6Ch 61h 73h 73h 69h
C1h	07h	Applet Version	M.m.r.p.c	M: major version m: minor version r: release version (0-9") p:product c: configuration. 'C' for Full, 'O* for Compact. 35h 2Eh 32h 2Eh 30h 2Eh 41h 2Eh 4Fh (Compact configuration on MultiappV5.0) 35h 2Eh 32h 2Eh 30h 2Eh 41h 2Eh 43h (Full configuration n MultiappV5.0)

The DO with tag COh contains the ASCII value of the product reference.
The DO with tag C1h contains the ASCII value of the product version.

Este es un resumen de las respuestas de las versiones:

Operation	APDU	Response	Analysis
Select IAS Classic instance	00A40400 0C A000000018400000001634200	9000	
Send the get data TAG 7F30	00CA7F30 Le=00	Response on IAS Classic V5 → Tarjetas duales 7F301B C00E **49415320436C6173736963207635** C109 **352E32**2E302E412EXX Response on IAS Classic V4 → Tarjetas hibridas 7F3019 C00E **49415320436C6173736963207634** C107 **342E30**2E302E41	Response on IAS Classic V5 → Tarjetas duales 49415320436C6173736963207635 352E322E302E412EXX → 5.2.0.A.? Response on IAS Classic V4 → Tarjetas hibridas **49415320436C6173736963207634** 342E302E302E41 → 4.0.0.A

MOC Cédula electrónica chip 2015

Guía de uso de Cédula de Identidad Digital a través de APDU

¿Qué es un APDU y por qué es relevante?

El Application Protocol Data Unit (APDU) es la unidad de comunicación entre un lector de tarjetas inteligentes y una tarjeta inteligente (en inglés, smart card). Dado que la Cédula de Identidad Digital es, en esencia, una smart card conformante con el estándar ISO 7816, esta es la unidad lógica utilizada para comunicarse con ella a bajo nivel.

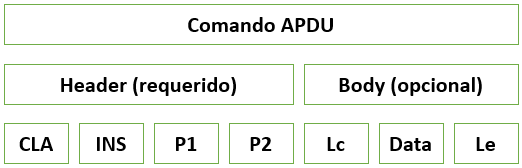
Si bien existen otras vías para comunicarse con la Cédula de Identidad Digital, como drivers PKCS#11, plug-ins, bibliotecas, etc., todas estas vías se implementan utilizando APDU, es decir, son wrappers. Poder interactuar con las aplicaciones de la Cédula de Identidad Digital a través de APDU tiene la ventaja de que otorga la máxima flexibilidad a nivel de plataformas en las que se puede implementar la interacción. Y, además, hay operaciones como el Match On Card (MOC) para las que no se cuenta con una biblioteca de más alto nivel, por lo que solo pueden realizarse a través de esta vía.

Estructura de un APDU

Hay dos tipos de APDU: comandos y respuestas. Los comandos APDU los envía el lector a la tarjeta, mientras que las respuestas APDU las envía la tarjeta al lector.

La estructura de un APDU está definida en los estándares ISO/IEC 7816.

Estructura de un comando APDU



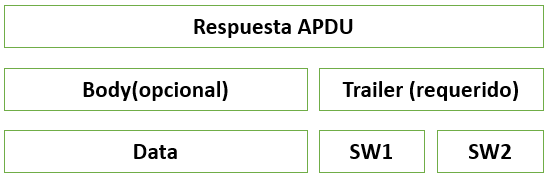
La trama APDU de tipo comando consta de los siguientes campos:

- **CLA:** Byte de clase
- **INS:** Byte de instrucción
- **P1, P2:** Parámetros
- **Lc:** tamaño del bloque de datos
- **Data**
- **Le:** Tamaño de la respuesta esperada

Los cuatro primeros son obligatorios, mientras que los relacionados con los datos y la respuesta esperada son opcionales. A partir del byte de instrucción, la tarjeta sabe qué es lo que se le pide.

Contienen una cabecera obligatoria de 4/5 bytes, y entre 0 y 255 bytes de datos.

Estructura de una respuesta APDU



La trama APDU respuesta consta de los campos:

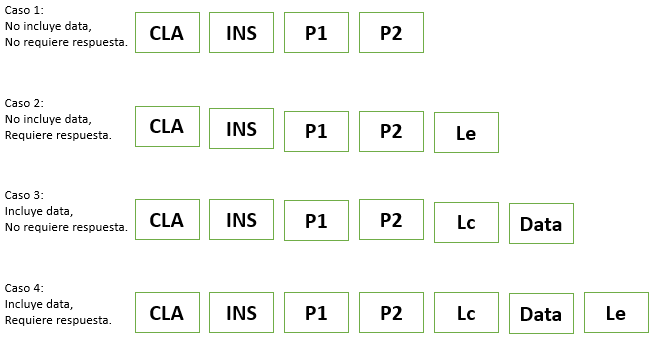
- **Data**
- **SW1, SW2:** Palabra de estado, donde se codifica el estado de la operación (correcta, error criptográfico, error general). Una vez más, los datos son opcionales, pero el código de estado es obligatorio.

Contienen una palabra de estado obligatoria de 2 bytes y entre 0 y 255 bytes de datos.

Casos de APDU

Existen cuatro casos definidos para comandos APDU donde la estructura varía de la siguiente forma:

1. El largo Lc es nulo; por lo tanto, los campos Lc y data van vacíos. El largo Le es también nulo; por lo tanto, el campo Le va vacío. Por consecuencia, el campo Body es vacío.
2. El largo Lc es nulo; por lo tanto, los campos Lc y data van vacíos. El largo Le no es nulo; por lo tanto, el campo Le está presente. Por consecuencia, el campo Body es el Le.
3. El largo Lc no es nulo; por lo tanto, el campo Lc está presente y define el largo de campo Data también presente. El largo Le es nulo; por lo tanto, el campo Le es vacío. Por consecuencia, el Body contiene al campo Lc seguido del campo data.
4. El largo Lc no es nulo; por lo tanto, el campo Lc está presente y define el largo del campo Data también presente. El largo Le no es nulo; por lo tanto, el campo Le esta presente. Por consecuencia, el Body consiste en el campo Lc seguido del campo Data y del campo Le.



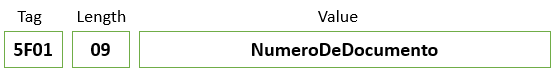
Formato TLV para datos

Para ciertos tipos de operaciones, la información dentro del campo Data en los comandos y respuestas APDU está representada en formato TLV "Tag Length Value" (Tipo Longitud Valor).

Este formato permite organizar mejor la información y tiene la siguiente lógica:

- **Tag:** Representa la información contenida en el campo Value.
- **Length:** Largo en bytes de la información contenida en el campo Value.
- **Value:** Información.

Ejemplo de un TLV que contiene el número de documento de la persona obtenido como parte del caso de uso de extracción de los datos de identificación:



Como se ve en el ejemplo, el TAG 5F01 representa al número de documento que tiene largo en bytes 09.

El campo Length viene habitualmente en un byte, pero cuando el tamaño del Value es mayor a 0x7F (127), en el campo Length el primer bit será 1 y los restantes indican el tamaño en bytes restantes del campo. Esto se traduce a que será 0x80 + tamaño restante de LENGTH. Entonces:

Si 0 = LENGTH 0x7F(127) => LENGTH 1 byte = XX

Si 0x7F LENGTH 0xFF(255) => LENGTH 2 byte = 81 XX

Si 0xFF LENGTH 0xFFFF(65535) => LENGTH 3 byte = 82 XX XX

Por ejemplo, en el caso de obtener la imagen de la persona, el campo Length contendrá un byte con el número 0x82 seguido del largo representado en 2 bytes; por ejemplo, T= 0x3F01 L= 0x8223FE D= 9214 bytes(0x23FE). También ocurre esto cuando se envían 42 o más minucias al match on card.

Comandos APDU y casos de uso

En esta sección se presentan los comandos APDU que luego serán utilizados en conjunto para la construcción de casos de uso como por ejemplo extraer los datos de identificación del documento.

Selección del applet de firma IAS - SELECTIAS

El comando select selecciona el Applet IAS de firma por su AID (Application Identifier) dentro del eID.

Es precondición para cualquier otro comando APDU descripto en este documento que se haga un select del Applet IAS antes.

La estructura de un comando de selección de aplicación por su AID es la siguiente:

Caso 3:
Incluye data,
No requiere respuesta.

CLA	INS	P1	P2	Lc	Data
00	A4	04	00	Lc	AID

El comando APDU que selecciona el applet de firma IAS:

CLA	INS	P1	P2	Lc	Data
00	A4	04	00	0C	A0 00 00 00 18 40 00 00 01 63 42 00 00

IMPORTANTE: El comando selectIAS se debe ejecutar al inicio antes que cualquier otro comando

Selección de un archivo por el ID de archivo - Selectfile

La información contenida en el documento como por ejemplo los datos de identificación o certificado de la persona se encuentra distribuida en archivos y cada archivo se identifica por un ID.

Para realizar la lectura de esta información se debe seleccionar el archivo por su correspondiente ID utilizando el comando APDU selectFile.

Luego de realizada la selección del archivo mediante su ID, el comando APDU readBinary se envía para extraer los datos. Para el envío del comando readBinary se debe conocer el tamaño del archivo a leer.

Este dato necesario para la lectura del archivo se encuentra en lo que se denomina FCI Template. El FCI Template de cada archivo se obtiene de la respuesta APDU al comando selectFile.

Por ejemplo, para leer los datos del certificado del usuario debemos seleccionar primero el archivo correspondiente y obtener su FCI Template.

El FCI Template contiene los siguientes datos de relevancia entre otros:

- Nombre del archivo.
- Tamaño del archivo.

El tamaño del archivo es necesario para realizar la lectura de la información utilizando el comando readBinary.

Entonces, si se quisieran leer los datos del certificado digital contenido en el eID se deben realizar los siguientes pasos:

1. Obtener el archivo FCI Template del certificado a través de su ID y la operación selectFile.
2. Extraer del FCI Template obtenido en el comando APDU respuesta al comando selectFile el tamaño en bytes del certificado.
3. Conociendo el tamaño del archivo que contiene el certificado se ejecuta el comando APDU readBinary para obtener la información del certificado.
4. Estructura del comando selectFile:

Caso 4:
Incluye data,
Requiere respuesta.

CLA	INS	P1	P2	Lc	Data	Le
00	A4	00	00	02	FileID	Le

Ejemplo de un comando de selectFile para seleccionar el FCI Template del certificado.

CLA	INS	P1	P2	Lc	Data	Le
00	A4	00	00	02	B0 01	Le

En la respuesta viene el archivo FCI Template correspondiente al certificado. El ID del archivo que contiene al certificado es **"B001"** como se ve en el ejemplo.

0	6Fh	Tag del FCI Template
1	L	Largo de FCI Data
2	81h	Tag de largo de archivo
3	02h	Largo del tamaño del archivo
4–5	Tamaño de archivo	Valor del tamaño del archivo
6	82h	FDB Tag
7	01h	Largo del FDB
8	FDB	Valor del FDB
9	83h	Tag del ID de archivo
10	02h	Largo del archivo ID
11–12	ID Archivo	Valor del ID archivo
13	8Ah	Tag de “Life Cycle Status byte for file"
14	01h	Largo de “Life Cycle Status byte for file”
15	Var.	Valor de “Life Cycle Status byte for file”
16	8Ch	Tag de atributos de seguridad
17	L	Largo de atributos de seguridad
18	AMB	Modo acceso a bytes
19–(18+X)	SCBs	Condición de seguridad bytes (X)

Como se ve en la imagen, el FCI template contiene el tamaño del archivo a leer (offset 4-5), necesario y suficiente para su lectura.

Lectura de un binario - Readbinary

La lectura de un binario se realiza sobre un archivo previamente seleccionado con el comando APDU selectFile.

Cada comando readBinary puede leer como máximo 0xFF bytes de un archivo, si el tamaño del archivo es superior a 0xFF se deberán enviar tantos comandos APDU readBinary como sean necesarios.

Por ejemplo, si el tamaño del certificado a leer (obtenido del FCI Template del archivo) es de 3.000 bytes, se deberán enviar 3000/255 + 1 comandos de readBinary para completar la lectura del certificado. En este caso, 12.

Estructura del comando readBinary:

	CLA	INS	P1	P2	Le
Caso 2: No Incluye data, Requiere respuesta.	**00**	**B0**	**P1**	**P2**	**Le**

Donde P1 P2 es el offset en hexadecimal donde empezar a leer datos.

Ejemplo de un comando readBinary obteniendo los primeros 255 bytes (0xFF) del archivo seleccionado:

CLA	INS	P1	P2	Le
00	**B0**	**00**	**00**	**FF**

Para los siguientes READ_BINARY se debe sumar 0xFF al offset y continuar leyendo datos, por ejemplo si L es el largo del archivo en hexadecimal:

00 B0 00 FF Le=FF

00 B0 01 FE Le= FF

00 B0 02 FD Le=FF

.

.

.

00 B01 L1 L2 Le=L3

Donde:

L1 || L2 = 0xFF x cociente(L/0xFF)

L3 = resto(L/0xFF)

Verificación de PIN - VERIFYPIN

La operación de verificación de PIN es requisito previo a las operaciones de firma.

	CLA	INS	P1	P2	Lc	Data
Caso 3: Incluye data, No requiere respuesta.	**CLA**	**20**	**00**	**P2**	**Lc**	**Pin de Verificación**

- **CLA:**
  - 00h Transmisión en plano
  - 0Ch Transmisión sobre canal seguro
- **INS:**
  - 20h Fijo para la operación de verificación.
- **P1:**
  - 00h Modo verificación
- **P2:**
  - 11h Para global PIN
- **Lc:**
  - 0Ch Siempre espera 12 bytes de largo, se agregan 0's al final como padding.

- **Data:**
  - El PIN va en codificado en ASCII y largo 12 bytes.

Ejemplo de un comando APDU para verificar el PIN 1234:

CLA	INS	P1	P2	Lc	Data
00	20	00	11	0C	31 32 33 34 00 00 00 00 00 00 00 00

Validar PIN verificado - ISVERIFIEDPIN

Valida si el PIN se encuentra verificado.

Caso 1:
No Incluye data,
No requiere respuesta.

CLA	INS	P1	P2	Le
CLA	20	00	P2	00

- **CLA:**
  - 00h Transmisión en plano
  - 0Ch Transmisión sobre canal seguro
- **INS:**
  - 20h Fijo para la operación de verificación.
- **P1:**
  - 00h Modo verificación
- **P2:**
  - 11h Para global PIN
- **Le:**
  - 00h No espera respuesta pero se debe enviar Le o Lc con 00h.
- **Data:**
  - Ausente

Ejemplo de un comando APDU para verificar si el PIN se encuentra verificado:

CLA	INS	P1	P2	Le
00	20	00	11	00

Validación de la huella digital de la persona MATCH ON CARD

Realiza la operación de Match On Card. Validación 1 a n comparando las minucias extraídas por un lector de huellas versus las minucias almacenadas en el chip del documento electrónico.

Las minucias deben ser extraídas conforme al estándar **ISO/IEC 19794-2** Compact Card, sin cabezales, es decir, solamente la información de las minucias.

En este formato, cada punto característico de la huella dactilar se corresponde a una minucia, que a su vez es codificada en 3 bytes: uno para la coordenada X, otro para la coordenada Y, y un tercero para el tipo de punto (valle, bifurcación, etc) y el ángulo de inclinación de su característica. En ese formato, un conjunto de minucias debe ser entonces siempre de una cantidad de bytes múltiplo de 3, de la forma:

X1|Y1|T1|X2|Y2|T2|....|Xn|Yn|Tn(el | marca la división de bytes)

Las minucias deben ordenarse primero ascendente por la coordenada Y y luego, en caso de dos minucias con igual Y, ascendente por la coordenada X; de lo contrario, el match fallará con error 6CXX (siendo XX la cantidad de intentos restante) o 6A80, que indica error de formateo de datos. Se recomienda tener especial cuidado con lenguajes de programación que manejen bytes con signo como Java para la extracción de minucias. En esos casos, cualquier comparación susceptible al signo (o >) debe ser hecha enmascarando los bytes con un bitwise AND (AND bit a bit, & en Java y C/C++) con 0xFF, lo cual fuerza a que sean considerados como bytes sin signo.

El largo máximo de las minucias soportado es de 192 bytes, es decir, 64 minucias de 3 bytes cada una.

Estructura del comando APDU para la operación Match On Card.

Caso 3:
Incluye data,
No requiere respuesta.

CLA	INS	P1	P2	Lc	Data
00	21	00	21	Lc	TLV Data

Tag	Length	Value
7F2E	L	TLV Minucias

Tag	Length	Value
81	L	Minucias

- **CLA:**
  - 00h Transmisión en plano
  - 0Ch Transmisión sobre canal seguro
- **INS:**
  - 21h Fijo para la operación de Match on Card.
- **P1:**
  - 00h Fijo para la operación de Match on Card.
- **P2:**
  - 21h Fijo para la operación de Match on Card.
- **Lc:**
  - Largo en bytes del TLV para validación Match on Card.
- **Data:**
  - TLV para la validación Match on Card que contiene las minucias extraídas de una huella.

Ejemplo de un comando APDU para verificación Match On Card:

CLA	INS	P1	P2	Lc	Data
00	21	00	21	Lc	72 FE 4F 81 4D 761460f21474971...

Importante: La cédula se bloquea luego de cinco intentos consecutivos de match on card sin éxito, devolviendo el error 0x6984

Las minucias para la versión 3 de los especímenes se solicitan a identificacion.electronica@agesic.gub.uy

Extracción de los datos de identificación de la persona

Los datos de identificación de la persona dentro del documento son los que muestra el plástico (menos la imagen de la huella e imagen de la firma).

Estos datos se encuentran en archivos que deberán ser leídos con las operaciones selectFile y readBinary.

La información obtenida de las respuestas APDU a los comandos readBinary para cada archivo está codificada en formato TLV.

A continuación, se presentan las operaciones de selectFile necesarias para la obtención de la información de identificación y la especificación de los TLV correspondientes a cada archivo.

Los datos del campo value se encuentran codificados en formato ASCII con excepción de la fecha de expedición.

selectFile del archivo que contiene el número de documento, ID 7001:

CLA	INS	P1	P2	Lc	Data	Le
00	A4	04	00	02	70 01	Le

Información en formato TLV obtenida luego de las operaciones de readBinary:

Tag	Length	Value
5F01	09	NumeroDeDocumento

selectFile del archivo que contiene los datos biográficos, ID 7002:

CLA	INS	P1	P2	Lc	Data	Le
00	A4	04	00	02	70 02	Le

Información en formato TLV obtenida luego de las operaciones de readBinary:

Tag	Length	Value
1F01	L	PrimerApellido
1F02	L	SegundoApellido
1F03	L	Nombres
1F04	03	Nacionalidad (ISO 3166)
1F05	08	FechaDeNacimiento (DDMMAAA)
1F06	L	LugarDeNacimiento (Ciudad/Pais en ISO 3166)

selectFile del archivo que contiene la imagen en formato JPG, ID 7004:

CLA	INS	P1	P2	Lc	Data	Le
00	A4	04	00	02	70 04	Le

Información en formato TLV obtenida luego de las operaciones de readBinary: (Length = 2 bytes)

Tag	Length	Value
3F0182	L	ImagenJPG

selectFile del archivo que contiene el MRZ, ID 700B:

CLA	INS	P1	P2	Lc	Data	Le
00	A4	04	00	02	70 0B	Le

Información en formato TLV obtenida luego de las operaciones de readBinary:

Tag	Length	Value
7F01	L	MRZ

Firma Digital

Antes de ejecutar alguna operación de Firma Digital se debe haber ejecutado la operación de verificación de PIN

La operación de Firma Digital consta del cifrado de un hash utilizando la clave privada del documento digital y un algoritmo seleccionado. Los algoritmos soportados para la operación de Firma Digital son RSA y ECDSA, aunque actualmente las claves de la Cédula Digital son RSA de 2048 bits.

De forma macro, los pasos para realizar la operación de firma son los siguientes:

1- Se realiza un hash del mensaje a firmar, el hash puede ser realizado de tres formas:

- Externo al eID (el más utilizado).
- Utilizando el eID.
- Parcialmente externo y utilizando el eID.

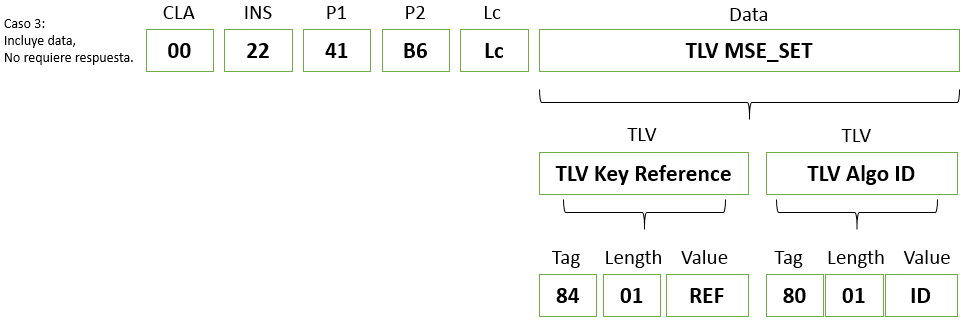
2. Se selecciona el algoritmo de firma y hash utilizando el comando APDU MSE_SET_DST.

3. Se envía el hash al documento eID mediante el comando APDU PSO_HASH.

4. El hash es cifrado con la clave privada del documento eID utilizando el algoritmo y parámetros seleccionados en los pasos anteriores mediante el comando APDU PSO–Compute Digital Signature.

5. Se obtiene el hash cifrado como resultado del paso anterior.

Comando APDU MSE_SET_DST:

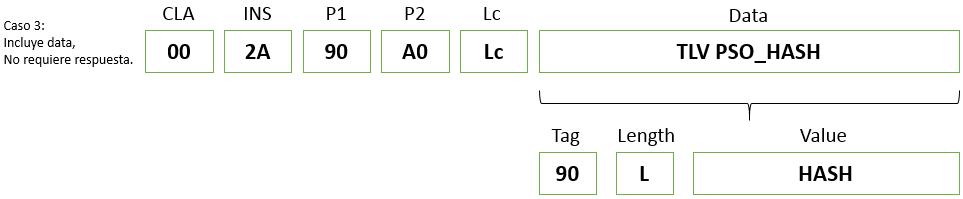


AlgoIDs:

01, 02, 03 = No hash
32, 35 = SHA224
41, 42, 45 = SHA256
52, 55 = SHA384
62, 65 = SHA512
x1=RSA padding ISO9796-2
x2=RSA padding PKCS#1v1.5
x3=RSA padding RFC2409
x5=RSA PSS

Por ejemplo, para utilizar RSA con hash SHA-256 y padding PKCS#1v1.5 utilizaremos el AlgoID=42

Comando APDU PSO_HASH: (Hash realizado fuera de la tarjeta)



Comando APDU PSO–Compute Digital Signature:

Como la clave utilizada es RSA de 2.048 bits, se espera un resultado de largo 256 bytes = 0x100.

Ejemplo de Firma Digital

A continuación, se presentan como ejemplo los comandos APDU enviados para la Firma Digital de un hash de un mensaje generado de forma externa con el algoritmo SHA-256.

Mensaje a firmar: "Ejemplo de firma en APDU utilizando el nuevo documento eID"
HASH en formato hexadecimal del mensaje con el algoritmo **SHA256**: "A3D00CBE708B435D6E7B898770378FD54319B2FD7571C769DB414094E7008624".

MSE_SET_DST:

CLA	INS	P1	P2	Lc	Data
00	22	41	B6	06	84 01 01 80 01 02

PSO_HASH:

CLA	INS	P1	P2	Lc	Data
00	2A	90	A0	20	A3D00CBE708B435D6E7B898770…

PSO-Compute Digital Signature:

CLA	INS	P1	P2	Le
00	A4	9E	9A	100

Hash cifrado con la clave privada del documento de identidad digital obtenido como resultado al comando PSO-CDS: "A91283AA1239213...".

Repositorios públicos de ejemplo

Existen estos dos desarrollos públicos en Github que utilizan APDU.

Diferentes servicios con APDU

Acceder a un ejemplo de uso con interfaz gráfica, lee los datos públicos de la CI y los muestra en pantalla.

MOC Cédula electrónica Chip 2022

Validación de la huella digital de la persona Match On Card para las nuevas eID (Dual interface)

Estructura del comando APDU para la operación Match On Card.

Caso 3:
Incluye data,
No requiere respuesta.

CLA	INS	P1	P2	Lc	Data
00	21	00	4A	Lc	TLV Data

Tag	Length	Value
7F2E	L	TLV Minucias

Tag	Length	Value
81	L	Minucias

- **CLA:**
  - 00h Transmisión en plano
  - 0Ch Transmisión sobre canal seguro
- **INS:**
  - 21h Fijo para la operación de Match on Card.
- **P1:**
  - 00h Fijo para la operación de Match on Card.
- **P2:**
  - 4Ah Fijo para la operación de Match on Card.
- **Lc:**
  - Largo en bytes del TLV para validación Match on Card.
- **Data:**

TLV para la validación Match on Card que contiene las minucias extraídas de una huella.

Ejemplo de un comando APDU para verificación Match On Card:

CLA	INS	P1	P2	Lc	Data
00	21	00	4A	Lc	72 FE 4F 81 4D 761460f21474971

NFC Cédula electrónica chip 2022 (PACE)

La nueva cédula electrónica chip 2022 cuenta con capacidades de lectura segura por NFC utilizando el protocolo PACE para más información puede remitirse a los siguientes documentos:

Technical Guideline TR-03110. Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token – Part 2: Protocols for electronic IDentification, Authentication and trust Services (eIDAS). Version 2.21. Date: 21. December 2016

- Technical Guideline BSI TR-03111. Elliptic Curve Cryptography. Version 2.10. Date: 2018-06-01
- Doc 9303 - Documentos de viaje de lectura mecánica. Parte 11: Mecanismos de seguridad para los MRTD

Integración de aplicación DSS a ASPNET

El presente documento detalla la solución para el desarrollo de un componente .NET que permita integrar aplicaciones ya desarrolladas sobre esta plataforma, con la solución de firma digital avanzada del portal del Estado uruguayo.

Se busca que el componente pueda ser incorporado en soluciones existentes de forma poco invasiva, de modo de simplificar su uso.

Requisitos

La implementación de referencia se encuentra disponible para clonar en github aquí

Los componentes utilizados en la presente guía requieren que la aplicación se encuentre desarrollada sobre la versión 4.6 del Microsoft .NET Framework, o superior. Asimismo, la misma deberá encontrarse publicada sobre HTTPS.

La aplicación deberá contar con un certificado de servicio emitido por una CA autorizada, el cual deberá encontrarse instalado en el Certificate Store del equipo donde se encuentra la aplicación, incluyendo la clave privada. A los efectos de la presente guía, se asume que el certificado se encuentra instalado en el store Personal del equipo (LocalMachine / My). A continuación, se incluye una captura del Certificate Store:

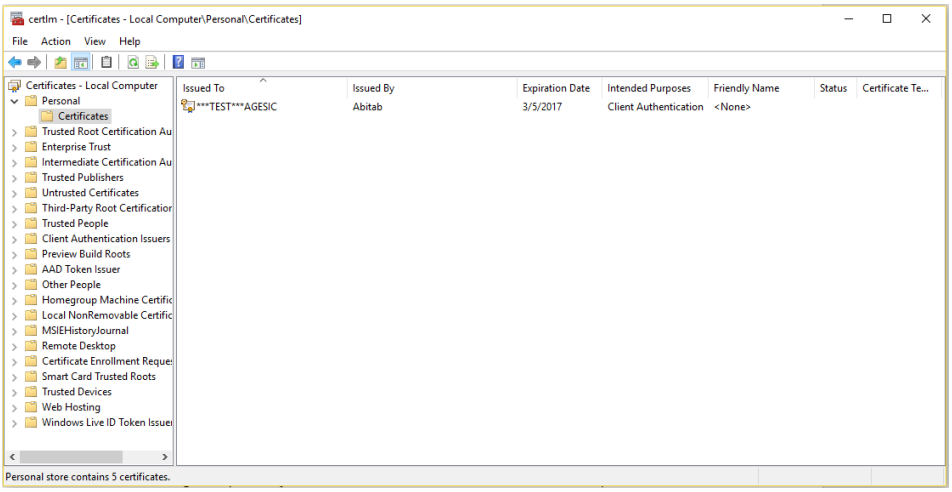


Figura 1. Store de certificados.

La cuenta configurada para la ejecución del sitio web deberá tener acceso a la clave privada del certificado para poder realizar la firma de los SignRequests enviados al IdP.

Aplicación .Net

Para la elaboración de la presente guía, se asume una aplicación .NET existente desarrollada sobre ASP.NET MVC 4, la cual cuenta con dos controladores (HomeController y SignatureController) siendo el primero quien se encarga de desplegar la página de inicio y el segundo que generar el SignRequest y validar el SignResponse.

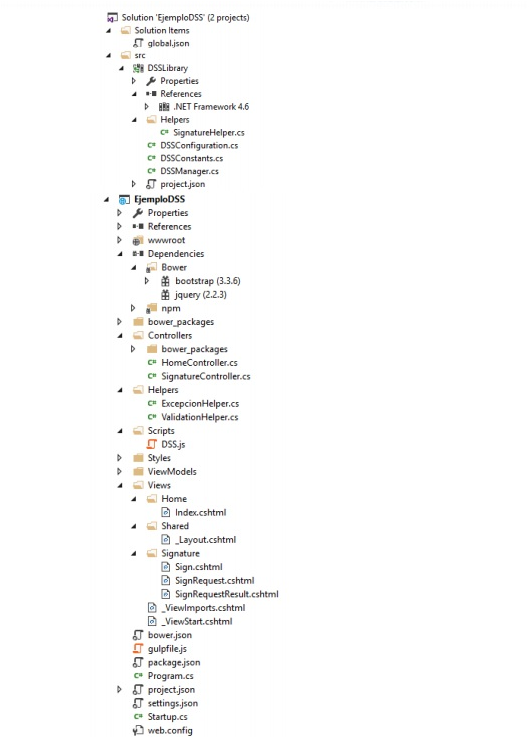


Figura 2. Diagrama de la solución.

En el diagrama anterior se pueden observar los componentes del sitio inicial, el cual cuenta con la siguiente página de inicio:

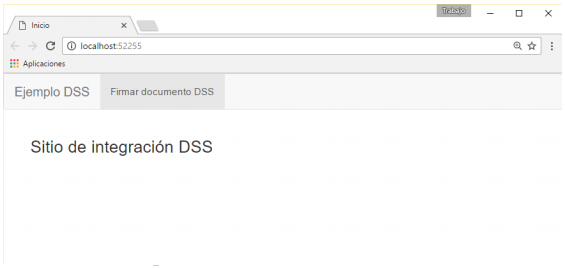


Figura 3. Página principal.

El código fuente de la solución de inicio se encuentra disponible junto con esta guía, de modo de permitir seguir los pasos detallados en la misma sobre esta aplicación de ejemplo.

Para la interacción con el DSS, se utilizó un binding POST. Asimismo, se implementaron los siguientes perfiles de firma de forma parcial, según el alcance detallado más adelante:

-CAdES.

-CAdES Hash.

-PAdES.

-XAdES.

Integración con la dll DSS Library

Incorporar dll DSS Library

El primer paso para integrar la librería del DSS es agregar el .dll como referencia al proyecto. A continuación, se indica cómo realizar esta operación:

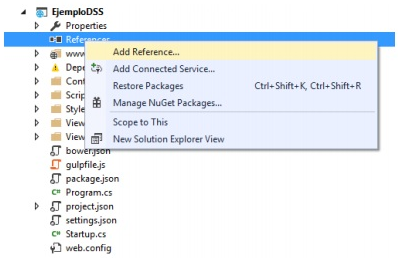


Figura 4. Agregar referencia en el sitio.

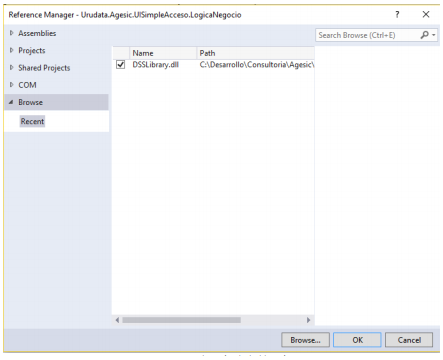


Figura 5. Selección de la librería.

Configurar parámetros en el web.config

Agregar las siguientes líneas de configuración en el web.config:

Agregar las siguientes líneas de configuración en el web.config, reemplazando los textos entre paréntesis rectos por los propios del servicio a integrar:

[URL de procesamiento de la SignResponse]

[Service Provider Name]

[URL del DSS]

[Nombre de la Store para certificado de validación]

[Ubicación de la Store para certificado de validación]

[Subject del certificado de validación]

[Nombre de la Store para certificado de firma]

[Ubicación de la Store para certificado de firma]

[Subject del certificado de firma]

[Prefijos de certificados de documento]

[Thumbprint del certificado raíz de AGESIC]

A continuación, se describe cada parámetro.

-**ResponseConsumerUrl**: URL a la que se debe invocar para procesar el SignResponse proveniente del DSS.

-**ServiceProviderName**: Nombre del Service Provider configurado en la aplicación DSS.

-**DSSServiceUrl**: URL a donde se debe realizar el Post con el SignRequest.

-**DSSCertificate_StoreName**: Nombre de la Store donde se encuentra el certificado utilizado para validar la firma de la SignResponse.

-**DSSCertificate_StoreLocation**: Ubicación de la Store donde se encuentra el certificado utilizado para validar la firma de la SignResponse.

-**DSSCertificate_Subject**: Subject del certificado utilizado para validar la firma de la SignResponse.

-**SigningCertificate_StoreName**: Nombre de la Store donde se encuentra el certificado utilizado para firmar la SignRequest.

-**SigningCertificate_StoreLocation**: Ubicación de la Store donde se encuentra el certificado utilizado para firmar la SignRequest.

-**SigningCertificate_Subject**: Subject del certificado utilizado para firmar la SignRequest.

-**DocumentCertificatePrefixes**: Prefijos de los certificados de documento utilizados para validar la firma del documento contenido en la SignResponse.

-**ThumbprintRootAGESIC**: Thumbprint del certificado raiz de AGESIC utilizado para validar la firma del documento contenido en la SignResponse.

Clase y métodos de la librería

public static class DSSManager

public static XmlDocument GenerateSignRequestCAdES(byte[] documentToSign, int signRequestID)

-Es el método que genera el SignRequest de tipo XML. El mismo recibe como parámetro el array de bytes del documento de tipo texto plano a firmar y el id, numérico, de la sesión de firma. El retorno es el documento xml firmado listo para ser enviado al sitio del DSS.

public static XmlDocument GenerateSignRequestPAdES(byte[] documentToSign, int signRequestID)

-Es el método que genera el SignRequest de tipo XML. El mismo recibe como parámetro el array de bytes del documento de tipo pdf a firmar y el id, numérico, de la sesión de firma. El retorno es el documento xml firmado listo para ser enviado al sitio del DSS.

public static XmlDocument GenerateSignRequestXAdES(byte[] documentToSign, int signRequestID)

-Es el método que genera el SignRequest de tipo XML. El mismo recibe como parámetro el array de bytes del documento de tipo xml a firmar y el id, numérico, de la sesión de firma. El retorno es el documento xml firmado listo para ser enviado al sitio del DSS.

public static XmlDocument GenerateSignRequestCAdES_Hash(byte[] documentToSign, int signRequestID)

-Es el método que genera el SignRequest de tipo XML. El mismo recibe como parámetro el array de bytes del documento a firmar, al que se le aplicará una función hash, y el id, numérico, de la sesión de firma. El retorno es el documento xml firmado listo para ser enviado al sitio del DSS.

public static bool ValidateSignResponse(XmlDocument signResponse)

-Es el método que valida el SignResponse enviado desde el sitio del DSS. Recibe como parámetro un XML (SignResponse) y valida su firma, retornando true en caso que de que la misma sea válida.

public static string GetDSSServiceUrl()

-Retorna la URL del servicio del DSS.

Acceder al servicio y probar la integración

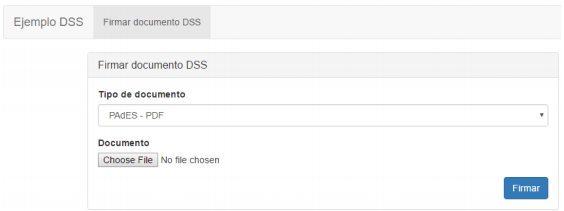


Figura 6. Servicio para archivos PDF

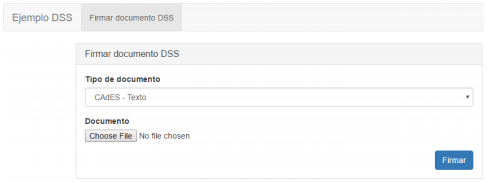


Figura 7. Servicio para archivos de texto plano (.txt)

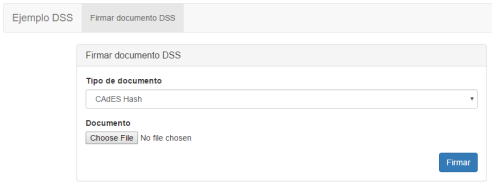


Figura 8. Servicio para cualquier tipo de archivos

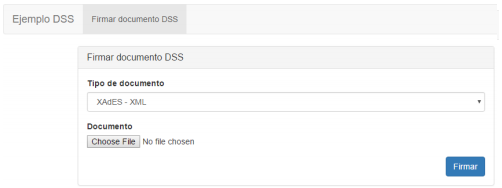


Figura 9. Servicio para archivos XML

Una vez realizado clic en el botón Firmar se redirige al sitio DSS.

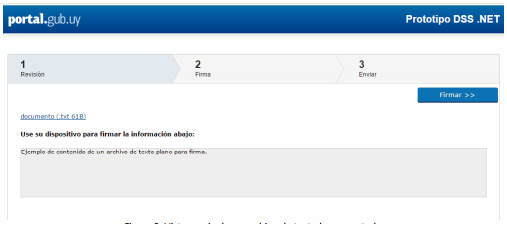


Figura 10. Vista previa de un archivo de text plano aceptado.

Figura 11. Vista previa de un archivo pdf aceptado.

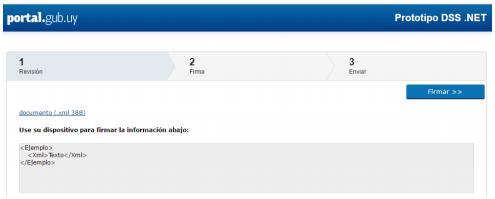


Figura 12. Vista previa de un archivo xml aceptado.

Figura 13. Vista previa de un hash aceptado.

Luego se debe proceder a firmar el documento siguiendo con el flujo establecido hasta obtener un mensaje de firma satisfactoria.

Figura 14. Firma digital.

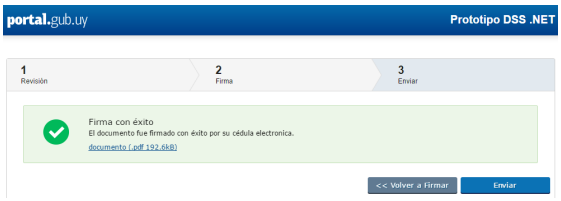


Figura 15. Firma exitosa.

Luego de darle click en el botón “Enviar” la propia aplicación DSS redirige al sitio web de ejemplo enviando un SignResponse de tipo XML.

Desde el sitio se obtiene el SignResponse se lo valida comprobando que el certificado utilizado por el DSS sea válido y que la firma incluida en el documento sea válida. Para esta última validación se corrobora que el certificado sea de la persona logeada (se compara número de documento y país) y se valida que el certificado utilizada sea emitido por una CA de Agesic, donde se recrea toda la cadena hasta la raíz. Si las distintas validaciones son satisfactorias se despliega en el sitio una pantalla de firma satisfactoria donde se incluye el SignResponse que fue validado.

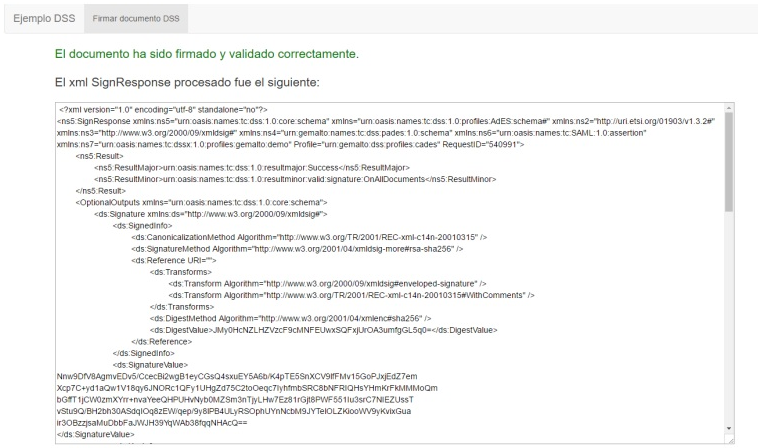


Figura 16. Firma exitosa vista desde el sitio