Sistemas Informáticos Abiertos, S.A.
Avenida de Europa, 2
Alcor Plaza Edificio B
Parque Oeste, Alcorcón 28922
Alcorcón - Madrid (España)
Telf.: (34) 902 480 580 Fax: (34) 91 307 79 80

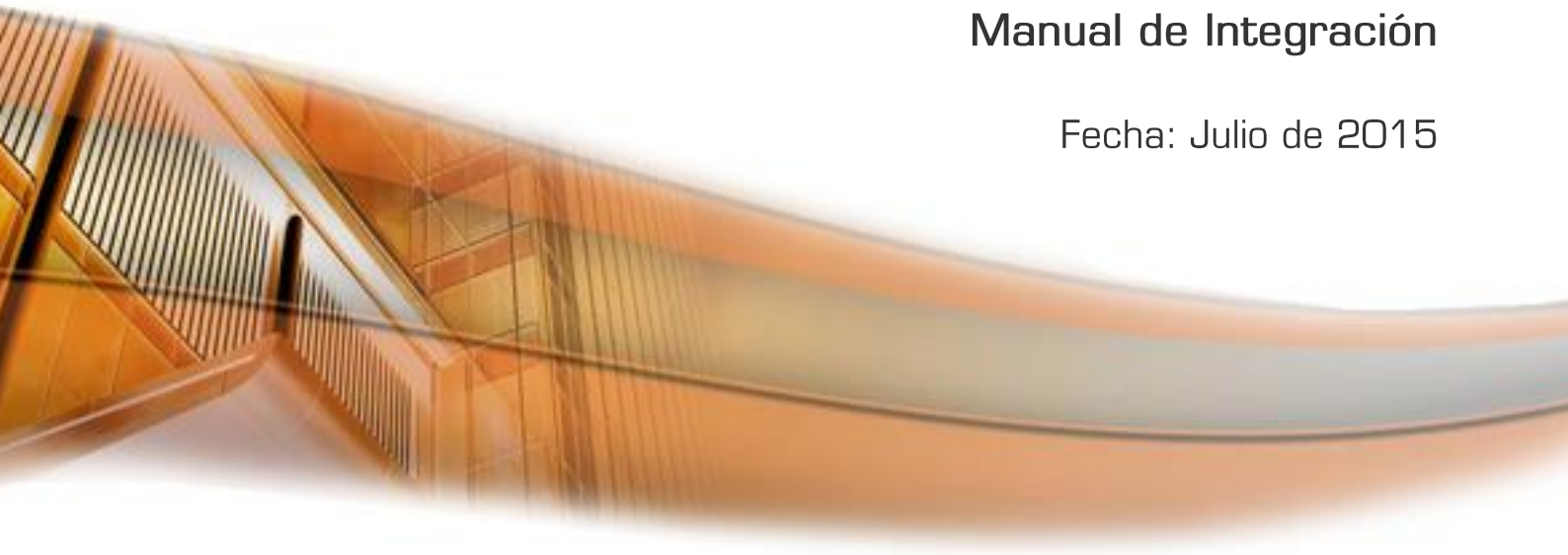
www.siainternational.com

SIAVAL Safecert v2.4.05

Pasarela de Firma

Manual de Integración

Fecha: Julio de 2015



ÍNDICE

RELACIÓN DE TABLAS

RELACIÓN DE ILUSTRACIONES

1. INTRODUCCIÓN

1.1 Contenido

En el presente documento se recoge la información relacionada con la integración del producto con aplicaciones del cliente.

Facilita una descripción de funcionalidades proporcionadas por el producto y de las tecnologías utilizadas para permitir que aplicaciones de terceros utilicen dichas funcionalidades.

1.2 Alcance del documento

La información contenida en el documento explica los mecanismos de integración con terceros que proporciona el producto **SIAVAL Safecert - Pasarela**, a través de los diferentes componentes de que consta.

1.3 Audiencia del documento

El presente documento está dirigido a todas aquellas personas que estén encargadas de implementar en sus propias aplicaciones la integración de las funcionalidades que proporciona el producto.

1.4 Documentación relacionada

Nombre del documento	Resumen
Manual de Instalación	Documento o conjunto de documentos que explican los requisitos y el procedimiento que es necesario llevar a cabo para realizar la instalación y despliegue de cada uno de los componentes del producto.

Tabla 1: Documentación relacionada

1.5 Temas tratados en el documento

Además del apartado introductorio actual, en el presente documento se tratan los siguientes temas:

- Descripción y aspectos generales del producto.
- API de Integración para el acceso al servicio de pasarela.
- Casos de uso.

2. DESCRIPCIÓN DEL PRODUCTO

2.1 Descripción general del producto

SIAVAL SafeCert es una solución de firma centralizada de la familia SIAVAL orientada a facilitar el uso/gestión de las claves y certificados de los usuarios finales, con las siguientes características generales:

- Permite a los usuarios finales la realización de firmas electrónicas de manera sencilla, sin que estos deban preocuparse de la gestión/mantenimiento de sus certificados.
- Los certificados y claves se mantienen seguros y controlados mediante el uso de hardware criptográfico (HSM).
- La gestión del ciclo de vida de los certificados se facilita puesto que las claves están centralizadas.
- Evita la dispersión o descontrol de las claves de los usuarios al no ser distribuidas y permanecer siempre bajo el control del HSM.
- Acceso controlado y auditado a las claves mediante varios niveles de seguridad (PIN, Segundo Factor de Autenticación, etc).
- Cliente CSP para Windows, que ofrece a las aplicaciones Windows ya existentes (que utilicen CAPI) la posibilidad de utilizar los certificados del usuario almacenados en SafeCert.
- Permite a los usuarios finales (titulares) importar certificados (con su clave privada) ya existentes de manera directa, a través del Cliente CSP para Windows.
- Integración desacoplada con terceros para delegar la generación/gestión/validación del segundo factor de autenticación (SFDA) que puede proteger el acceso a los certificados.
- Posibilidad de uso de IdentityGuard como segundo factor de autenticación, bien mediante el uso de OTPs (One Time Password) enviado por SMS, bien mediante claves de un solo uso generadas en Token físico o software o, por último, utilizando Tarjeta de Coordenadas.
- Ofrece una consola web para la administración y gestión centralizada del producto, con distintos niveles de acceso mediante perfilado configurable. Mediante la consola puede realizarse la configuración de repositorios HSMs, conectores de segundo factor de autenticación, sistemas de segundo factor de autenticación, gestión de titulares, asignación y parametrización de segundos factores de autenticación para el uso del certificado de cada titular, etc.

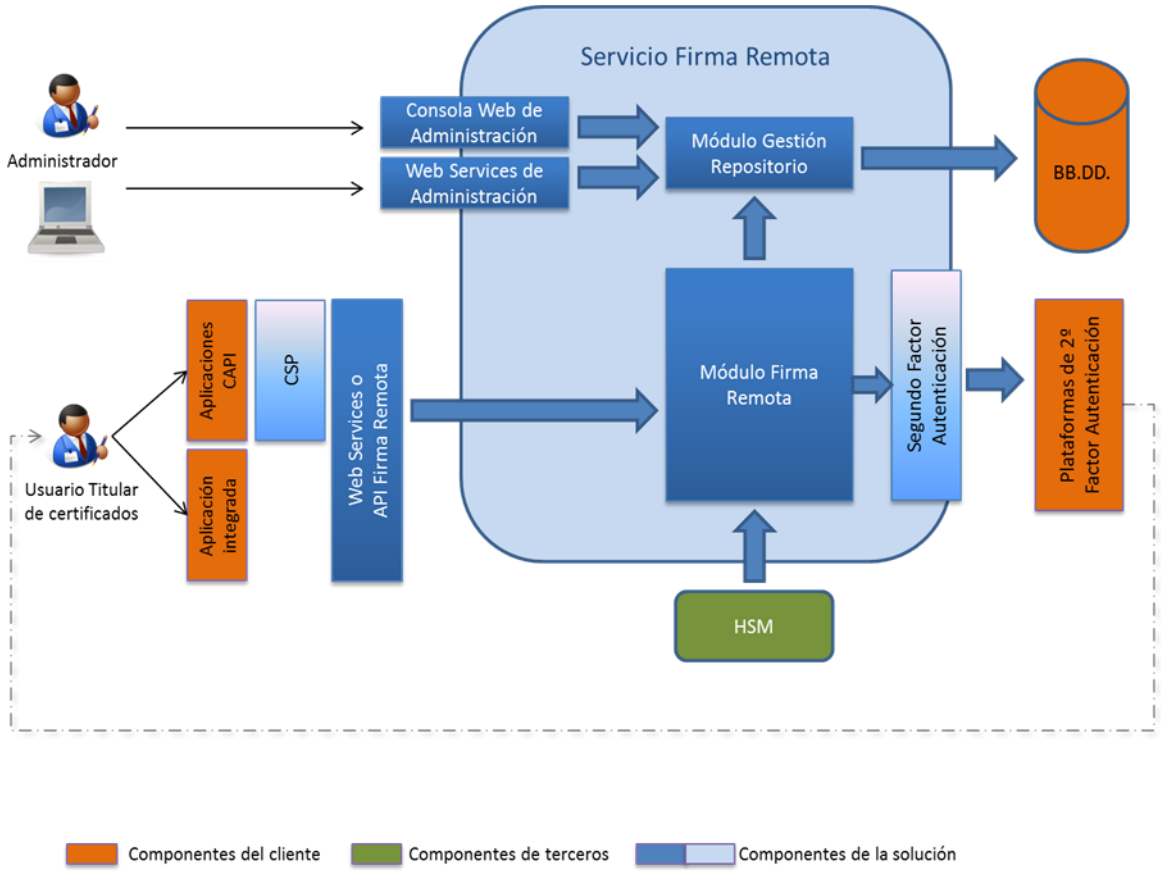


Ilustración 1: Arquitectura Lógica SIAVAL SafeCert

La solución completa SIAVAL SafeCert se compone de:

- Módulo de firma. Software servidor que proporciona los servicios de firma, importación de claves mediante PKCS#12 y cambio de PIN de certificados a través de Servicios Web convencionales y mediante el uso del API de Integración Java.
- Módulo de administración. Consola web y Servicios Web que en su conjunto permiten la gestión de la infraestructura, conectores y sistemas de segundo factor de autenticación, alta/baja de certificados/titulares, activación/desactivación temporal de certificados de titular, asignación de segundo factor de autenticación a certificados, etc.
- Repositorio de claves HSM. Hardware criptográfico encargado de la protección segura de las claves, que realiza las operaciones criptográficas de bajo nivel.
- BBDD para el almacenamiento de la configuración de funcionamiento del sistema, trazas de auditoría/operaciones, etc.
- API Java para la explotación de los servicios de firma (también disponibles como Servicios Web convencionales accesibles de manera directa).
- Cliente CSP de Windows que permite el uso transparente de los certificados del titular por parte de aplicaciones de terceros que utilicen CAPI.

- Pasarela de Firma que permite, en sistemas web, asegurar la entrega de credenciales de firma por parte de los usuarios en un entorno seguro, independiente de las aplicaciones que se utilicen para firmar documentos.

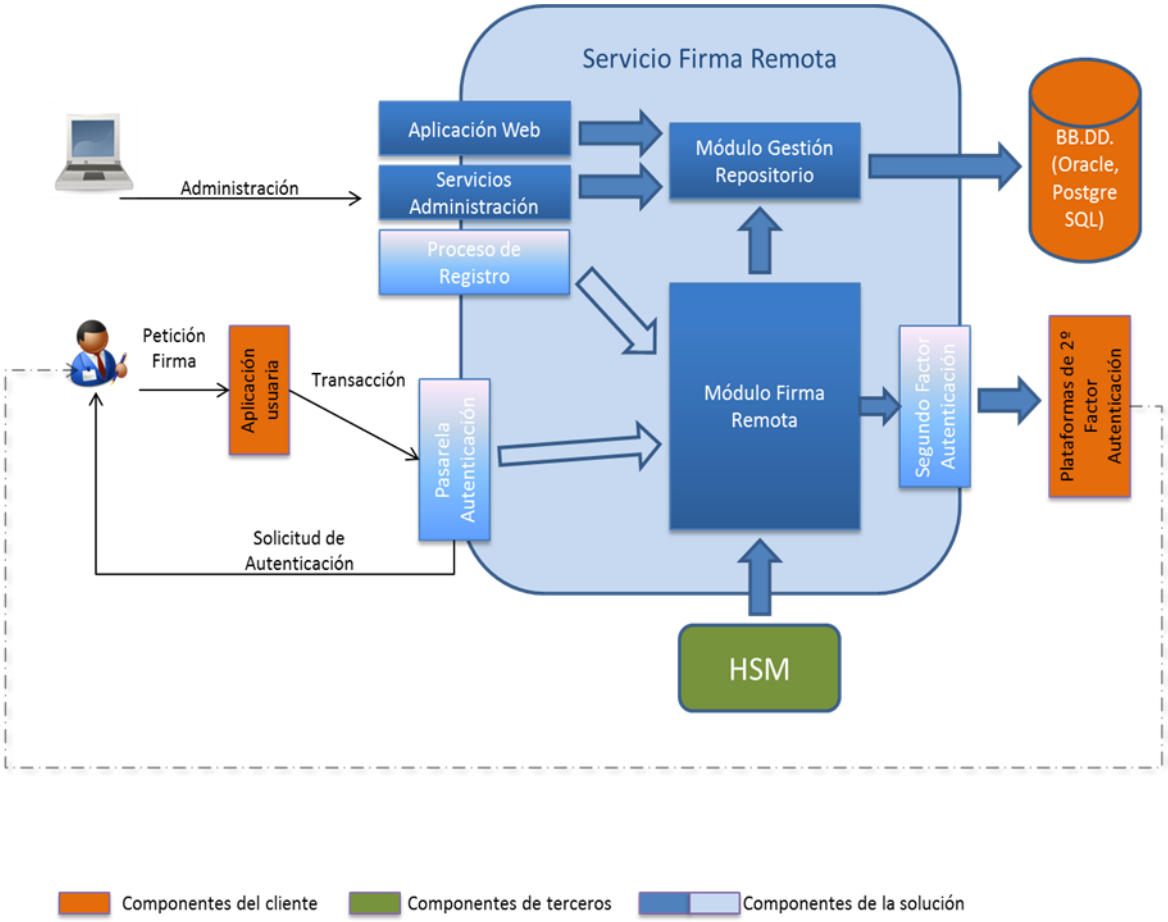


Ilustración 2: Arquitectura Lógica SIAVAL SafeCert-Pasarela

2.2 Descripción pasarela de firma del producto

SIAVAL/SafeCert - Pasarela es un módulo adicional de la familia SIAVAL/SafeCert que se encarga de conectar a distintas aplicaciones de firma con el sistema de firma centralizada SIAVAL/SafeCert, de tal forma que éstas interactúan de manera segura iniciando una transacción de firma, autenticación o emisión/renovación de certificados y recuperando los datos resultantes de la operación.

Características generales:

- Realiza el proceso de firma, autenticación o emisión/renovación de certificados de firma y autenticación.
- Obtiene los certificados asociados a un titular.
- El proceso de petición de datos de autenticación es transparente para las aplicaciones, *SIAVAL/SafeCert - Pasarela* se encarga de solicitar las claves y/o segundo factor de autenticación necesarios en función del certificado utilizado para la firma/autenticación.
- *SIAVAL/SafeCert – Pasarela* permite para la operación de firma, cachear opcionalmente el PIN del certificado utilizado, asociándolo a la sesión del usuario durante un tiempo determinado, no siendo necesario introducirlo de nuevo.
- *SIAVAL/SafeCert-Pasarela* permite iniciar una transacción para la emisión o renovación de certificados de firma o autenticación.

Arquitectura Lógica

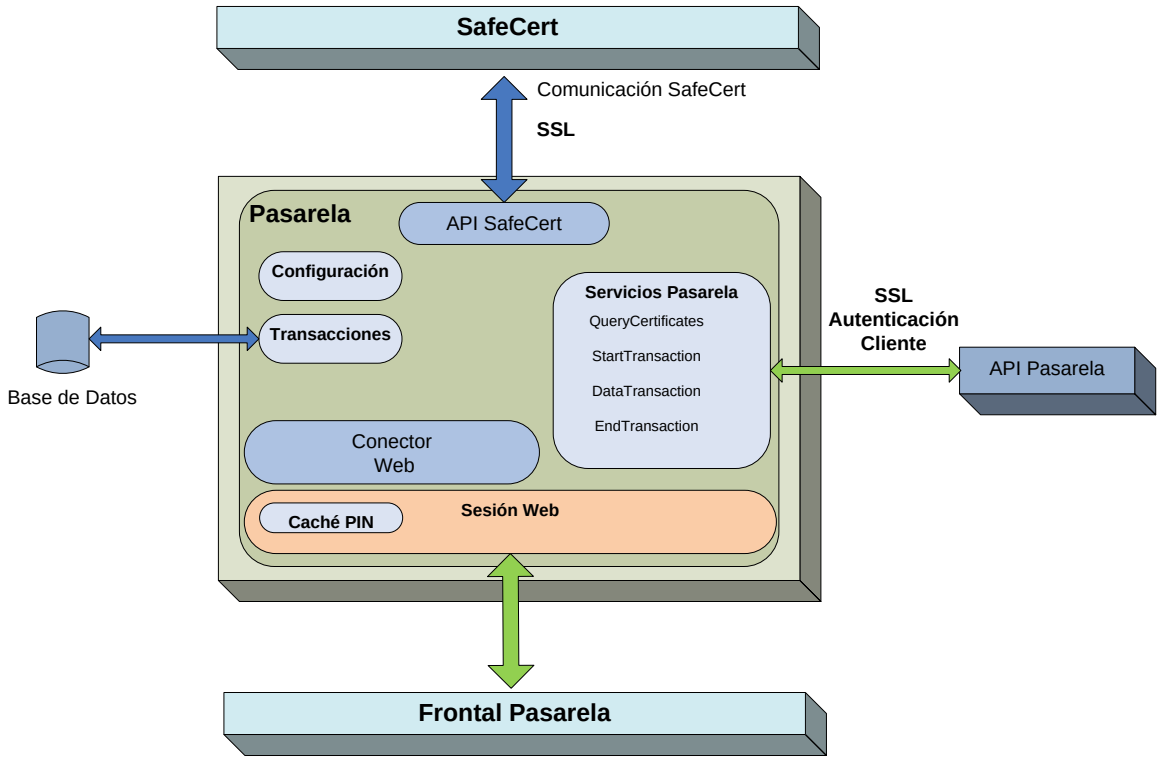


Ilustración 3: Arquitectura Lógica SIAVAL/SafeCert Pasarela

La solución completa SIAVAL/SafeCert Pasarela se compone de:

- Módulo de firma. Software servidor que proporciona los servicios de firma/autenticación y recuperación de certificados, mediante el uso del API de Integración Java.
- API de integración de Pasarela.

3. CONCEPTOS GENERALES

SIAVAL/SafeCert Pasarela dispone de un API de integración en Java que accede a los servicios de operaciones mediante interfaz específica (Hessian), de manera que se facilita la integración con las aplicaciones cliente para realizar la firma/autenticación remota a través de este API.

4. API DE INTEGRACIÓN

Como anteriormente se ha descrito, se dispone de un API de integración Java a través del cual se interactúa con el producto.

4.1 Configuración del API de Integración

El API de integración de la pasarela utiliza un fichero de configuración *gatewayapi.properties,* que, hasta la versión 2.4.00 debe estar disponible para el API como recurso en el classpath.

Hasta la versión 2.4.00 el API de Integración de la Pasarela comparte contexto entre sus instancias, por lo que si se utiliza por varias aplicaciones en un mismo servidor de aplicaciones, el API se deberá cargar en un classloader independiente para cada una de ellas.

Desde la versión 2.4.01 cada instancia del API tiene su propio fichero de configuración independiente, por lo que pueden convivir distintas instancias del API bajo un mismo entorno de ejecución, cada una con su configuración.

Propiedades configurables en el fichero:

URL_GATEWAY: URL donde se encuentran desplegados los servicios Hessian a los que accede el API.

```
URL_GATEWAY=https://servidor/rss-gateway/HESS/OperationGateWayRSS
```

AUTH_STORE: ruta al almacén de claves que servirán para autenticar a la aplicación.

```
AUTH_STORE=/opt/pasarela/aplicacion.pfx
```

AUTH_STORE_PASS: Contraseña de acceso al almacén de claves de autenticación. Esta contraseña estará cifrada y será proporcionada junto con el almacén de claves que se utilizará para la autenticación con los servicios de la pasarela.

```
AUTH_STORE_PASS=secret
```

SOCKET_TIMEOUT: (desde 2.4.05). Valor, en milisegundos, para el timeout de conexión utilizado por el API de Pasarela. Su valor por defecto son 30.000 ms (30 segundos)

```
SOCKET_TIMEOUT=20000
```

SSL_PROTOCOL: (desde 2.4.05). Protocolo utilizado para realizar la conexión SSL. Por defecto se utilizará TLSv1.2.

En la siguiente URL se puede ver una lista completa de los valores que puede tomar SSL_PROTOCOL.
http://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html#SSLContext

```
SSL_PROTOCOL=TLSv1.2
```

LOAD_BC_PROVIDER: (desde 2.4.05-20150702-1327)

Valores: **true | false**

```
LOAD_BC_PROVIDER=true
```

Valor por defecto si no se especifica el atributo, **true.**

Indica si se utilizará o no explícitamente el proveedor de BouncyCastle para la carga del almacén PKCS#12 utilizado en la autenticación mediante certificados a través de la conexión SSL/TLS.

Si se especifica el valor **true,** se utilizará el proveedor de BouncyCastle para la carga del almacén PKCS#12, por tanto deberán incluirse las librerías necesarias de BouncyCastle en el classpath de ejecución del API, de no estar disponible dichas librerías, fallará la carga del *provider* y se **devolverá error en la ejecución de la operación**.

Si se especifica el valor **false**, no se cargará explícitamente el proveedor de BouncyCastle y se utilizará el proveedor que tenga configurado Java por defecto.

4.2 Autenticación de una aplicación

Cada aplicación que se integre con el sistema de Pasarela deberá tener asociado un certificado a través del cual se realice la autenticación de la propia aplicación. Este certificado deberá estar en concordancia con el certificado utilizado para configurar el acceso mediante SSL con autenticación de cliente. Por lo tanto, este certificado deberá ser proporcionado por los administradores de la infraestructura del sistema.

4.3 Operaciones del API de integración

El API de Integración de la Pasarela se utiliza a través de la clase ***GateWayAPI*** proporcionada en el *gateway-api-v.xxx.jar,* y provee las funciones que se describen a continuación.

A través del API de integración de la pasarela se podrán ejecutar las siguientes operaciones:

- **queryCertificates:** Esta operación devuelve una lista de los certificados operativos que un titular dispone en SafeCert. No se devuelven los certificados bloqueados, desactivados, etc.
- **queryCertificatesFiltered:** Esta operación devuelve una lista de los certificados operativos filtrados por el tipo de operación (actualmente firma, autenticación o ambas) que un titular dispone en SafeCert. No se devuelven los certificados bloqueados, desactivados, etc.
- **startTransaction:** Inicio de una transacción de firma, se envía toda la información necesaria para realizar la firma.
- **startOpTransaction:** Inicio de una transacción cuya operación se especifica como parámetro.
- **dataTransaction:** Recupera los datos de una transacción de firma una vez finalizada la transacción.

- **startAuthTransaction:** Inicio de una transacción de autenticación en función de la información asociada al certificado en SafeCert. En esta transacción de autenticación no se indica el nivel, por lo que se solicitará PIN y SFDA si así lo tiene asociado el certificado utilizado. Esta operación desaparecerá; en su lugar debe utilizarse startAuthByLevelTransaction.
- **startAuthByLevelTransaction:** Inicio de una transacción para autenticación indicando el tipo de autenticación que se requiera. La aplicación decide qué datos se piden al usuario para realizar la autenticación.
- **dataAuthByLevelTransaction:** Recupera los datos de una transacción de autenticación por tipo una vez finalizada la transacción.
- **endTransaction:** Una vez finalizada la operación de una determinada transacción, esta es eliminada del sistema.
- **listOwnerCertificates:** recupera la lista completa de certificados de un titular. Se diferencia de queryCertificates porque esta operación sí recupera aquellos certificados que pudieran encontrarse en estado no activo, así como información relacionada con el certificado.

Para ver más detalles acceder al JavaDoc disponible en la distribución del producto.

4.3.1 Operación queryCertificates

```
public QueryCertificatesResult queryCertificates(String owner)
        throws SafeCertGateWayException
```

Esta función devuelve una lista de los certificados disponibles para un usuario concreto, para ello se le pasa el parámetro *owner,* que se corresponderá con el identificador del titular de los certificados en SafeCert. Solamente se devuelven los certificados que el titular puede utilizar en este momento. Los certificados no activos, bloqueados, etc no se incluirán en la lista.

Parámetros		
Parámetro	**Tipo**	**Descripción**
*[1]owner	String	Identificador del usuario propietario de los certificados en SafeCert.

Tabla 2: Parámetros operación queryCertificates

[1] Obligatorio

Respuesta	
Tipo	**Descripción**
QueryCertificatesResult	Objeto resultado de la función. *owner*: usuario propietario de los certificados. *certificates*: lista de certificados del usuario que están disponibles para su uso en este momento.

Tabla 3: Respuesta operación queryCertificates

En caso de que el usuario no disponga de ningún certificado disponible, se devuelve una lista vacía. Los certificados bloqueados, desactivados, pendientes, etc, no se incluyen en esta lista.

Exception	
Tipo	**Descripción**
SafeCertGateWayException	Excepción que contiene un código y una descripción del error producido.

Tabla 4: Excepción operación queryCertificates

4.3.2 Operación queryCertificatesFiltered

```
public QueryCertificatesResult queryCertificatesFiltered(String owner, int
filterOperation)
            throws SafeCertGateWayException
```

Esta función devuelve una lista de los certificados disponibles para un usuario concreto filtrados por el tipo de operación indicada como parámetro, para ello se le pasa el parámetro *owner,* que se corresponderá con el identificador del titular de los certificados en SafeCert y el filtro requerido *filterOperation*.

Parámetros

Parámetro	Tipo	Descripción
*[2]owner	String	Identificador del usuario propietario de los certificados en SafeCert.
filterOperation	int	filtro de certificados por tipo de operación. Los posibles valores pueden ser: ConstantsGateWay.OPERATION_ALL ConstantsGateWay.OPERATION_AUTHENTICATION ConstantsGateWay.OPERATION_SIGN

Tabla 5: Parámetros operación queryCertificatesFiltered

Respuesta	
Tipo	**Descripción**
QueryCertificatesResult	Objeto resultado de la función. *owner*: usuario propietario de los certificados. *certificates*: lista de certificados del usuario que están disponibles para su uso en este momento.

Tabla 6: Respuesta operación queryCertificatesFiltered

En caso de que el usuario no disponga de ningún certificado disponible, se devuelve una lista vacía. Los certificados bloqueados, desactivados, pendientes, etc, no se incluyen en esta lista.

Exception

[2] Obligatorio

Tipo	Descripción
SafeCertGateWayException	Excepción que contiene un código y una descripción del error producido.

Tabla 7: Excepción operación queryCertificatesFiltered

4.3.3 Operación startTransaction

```
public StartTransactionResult startTransaction(
                String owner,
                DataToSign datatosign,
                ParameterAux[] parametersAux)
        throws SafeCertGateWayException
```

Esta función inicia una transacción de firma asociada a una aplicación y a un titular (*owner*).

Parámetros de entrada a la función:

Parámetros		
Parámetro	**Tipo**	**Descripción**
*owner	String	Identificador del usuario que va a realizar la firma. Desde la versión 2.4.01 se puede iniciar una transacción de firma sin indicar el id del titular. La pasarela se encargará de autenticar al titular para obtener su identificador antes de comenzar la operación de firma.
*datatosign	DataToSign	Información necesaria para realizar el proceso de firma: *language*: idioma que se utilizará en la consola web. *description*: descripción de la transacción. *certificate*: certificado que se va a utilizar para la firma. **documents*: lista de hashes a firmar. **digestAlgorithm*: algoritmo del hash utilizado para generar el hash. *forcePIN*: obligatoriedad de introducir el PIN.

		forceSFDA: obligatoriedad de introducir el SFDA. **redirectOK*: URL de redirección cuando fin correcto. **redirectError*: URL de error cuando fin erróneo.
parametersAux	ParameterAux[]	Sin implementar en esta versión. Para futuros usos.

Tabla 8: Parámetros operación startTransaction

Como respuesta se obtiene un objeto de tipo *StartTransactionResult* que contiene el identificador único de la transacción y la URL de redirección.

Respuesta	
Tipo	**Descripción**
StartTransactionResult	Contiene el identificador único de la transacción. Este identificador será necesario para invocar al resto de funciones del API. *idTransaccion*: identificador único de la transacción. *redirect*: URL de redirección a la consola web de la pasarela para realizar el proceso de firma. La aplicación integrada deberá redirigir al usuario hacia la URL indicara para comenzar el proceso de firma.

Tabla 9: Respuesta operación startTransaction

Exception	
Tipo	**Descripción**
SafeCertGateWayException	Excepción que contiene un código y una descripción del error producido.

Tabla 10: Excepción operación startTransaction

4.3.4 Operación startOpTransaction

Esta función inicia una transacción del tipo indicado asociado a una aplicación y a un titular (*owner*).

```
public StartOpTransactionResult startOpTransaction(
                String owner,
                StartOperationInfo operationInfo,
                ParameterAux[] parametersAux)
```

```
throws SafeCertGateWayException
```

Parámetros de entrada a la función:

Parámetros		
Parámetro	**Tipo**	**Descripción**
*owner	String	Identificador del usuario que va a realizar la firma.
* operationInfo	StartOperationInfo	Información necesaria para realizar el proceso de firma: **operationName*: tipo de operación a realizar durante la transacción. Se permiten las siguientes operaciones: - ISSUE_CERTIFICATE *language*: idioma que se utilizará en la consola web. *description*: descripción de la transacción. **redirectOK*: URL de redirección cuando fin correcto. **redirectError*: URL de error cuando fin erróneo.
*parametersAux	ParameterAux[]	Para la operación ISSUE_CERTIFICATE se puede indicar los siguientes valores del tipo clave-valor: - Clave: *CERTIFICATE_TYPE (tipo de certificado a emitir). - Valor: *SIGN (certificado de firma).

Tabla 11: Parámetros operación startOpTransaction

Como respuesta se obtiene un objeto de tipo *StartOpTransactionResult* que contiene el identificador único de la transacción y la URL de redirección.

Respuesta	
Tipo	**Descripción**
StartOpTransactionResult	Contiene el identificador único de la transacción. Este identificador será necesario para invocar al resto de funciones del API.

	idTransaccion: identificador único de la transacción. *redirect*: URL de redirección a la consola web de la pasarela para realizar el proceso de firma. La aplicación integrada deberá redirigir al usuario hacia la URL indicara para comenzar el proceso de firma.

Tabla 12: Respuesta operación startOpTransaction

Exception	
Tipo	**Descripción**
SafeCertGateWayException	Excepción que contiene un código y una descripción del error producido.

Tabla 13: Excepción operación startOpTransaction

4.3.5 Operación startAuthTransaction (obsoleta)

```
public StartAuthTransactionResult startAuthTransaction(
                String owner,
                DataToSign datatosign,
                ParameterAux[] parametersAux)
        throws SafeCertGateWayException
```

Esta función desaparecerá; se debe utilizar en su lugar startAuthByLevelTransaction. Esta función inicia una transacción de autenticación asociada a una aplicación y a un titular (*owner*). En esta transacción de autenticación no se indica el nivel, por lo que se solicitará PIN y SFDA si así lo tiene asociado el certificado utilizado.

Parámetros de entrada a la función:

Parámetros		
Parámetro	**Tipo**	**Descripción**
*owner	String	Identificador del usuario que va a realizar la autenticación.
*datatosign	DataToSign	Información necesaria para realizar el proceso: *language*: idioma que se utilizará en la consola web. *description*: descripción de la transacción. *certificate*: certificado que se va a utilizar para la

		autenticación. **documents*: hashes de los datos a firmar. Sin uso para esta operación. Estos hashes se ignorarán en el proceso de autenticación. **digestAlgorithm*: algoritmo del hash. Sin uso para esta operación *forcePIN*: obligatoriedad de introducir el PIN. *forceSFDA*: obligatoriedad de introducir el SFDA. **redirectOK*: URL de redirección cuando fin correcto. **redirectError*: URL de error cuando fin erróneo.
parametersAux	ParameterAux[]	Sin implementar en esta versión. Para futuros usos.

Tabla 14: Parámetros operación startAuthTransaction

Como respuesta se obtiene un objeto de tipo *StartAuthTransactionResult* que contiene el identificador único de la transacción y la URL de redirección.

Respuesta	
Tipo	**Descripción**
StartAuthTransactionResult	Contiene el identificador único de la transacción. Este identificador será necesario para invocar al resto de funciones del API. *idTransaccion*: identificador único de la transacción. *redirect*: URL de redirección a la consola web de la pasarela para realizar el proceso de autenticación.

Tabla 15: Respuesta operación startAuthTransaction

Exception	
Tipo	**Descripción**

SafeCertGateWayException	Excepción que contiene un código y una descripción del error producido.

Tabla 16: Excepción operación startAuthTransaction

4.3.6 Operación startAuthByLevelTransaction

```
public StartAuthTransactionResult startAuthByLevelTransaction(
                String owner,
                DataToAuth datatoauth,
                ParameterAux[] parametersAux)
        throws SafeCertGateWayException
```

Esta función inicia una transacción de autenticación especificando el nivel de autenticación a realizar. Es la aplicación la que decide si para autenticar a un usuario a través de la Pasarela se le solicitará solamente el PIN (contraseña), solamente el segundo factor de autenticación (SFDA), o ambos.

Únicamente se podrá usar la autenticación con SFDA (o PIN+SFDA) si el certificado del usuario tiene asociado un segundo factor de autenticación.

Es posible iniciar la transacción sin indicar el identificador del titular a autenticar. En ese caso, será la interfaz de la pasarela la que solicite el dato al usuario. No se permite iniciar una transacción de autenticación con nivel de solo SFDA sin indicar el identificador del titular.

La autenticación utilizando solo SFDA se debe utilizar de manera muy controlada, puesto que puede dar lugar a que usuarios malintencionados puedan utilizar sistemas de envío de SMS de manera descontrolada. La autenticación utilizando solamente SFDA se recomienda utilizar solamente como una subida de nivel en la autenticación, conociendo de antemano cual es el usuario previamente autenticado.

Parámetros de entrada a la función:

Parámetros		
Parámetro	**Tipo**	**Descripción**
owner	String	Usuario que va a realizar la operación. Si este dato es null o cadena vacía, será la consola de la pasarela quien solicite el dato al usuario.
*authLevel	int	Nivel de autenticación, posibles valores: **`ConstantsGateWay.AUTH_PIN`:** Mediante solo contraseña. **`ConstantsGateWay.AUTH_OTP`:** Mediante solo segundo factor

		de autenticación (no puede utilizarse sin indicar el identificador del titular en el parámetro owner). **`ConstantsGateWay.AUTH_PIN_OTP`:** Mediante contraseña y segundo factor de autenticación.
*datatoauth	DataToAuth	Información necesaria para realizar el proceso de autenticación: *language*: idioma que se utilizará en la consola web. *description*: descripción de la transacción. *certificate*: certificado que se va a utilizar para la transacción. **redirectOK*: URL de redirección cuando fin correcto. **redirectError*: URL de error cuando fin erróneo.
parametersAux	ParameterAux[]	Sin implementar en esta versión. Para futuros usos.

Tabla 17: Parámetros operación startAuthByLevelTransaction

Como respuesta se obtiene un objeto de tipo *StartAuthTransactionResult* que contiene el identificador único de la transacción y la URL de redirección.

Respuesta	
Tipo	**Descripción**
StartAuthTransactionResult	Contiene el identificador único de la transacción. Este identificador será necesario para invocar al resto de funciones del API. *idTransaccion*: identificador único de la transacción. *redirect*: URL de redirección a la consola web de la pasarela para realizar el proceso de autenticación.

Tabla 18: Respuesta operación startAuthByLevelTransaction

Exception	
Tipo	**Descripción**

SafeCertGateWayException	Excepción que contiene un código y una descripción del error producido.

Tabla 19: Excepción operación startAuthByLevelTransaction

4.3.7 Operación dataTransaction

```
public DataTransactionResult dataTransaction(String id_transaction)
        throws SafeCertGateWayException
```

Esta función recupera la información de una transacción una vez que se ha realizado una operación de firma.

En caso de invocar a esta función antes de que el usuario realice la operación, se retornará un error.

Parámetros de entrada a la función:

Parámetros		
Parámetro	**Tipo**	**Descripción**
*id_transaccion	String	Identificador único de la transacción.

Tabla 20: Parámetros operación dataTransaction

Como respuesta se obtiene un objeto de tipo *DataTransactionResult*:

Respuesta	
Tipo	**Descripción**
DataTransactionResult	Contiene información de la firma. Incluye los documentos firmados *owner*: usuario que ha realizado la operación. *StateTransaction*: datos del estado de la transacción. *certificate*: certificado utilizado en la operación *signs*: lista de documentos firmados.

Tabla 21: Respuesta operación dataTransaction

Exception	
Tipo	**Descripción**
SafeCertGateWayException	Excepción que contiene un código y una descripción del error producido.

Tabla 22: Excepción operación dataTransaction

4.3.8 Operación dataAuthByLevelTransaction

```
public DataAuthByLevelTransactionResult dataAuthByLevelTransaction(String id_transaction)
        throws SafeCertGateWayException
```

Esta función recupera la información de una operación de autenticación por nivel una vez finalizada la transacción.

En caso de invocar a esta función antes de que el usuario realice la operación, se retornará un error.

Parámetros de entrada a la función:

Parámetros		
Parámetro	**Tipo**	**Descripción**
*id_transaccion	String	Identificador único de la transacción.

Tabla 23: Parámetros operación dataTransaction

Como respuesta se obtiene un objeto de tipo *DataAuthByLevelTransactionResult*:

Respuesta	
Tipo	**Descripción**
DataAuthByLevelTransactionResult	Contiene información sobre el resultado de la operación de

	autenticación por nivel. *owner*: identificador correspondiente al titular que se ha intentado autenticar. Podría estar vacío si la aplicación no indicó un ID de titular y el usuario no ha introducido ninguno. *StateTransaction*: estructura para conocer el resultado de la operación. *certificate*: certificado con el que se ha realizado la operación cuando ésta ha concluido correctamente. *OwnerInfoTransaction:* estructura con la información del titular que ha realizado la operación cuando esta ha concluido correctamente.

Tabla 24: Respuesta operación dataAuthByLevelTransaction

Exception	
Tipo	**Descripción**
SafeCertGateWayException	Excepción que contiene un código y una descripción del error producido.

Tabla 25: Excepción operación dataAuthByLevelTransaction

StateTransaction	
Tipo	**Descripción**
StateTransaction	Contiene la información necesaria para conocer el resultado de una transacción finalizada. *state: estado de la transacción,* en este caso siempre estará en estado finalizada. *valor "1"* result: resultado de la operación, "OK" o "ERROR" code_error: código del error en caso de que result=="ERROR" description: descripción del error en caso de que result=="ERROR"

Tabla 26: StateTransaction

OwnerInfoTransaction	
Tipo	**Descripción**
OwnerInfoTransaction	Contiene la información personal del titular autenticado. idOwner: Identificador del titular ou: Unidad Organizativa en el sistema SafeCert email: dirección de correo electrónico datoPersonal1: Dato Personal , suele coincidir con el Nombre del titular datoPersonal2: Dato Personal , suele coincidir con el Primer apellido del titular datoPersonal3: Dato Personal , suele coincidir con el Segundo apellido del titular datoPersonal4: Dato Personal , suele coincidir con el NIF del titular datoPersonal5: Dato Personal , suele coincidir con el teléfono del titular datoPersonal6: Dato Personal , dato adicional del sistema datoPersonal7: Dato Personal , dato adicional del sistema

Tabla 27: OwnerInfoTransaction

4.3.9 Operación endTransaction

```
public EndTransactionResult endTransaction(String id_transaction)
throws SafeCertGateWayException
```

Esta función elimina una transacción una vez que ha terminado el proceso de la misma. **Se debe invocar esta función tras terminar la operación, para un correcto mantenimiento de las transacciones**.

En caso de invocar a esta función antes de que el usuario finalice la operación, se retornará un error.

Parámetros de entrada a la función:

Parámetros		
Parámetro	**Tipo**	**Descripción**
*id_transaccion	String	Identificador único de la transacción.

Tabla 28: Parámetros operación endTransaction

Como respuesta se obtiene un objeto de tipo End*TransactionResult*:

Respuesta	
Tipo	**Descripción**
EndTransactionResult	Contiene el código y descripción del resultado de la operación de eliminación. *result*: código del resultado (GateWayTransactionCtes.STATE_TRANSACTION_CLOSE en caso correcto). *description*: descripción del resultado.

Tabla 29: Respuesta operación endTransaction

Exception	
Tipo	**Descripción**
SafeCertGateWayException	Excepción que contiene un código y una descripción del error producido.

Tabla 30: Excepción operación endTransaction

4.3.10 Operación ListOwnerCertificates

```
public RSSListOwnerCertificatesResult listOwnerCertificates(String owner) throws
SafeCertGateWayException
```

Esta operación permite recuperar el listado de certificados de un titular. La diferencia con respecto al método QueryCertificates radica en el hecho de que la operación ListOwnerCertificates recupera todos los certificados de un titular, incluyendo aquellos que no se encuentra en estado activo. Junto con cada uno de los certificados se incluye información adicional relativa al certificado.

Parámetros		
Parámetro	**Tipo**	**Descripción**
owner	String	Identificador del usuario propietario de los certificados en

		SafeCert.

Tabla 31: Parámetros operación ListOwnerCertificates

Como respuesta se obtiene un objeto de tipo RSSListOwnerCertificatesResult

Respuesta	
Tipo	**Descripción**
RSSListOwnerCertificatesResult	Contiene la información de los certificados del titular. *id_owner*: identificador del titular en SafeCert. *certificates*: lista de certificados del titular junto con la información.

Tabla 32: Respuesta operación ListOwnerCertificates

4.4 Ejemplo de USO DEL API

4.4.1 Operación completa de firma

Para generar una nueva transacción de firma, lo primero que hacemos es obtener la lista de certificados del usuario. De esta lista seleccionaremos el certificado que se va a utilizar en el proceso de firma, que será el que se envíe a la transacción.

```
QueryCertificatesResult lCertificados = new GateWayAPI().queryCertificates(titular);
```

Una vez que tenemos identificado el certificado, se añade éste a la información de la transacción:

```
DataToSign datatosign = new DataToSign();

datatosign.setCertificate(certificado);
```

Existe la posibilidad de no indicarle el certificado de firma a la transacción, sólo válida en el caso de que el usuario disponga de un único certificado en SafeCert tras aplicar el filtro de certificados para operaciones de firma. En este caso el sistema de Pasarela obtendrá ese certificado de SafeCert para realizar la firma. Si el usuario tuviese más de un certificado válido se retorna un error.

Se indican las URLs de retorno para los casos de fin OK y fin Error:

```
datatosign.setRedirectError("http://servidor/aplicacion/paginadeerror.html");

datatosign.setRedirectOK("http://servidor/aplicacion/paginadeok.html");
```

Se calculan los hashes que se desea enviar a firmar y se genera la lista de los mismos:

```
datatosign.setDigestAlgorithm("SHA1");
List<DocumentsToSign> _documents = new ArrayList<DocumentsToSign>();
DocumentsToSign doc = new DocumentsToSign();
```

Los hashes se pueden enviar codificados en Base64 o sin codificar, indicándolo en la información que se envía a la transacción:

```
doc.setEncodeB64(false/true);
doc.setData(hashDocumento);

_documents.add(doc);
```

Iniciamos la transacción:

```
StartTransactionResult result = new GateWayAPI().startTransaction(usuarioSafecert,
datatosign, null);
```

La operación nos devuelve el identificador único de la transacción, con el que realizaremos las siguientes peticiones:

```
String id_transaction = result.getIdTransaction();
```

También nos devuelve la URL de redirección hacia la consola web de la Pasarela, a la que se debe redirigir al usuario para que éste realice la firma:

```
String URLRedireccion = result.getRedirect();
```

El usuario realiza la firma de los documentos en la consola web de la Pasarela y se le redirige a la URL correspondiente según el resultado de la operación (OK o Error).

Una vez que el usuario ha finalizado la firma, la aplicación puede obtener el resultado del proceso y recoger los hashes firmados si la operación ha resultado correcta:

```
DataTransactionResult resultado = new GateWayAPI().dataTransaction(id_transaction);
```

Para dar por terminada la transacción, se finaliza:

```
EndTransactionResult result = new GateWayAPI().endTransaction(id_transaction);
```

4.4.2 Operación completa de emisión de certificado de firma

Para comenzar una transacción de emisión de certificado de firma, es necesario realizar una llamada al método startOpTransaction. En primer lugar se añade la información para la transacción:

```
StartOperationInfo opInfo = new StartOperationInfo();
```

Se indican las URLs de retorno para los casos de fin OK y fin Error:

```
opInfo.setRedirectError("http://servidor/aplicacion/paginadeerror.html");

opInfo.setRedirectOK("http://servidor/aplicacion/paginadeok.html");
```

Indicamos que la operación que vamos a comenzar es la operación de emisión/renovación del certificado:

```
opInfo.setOperationName(StartOpTransactionCtes.ISSUE_CERTIFICATE);
```

Indicamos mediante los parámetros extra que la emisión será de un certificado de firma:

```
ParameterAux param = new ParameterAux();

param.setKey(StartOpTransactionCtes.CERTIFICATE_TYPE);

param.setData(StartOpTransactionCtes.CERTIFICATE_TYPE_SIGN);

ParameterAux[] paramsList = new ParameterAux[1];

paramsList[0]=param;
```

Realizamos la llamada a startOpTransaction para comenzar la transacción de emisión:

```
StartOpTransactionResult result = new
GateWayAPI().startOpTransaction(usuarioSafecert, opInfo, paramsList);
```

La operación nos devuelve el identificador único de la transacción, con el que realizaremos las siguientes peticiones:

```
String id_transaction = result.getIdTransaction();
```

También nos devuelve la URL de redirección hacia la que se debe redirigir al usuario para continuar con la emisión/renovación del certificado de firma:

```
String URLRedireccion = result.getRedirect();
```

La aplicación realizará una redirección al usuario hacia la URL de getRedirect.

El usuario realiza la emisión/renovación de su certificado y se le redirige a la URL correspondiente según el resultado de la operación (OK o Error).

La aplicación volverá a tomar el control posteriormente cuando la Pasarela redirija al usuario de nuevo hacia una de las URLs que suministró la aplicación, ya sea la indicada con setRedirectOK o la pasada como setRedirectError.

Una vez que el usuario ha finalizado la emisión/renovación, la aplicación puede obtener el resultado de la operación.

```
DataTransactionResult resultado = new GateWayAPI().dataTransaction(id_transaction);
```

Comprobamos si la operación ha ido correctamente:

```
String resultOp = resultado.getStateTransaction().getResult();
```

El resultado contendrá el texto "OK" si la operación ha ido correctamente y "ERROR" en caso contrario.

Si la emisión/renovación del certificado ha sido correcta, se podrá recuperar el certificado de la transacción:

```
byte[] certificadoEmitido = respuesta.getCertificate();
```

Para dar por terminada la transacción, se finaliza:

```
EndTransactionResult result = new GateWayAPI().endTransaction(id_transaction);
```

5. INTEGRACION CON LA PASARELA DE FIRMA

5.1 Flujo funcional operación de firma

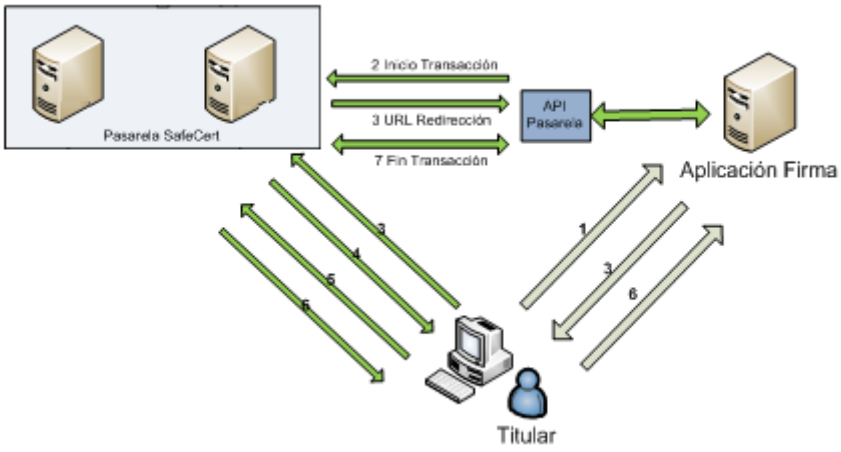


Ilustración 4: Flujo funcional Pasarela

Paso 1 – El usuario a través de la aplicación de Firma solicita la firma de un documento o conjunto de documentos.

Paso 2 – La aplicación una vez que desea iniciar una transacción de firma, puede solicitar el certificado del usuario y calcular el Hash de los documentos a firmar e iniciar la transacción.

Paso 3 – Como respuesta del inicio de transacción, el sistema de la Pasarela le contesta con la URL donde navegará para iniciar el proceso de firma. Esta navegación se realizará a través del navegador del usuario y llegará al sistema de la pasarela como solicitud de firma de una determinada transacción.

Paso 4 – Tras la recepción de la solicitud de firma por parte de la pasarela, se determinará la página que se le mostrará al usuario donde se visualizará el resumen con la información de los datos que se van a firmar y donde el usuario deberá proporcionar las credenciales de uso del certificado.

Paso 5 – En la respuesta del paso anterior se establecerá una cookie de sesión en el dominio de la pasarela para mantener la sesión con un determinado servidor Pasarela, de esta manera se podrá cachear el PIN asociada a la sesión del usuario si así se hubiera determinado por configuración.

Finalmente el usuario proporciona las credenciales de uso del certificado, siendo enviadas al mismo servidor Pasarela donde se estableció la sesión en el dominio de la pasarela.

Paso 6 – Una vez el usuario envía las credenciales de uso del certificado, la pasarela realiza la firma invocando a SafeCert, para posteriormente realizar la redirección a la URL donde la aplicación de firma podrá recuperar la información necesaria para componer los documentos firmados.

Paso 7 – Una vez que el navegador realiza el redirect, llegará nuevamente la llamada a la aplicación de firma para que recupere los datos asociados a la transacción de firma y poder componer la firma de cada documento. Una vez recuperado los datos, la aplicación podrá dar por terminada la transacción.

5.2 Flujos en función de configuraciones

5.2.1 Firma con PIN y/o SFDA

A la hora de realizar una firma, los certificados utilizados pueden requerir unas credenciales de autenticación para verificar que el certificado pertenece al usuario que está solicitando la firma.

Los certificados de firma de los que dispone un usuario en SafeCert pueden solicitar la autenticación mediante uno o varios tipos de credenciales según su configuración:

PIN: contraseña que protege el certificado.

SFDA: Segundo Factor de Autenticación. Este factor puede tener varios formatos: un token que le llega al usuario por SMS, una tarjeta de coordenadas que está en posesión del usuario…

Cuando un usuario vaya a realizar la firma de uno o varios documentos, se le presentará la página de la consola web del sistema de Pasarela, en la que se le solicitarán las credenciales necesarias en función del certificado utilizado, si el PIN está cacheado o no y si es necesario un SFDA.

5.2.2 Caché de PIN

El sistema Pasarela dispone para las operaciones de firma, de un procedimiento opcional de cacheo del PIN del certificado en la sesión web del usuario, de tal forma que sólo se le solicita en la primera firma, siempre que la configuración de la aplicación lo permita.

Esta caché del PIN es única por titular y certificado, y estará vinculada a la sesión iniciada por el navegador del usuario en el dominio de la pasarela, mediante una cookie de sesión.

Se recomienda utilizar la caché de PIN exclusivamente en entornos en los que no requieran de máxima seguridad en la generación de firmas. Por defecto, la caché de PIN deberá estar desactivada.

5.2.2.1 Configuración de la caché del PIN

La posibilidad de cachear el PIN en la sesión del usuario es opcional y configurable para cada aplicación que se integra con el sistema de Pasarela. En caso de que la aplicación no tenga configurada ésta propiedad, la posibilidad de cachear o no el PIN se delega en la configuración general de la Pasarela, según se muestra en la siguiente tabla:

Configuración	Cacheo de PIN
[GENERAL] *CACHE_PIN=TRUE* [APP] *CACHE_PIN=TRUE*	Sí se cachea el PIN.
[GENERAL] *CACHE_PIN=TRUE* [APP] *CACHE_PIN=FALSE*	NO se cachea el PIN.
[GENERAL] *CACHE_PIN=TRUE* [APP] *CACHE_PIN=*	Sí se cachea el PIN.
[GENERAL] *CACHE_PIN=TRUE* [APP]	Sí se cachea el PIN.
[GENERAL] *CACHE_PIN=FALSE* [APP]	NO se cachea el PIN.
[GENERAL] *CACHE_PIN=* [APP]	NO se cachea el PIN.

Tabla 33: Configuración caché PIN

5.2.2.2 Tiempo de caché y tiempo de inactividad

En caso de que la aplicación o, en su defecto, la configuración general permita el cacheado de PIN, se podrá indicar el tiempo que permanecerá el PIN cacheado en la sesión del usuario a nivel de aplicación y/o a nivel general de la Pasarela atendiendo a dos parámetros: *TIME_CACHE_PIN=3* (en minutos) y *TIME_INACTIVITY=1* (en minutos).

Con el parámetro *TIME_CACHE_PIN* se indica el tiempo en minutos que estará cacheado el PIN en la sesión del usuario. Este tiempo nunca podrá ser mayor que el tiempo de sesión establecido como configuración de sesión de la aplicación Pasarela en el servidor de aplicaciones.

Con el parámetro *TIME_INACTIVITY* se indica el tiempo máximo que puede transcurrir desde el último uso del certificado para que no expire el cacheo del PIN. Si este parámetro es 0 se considerará que, en cada uso, se habrá superado el tiempo de inactividad por lo que siempre se pedirá el PIN al usuario. Si el valor es inferior al valor de *TIME_CACHE_PIN* se solicitará PIN al usuario cuando se haya superado el tiempo de inactividad configurado. Si el valor es superior al *TIME_CACHE_PIN* el tiempo de inactividad no tendrá repercusión sobre el cacheo del PIN (ya que el tiempo de caché de PIN expirará antes que el de inactividad).

Por lo tanto, se solicitará el PIN al usuario en caso de:

- No hay caché de PIN.
- El tiempo de inactividad del certificado ha expirado.
- El tiempo de caché de PIN ha expirado.
- El tiempo de sesión web ha expirado.
- El usuario cierra el navegador y abre uno nuevo.
- El PIN introducido no es correcto.

5.2.3 Forzado de PIN

Aunque una aplicación tenga configurado el cacheo de PIN de los certificados, ésta puede obligar al usuario a introducir el PIN en todas las firmas, enviando el parámetro *forcePIN*=true, siempre que la configuración de la aplicación lo permita.

5.2.4 Forzado de SFDA

Si un certificado de firma tiene establecido como Segundo Factor un SFDA a la carta, en función de las condiciones que tenga configuradas puede ocurrir que no sea necesario introducir los valores del SFDA cuando se realiza una firma.

En este caso, la aplicación puede obligar al usuario a introducir los valores del SFDA en todas las firmas, enviando el parámetro *forceSFDA*=true, siempre que la configuración de la aplicación lo permita.

5.2.5 Modo 'silencioso'

Este flujo de firma se puede llevar a cabo cuando se cumplen las siguientes premisas:

La configuración de la aplicación, o en su defecto la configuración general, permite el cacheo y el PIN ya está cacheado.

El certificado de firma no requiere SFDA o es un SFDA a la carta y ya se han introducido los valores.

La configuración de la aplicación, o en su defecto la configuración general, indica que no se le muestre al usuario un resumen de los datos a firmar.

En la primera firma que realice el usuario se le solicitará el PIN del certificado y éste se guardará en la caché el tiempo estipulado.

Se denomina *silencioso* porque las firmas posteriores se realizarán de forma transparente para el usuario, sin solicitarle ninguna credencial y redirigiendo su navegador a la aplicación de firma.

5.3 Periodo de vida de una transacción

El periodo de vida de una transacción es limitado. Cuando se inicia una nueva transacción, se establece su fecha de caducidad en función del momento de creación y el tiempo de vida útil configurado para las transacciones de esa aplicación (*MAX_TIME_LIFE_TRANSACTION=5*).

Si una transacción iniciada no se firma dentro de ese periodo de tiempo, la transacción caducará y ya no se podrá firmar.

5.4 Recuperación de datos de firma

Cuando una transacción activa concluye correctamente con la generación de la firma o firmas por el usuario, se puede recuperar la información de la misma con la operación 'dataTransaction':

```
DataTransactionResult resultado = new GateWayAPI().dataTransaction(id_transaction);
```

Si una transacción ha finalizado correctamente, el resultado general de la operación será *'OK'*:

```
resultado.getStateTransaction().getResult();
```

Si ha habido algún error en la operación, obtendremos el código del error y una descripción del mismo:

```
resultado.getStateTransaction().getCode_error();
```

```
resultado.getStateTransaction().getDescription();
```

Si la operación ha finalizado correctamente, la lista de hashes firmados pueden recuperarse en dos formatos:

Sin codificar:

```
resultado.getSigns().get(n).getSign();
```

Codificados en Base64:

```
resultado.getSigns().get(n).getSignB64();
```

5.5 Interacción de las aplicaciones usuarias de la Pasarela

En este apartado nos centramos en las interacciones de las aplicaciones con los diferentes elementos implicados en la solución de la pasarela de firma.

El inicio del proceso de una determinada operación partirá a través de la aplicación integrada con la pasarela.

5.5.1 Operación de firma o autenticación

Cuando el usuario decide firmar o autenticarse, la secuencia de pasos a realizar por parte de la aplicación integrada con la pasarela será:

1. Solicitar al API de la pasarela la lista de certificados que el usuario tiene asociados en Safecert y elegir con cual desea que el usuario realice la operación. Normalmente, esta operación no será necesaria porque el usuario solo dispondrá de un certificado activo, en este caso se podrá directamente iniciar la transacción sin especificar un certificado. En caso de haber varios será responsabilidad de la aplicación seleccionar uno de los certificados ya sea delegando la elección en el usuario o seleccionando uno directamente en base a algún tipo de filtro basado en usos de clave o similar.

2. Invocar al API de la pasarela para iniciar una nueva transacción.

    a. En caso de que la operación a llevar a cabo sea una firma, en el inicio de la transacción se deberán indicar al API la información sobre los documentos que se deben firmar para la transacción en curso. Para cada documento se debe indicar:
        i. Identificador único del documento. Este identificador servirá para identificar y recuperar el documento una vez firmado.
        ii. Nombre del documento.

iii. Título del documento. Tanto el nombre como el título del documento permitirán al usuario conocer qué va a firmar en la consola de la pasarela de firma.
iv. Hash del documento
v. Algoritmo del hash utilizado.

b. Indicar al API de la pasarela que inicie el proceso. Esto devolverá como respuesta una URL a la que la aplicación debe redirigir al usuario, por ejemplo con una redirección HTTP 302:

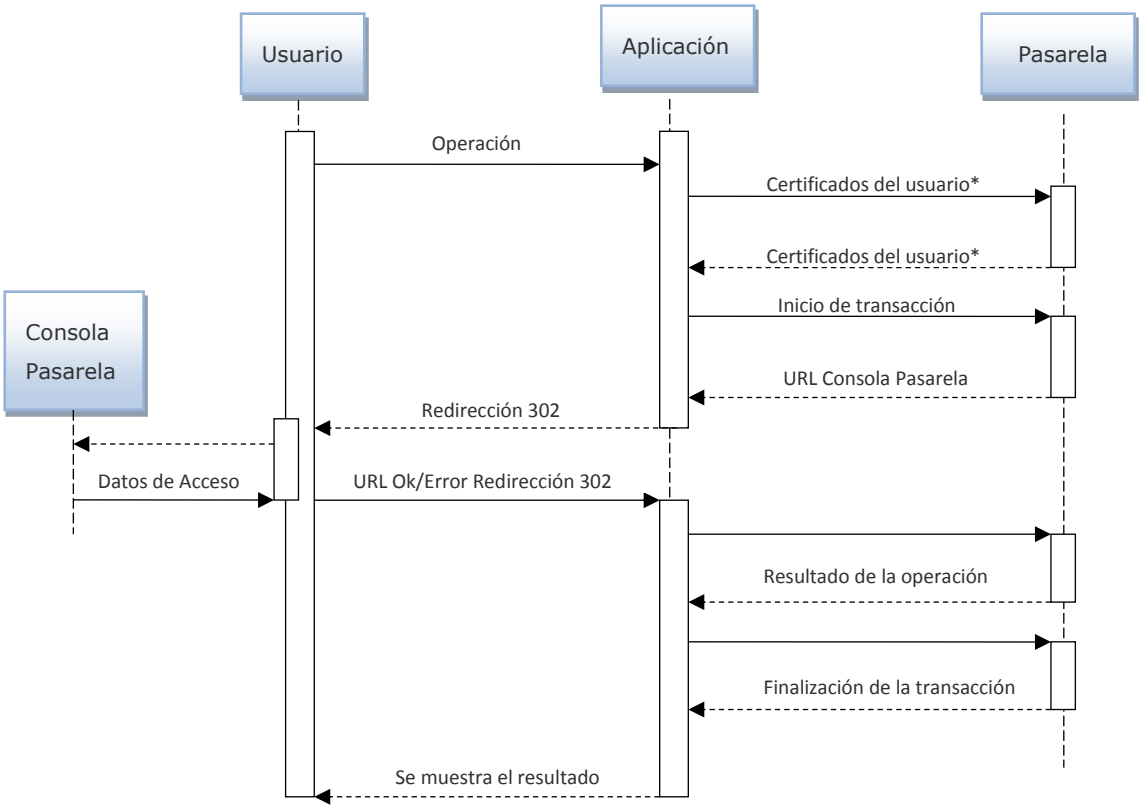
```
HTTP/1.1 302 Found
Location:https://safecert/rss-gateway/AuthGateWayServlet?id_transaction=62afbde9febf123ec12
```

3. Cuando el usuario haya finalizado la operación de forma satisfactoria, se redirigirá al usuario desde la pasarela a la URL indicada en el inicio de la transacción como redirectOK, en caso contrario se le redirigirá a la redirectError. La aplicación por lo tanto deberá ser capaz de a través de esta URL de recuperar el control del proceso del usuario en la aplicación.

4. La aplicación solicitará al API de la pasarela que le devuelva el estado final de la operación. Para ello deberá indicar el identificador de la transacción y si el proceso acabó correctamente se podrá recuperar los datos asociados a la operación. Para una operación de firma los documentos firmados y para una operación de autenticación el certificado y los datos del titular.

5. Tras finalizar el proceso, la aplicación deberá solicitar a través del API el cierre de la transacción de forma que se pueda eliminar definitivamente la transacción. Desde este momento ya no estará disponible la información de la transacción.

El siguiente diagrama explica el proceso de una operación en la pasarela de firma.



**Operación opcional*

5.5.2 Operación de firma incluyendo la emisión/renovación desde Pasarela

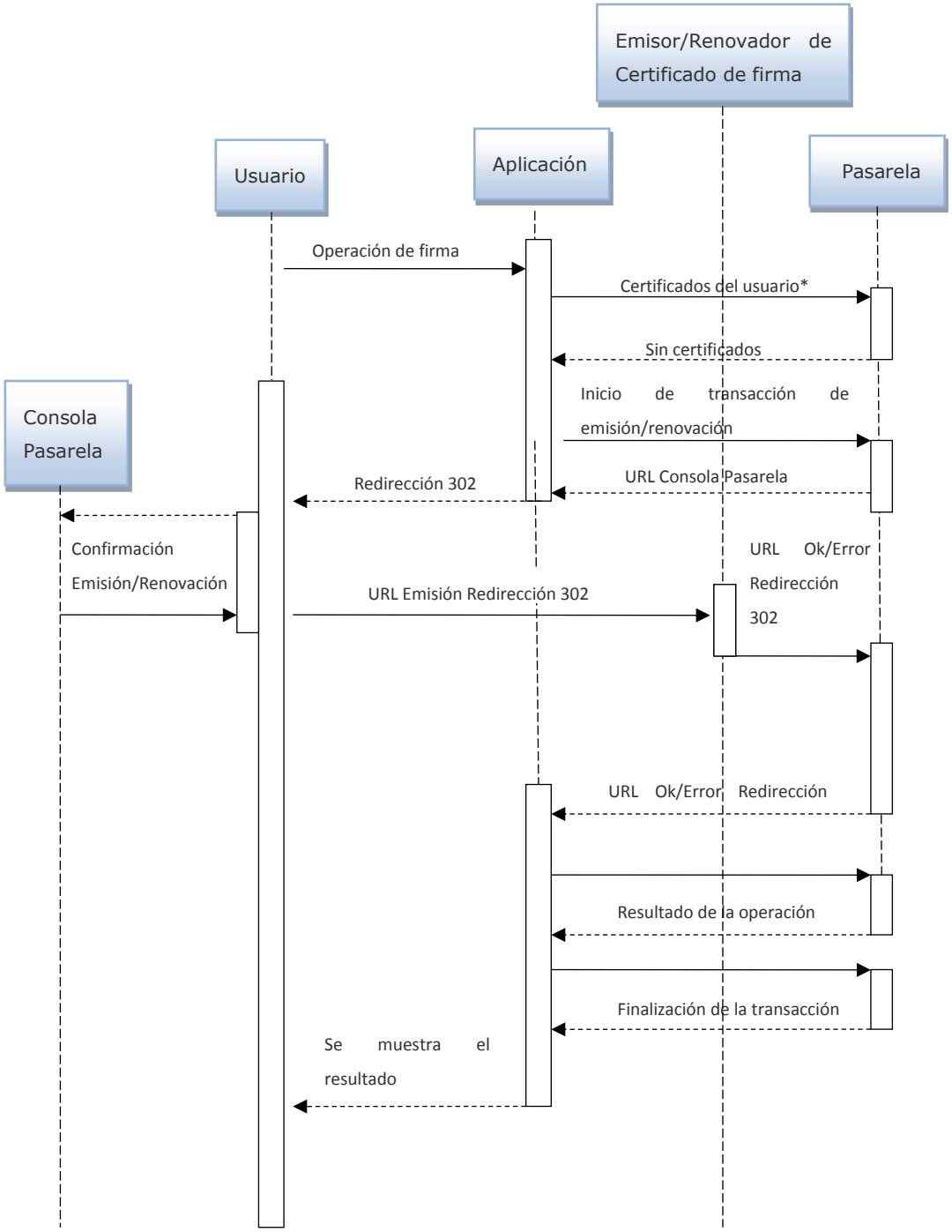
El caso que a continuación se detalla se produce cuando la emisión/renovación de certificados de firma se realiza desde Pasarela.

La interacción entre la aplicación integradora y pasarela en un proceso de firma será tal y como se describe a continuación:

1. La aplicación integradora solicita el/los certificados de firma a pasarela a través del API.

2. Pasarela tiene el parámetro FORCE_REISSUE_CERTIFICATE activado. El certificado destinado a la firma del usuario se encuentra en uno de los siguientes estados:
    - El usuario no tiene certificado de firma.
    - El certificado de firma del usuario se encuentra caducado.
    - El certificado de firma del usuario se encuentra en ventana de caducidad.
3. Si la aplicación obtiene un certificado, ya puede comenzar la transacción de firma.
4. Si la aplicación integradora no recibe ningún certificado, esta debe comenzar una transacción de emisión/renovación de certificado.
5. SafeCert Pasarela retorna una URL como resultado de inicio de transacción de emisión/renovación a la aplicación integradora y esta debe redirigir al usuario hacia esa URL.
6. El usuario interactúa con el portal de emisión/renovación y completa el proceso. Una vez termine, se devolverá el control a la aplicación.
7. La aplicación solicitará al API de la pasarela que le devuelva el estado final de la operación. Para ello deberá indicar el identificador de la transacción y si el proceso acabó correctamente se podrá recuperar los datos asociados a la operación. Para la operación de emisión tan solo se puede obtener si la operación ha ido correctamente o se ha producido algún error.

Tras comprobar que la emisión o renovación del certificado ha sido correcta, la aplicación integradora deberá comenzar una nueva transacción de firma.



5.6 Integración de las aplicaciones

Las aplicaciones deberán tener en cuentas los siguientes puntos:

- En las operaciones de firma, la identificación y autenticación del usuario corre a cargo de las aplicaciones. El sistema de la Pasarela confiará en que el identificador de usuario que la aplicación le proporcione como parámetro se corresponde con el identificador de dicho usuario en el sistema.

- El proceso de una operación en la pasarela se realizará para un certificado concreto del usuario. Aunque normalmente un usuario tendrá un solo certificado activo, las aplicaciones deben contemplar la posibilidad de que tenga varios. El sistema podrá filtrar los certificados por operación en función del KeyUsage de los mismos, si aun así el usuario dispusiera de varios certificados para una determinada operación, corre a cargo de las aplicaciones seleccionar el certificado a utilizar en la operación.

- La consola de la pasarela no mostrará al usuario más información que la estrictamente necesaria para la operación, y solo se le mostrarán errores sobre datos erróneos introducidos, en otro caso se redirigirá al usuario a la URL de Error indicada por la aplicación. La aplicación podrá consultar el error producido a través del API de firma y decidir cómo y el qué mostrar al usuario.

- El API de la pasarela se ejecuta en el entorno de la aplicación integrada. Las dependencias del API de la pasarela son las siguientes.

Librería	Versión	Descripción
gateway-api-2.4.04.jar	2.4.04	API de pasarela
gateway-api-comun-2.4.04.jar	2.4.04	API de pasarela
log4j-1.2.14.jar	1.2.14	Librería generación de logs
hessian-3.2.0.jar	3.2.0	Librería de llamada a servicios de la pasarela
commons-lang-2.6.jar	2.6.0	Procesamiento de peticiones a los servicios de SafeCert.

- La configuración de log se realizará utilizando LOG4J.
- La versión mínima de máquina virtual JAVA del API de firma será la 1.5
- El API pasarela requiere de un properties para su correcto funcionamiento.

5.7 Casos de uso más comunes

A continuación se especificarán los casos de uso más frecuentes, determinando su configuración y los resultados esperados.

5.7.1 Autenticación con usuario inexistente

Cuando una aplicación intente iniciar una transacción de autenticación indicando un titular que no esté dado de alta en el sistema SafeCert, recibirá una excepción ***SafeCertGateWayException*** con código de error **OPSTR00011**.

5.7.2 Autenticación con nivel OTP

Cuando una aplicación intente iniciar una transacción de autenticación con un nivel de ConstantsGateWay.AUTH_OTP sin establecer el identificador de titular recibirá una excepción ***SafeCertGateWayException*** con código de error **OPSTR00013**. Una transacción de autenticación con nivel **ConstantsGateWay.AUTH_OTP** requiere necesariamente que se especifique el identificador del titular.

5.7.3 Transacción de autenticación sin especificar identificador de titular

Cuando se inicie la transacción de autenticación sin especificar identificador de titular, se recibirá la URL de redirección a la pasarela, en esta se presentará una pantalla solicitando identificador de titular y contraseña.

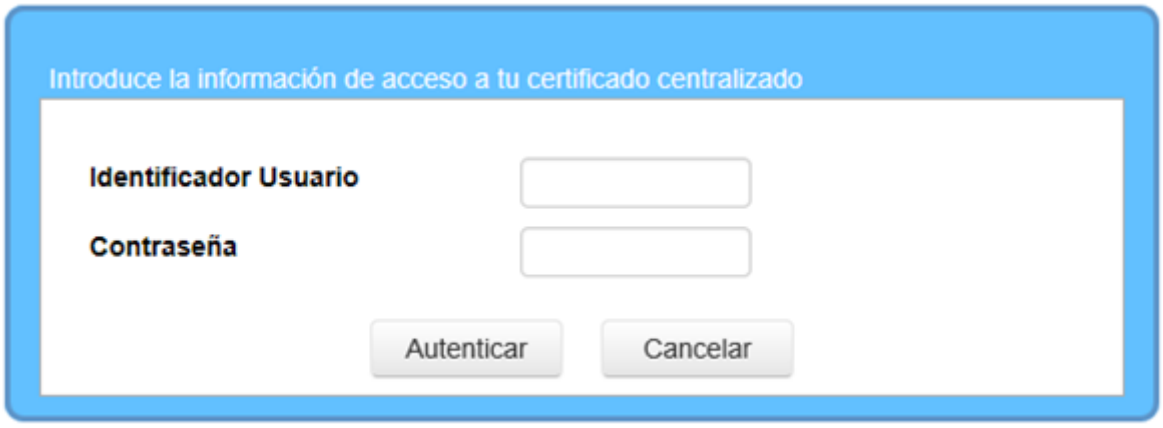


Ilustración 5: Solicitud de autenticación con contraseña

En caso de que el nivel de autenticación sea **ConstantsGateWay.AUTH_PIN_OTP,** se le mostrará posteriormente una pantalla solicitando el valor de la OTP:

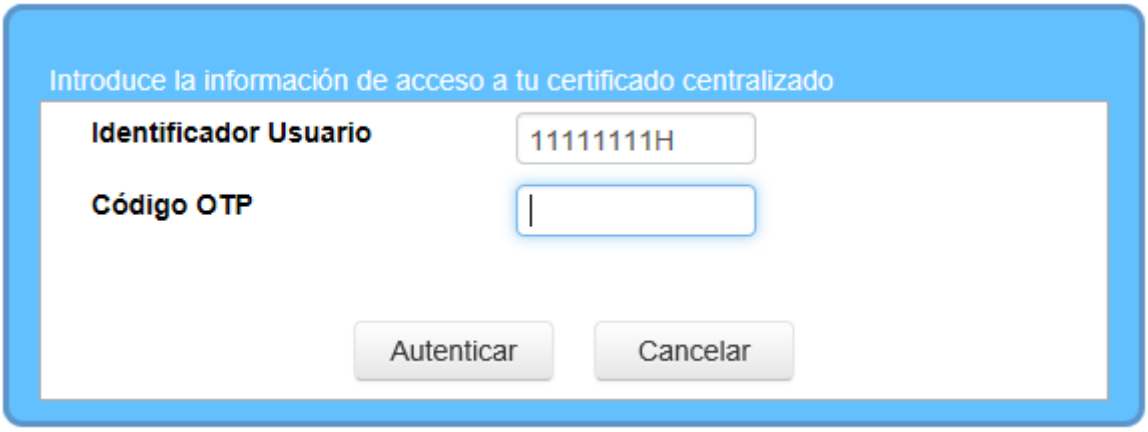


Ilustración 6: Autenticación con contraseña y OTP

5.7.3.1 Titular inexistente

Cuando se establece un identificador de titular que no existe en el sistema SafeCert, se mostrará un error indicando que la validación ha sido incorrecta sin dar información de la no existencia del identificador del titular en el sistema.

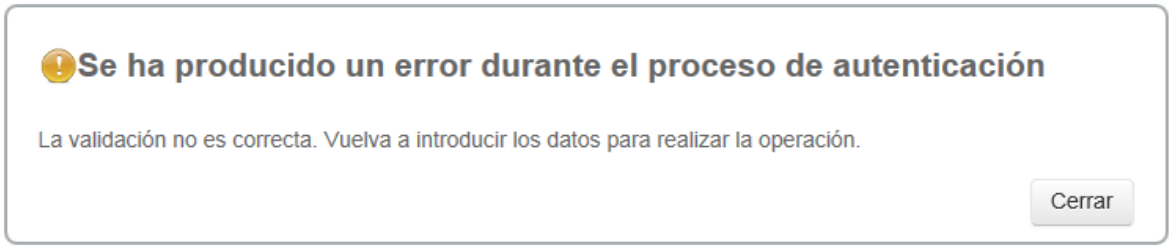


Ilustración 7: Usuario no existe en el sistema

Si el usuario pulsa el botón cerrar, se volverá a la pantalla anterior.

5.7.4 Contraseña incorrecta

Cuando el usuario introduce una contraseña incorrecta en la pasarela, se mostrará un aviso informativo al usuario.

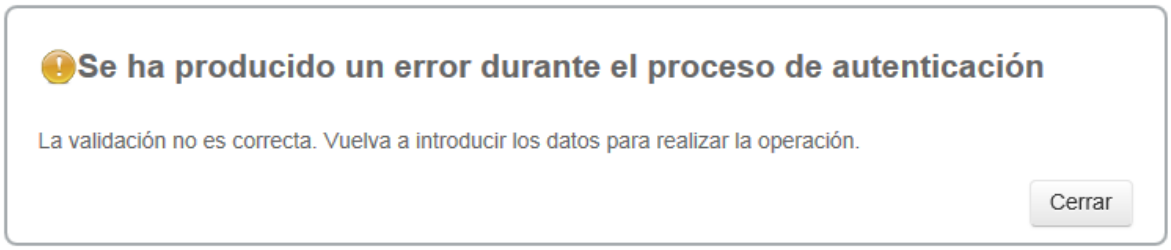


Ilustración 8: Contraseña incorrecta

El sistema le solicitará la contraseña tantas veces como el número máximo de intentos permitidos se establezcan en el sistema, tanto para el bloqueo del certificado como para la suspensión del certificado.

En caso de llegar hasta el número máximo permitido hasta la suspensión, se mostrará un aviso indicando que el uso del certificado se ha suspendido temporalmente.

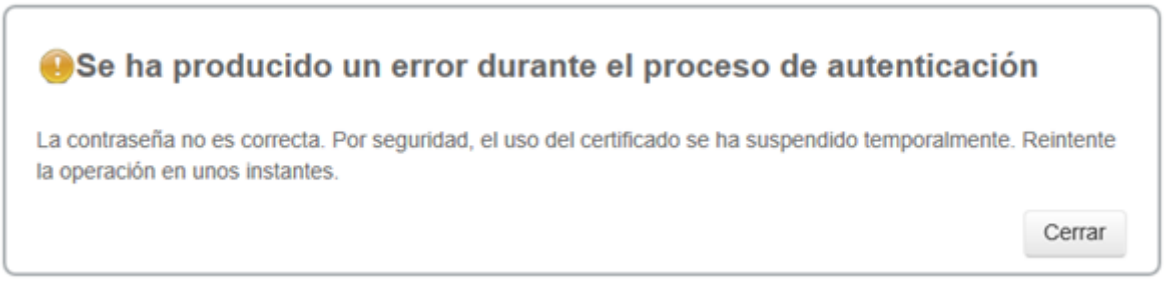


Ilustración 9: Suspensión temporal del uso del certificado

Si se reintenta la operación y aún el certificado se encuentra suspendido, se mostrará el siguiente mensaje.

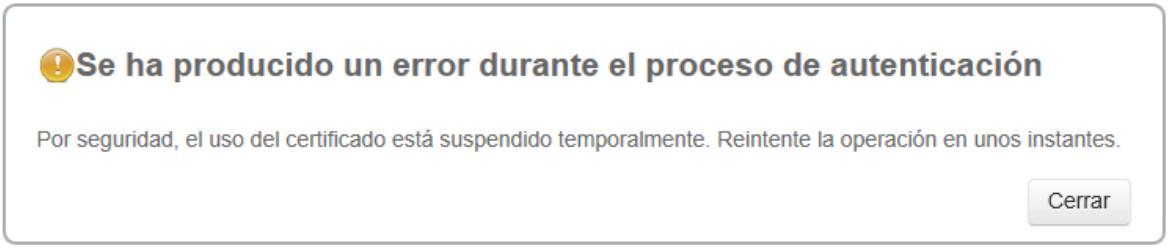


Ilustración 10: Certificado temporalmente suspendido

5.7.5 Certificado bloqueado

Si se alcanza el número máximo de intentos permitidos antes de bloquear el certificado, se finalizará la transacción indicando que el certificado ha sido bloqueado mediante el código de error **AUTHE00104** si la operación es una autenticación o mediante **MSIGN00104** si es de firma.

En caso de iniciar una transacción indicando el identificador de titular y este tenga el certificado bloqueado por superar el número de intentos erróneos, se devolverá una excepción ***SafeCertGateWayException*** con código de error **OPSTR00017.** En caso de que el certificado esté bloqueado por causas administrativas el código será **OPSTR00016.**

En caso de iniciar una transacción sin especificar el identificador de titular, y el usuario ya tiene el certificado bloqueado, una vez que se efectúe la validación en la pasarela, se redirigirá al navegador a la URL de error finalizando la transacción con el código de error **WEBCT00021,** en caso de que el certificado esté bloqueado por causas administrativas el código será **WEBCT00020.**

5.7.6 Caducidad de contraseñas

Tras cada operación de autenticación o de firma si se detecta que la contraseña ha caducado se le solicitará al usuario que debe cambiar su contraseña antes de poder finalizar con su operación.

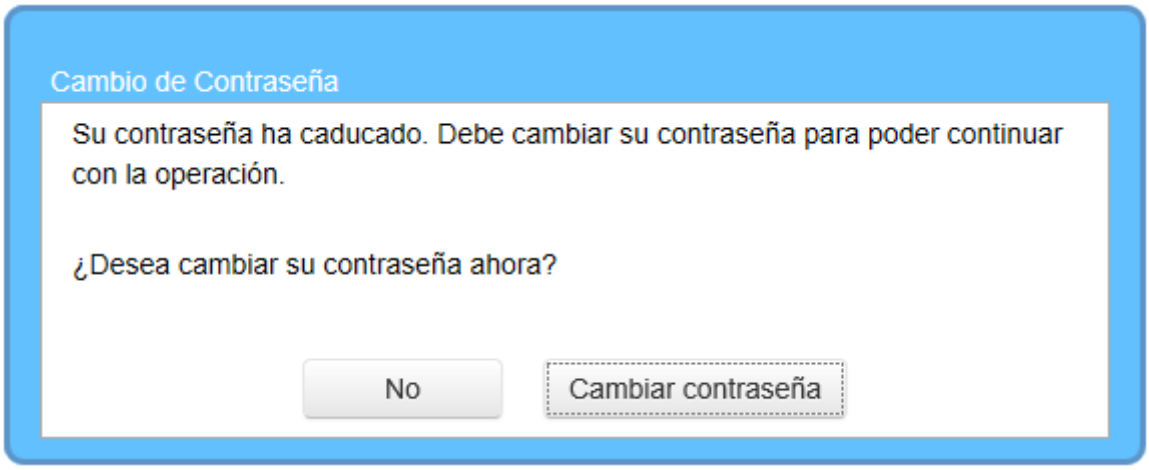


Ilustración 11: Contraseña caducada

En el caso de que la contraseña esté cercana a caducar, una vez finalizada la operación en curso, se le informará al usuario de la proximidad de caducidad dándole la opción de cambiarla en ese momento.

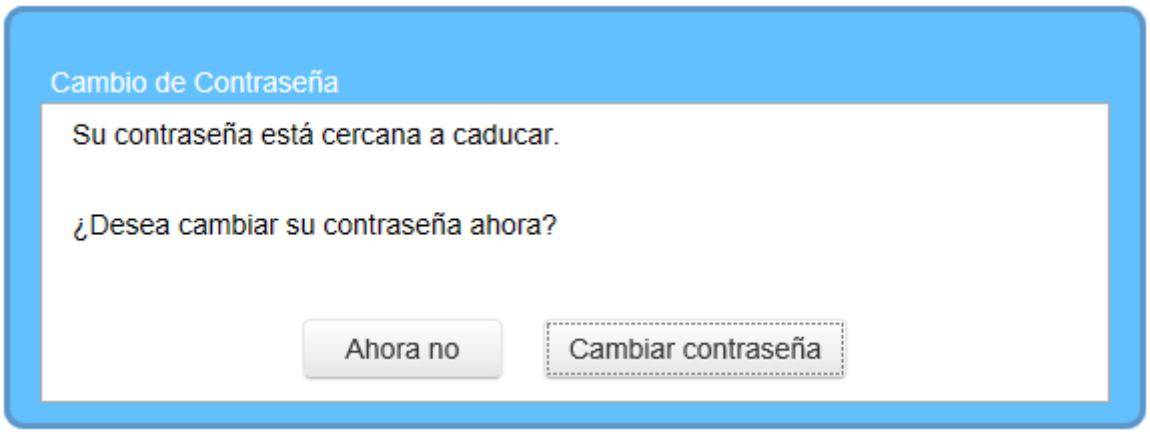


Ilustración 12: Contraseña cercana a caducar

5.7.7 Autenticación mediante SFDA

Cuando la autenticación se realiza utilizando un segundo factor de autenticación, y se introduce una OTP incorrecta se visualizará el mismo mensaje de error de validación incorrecta.

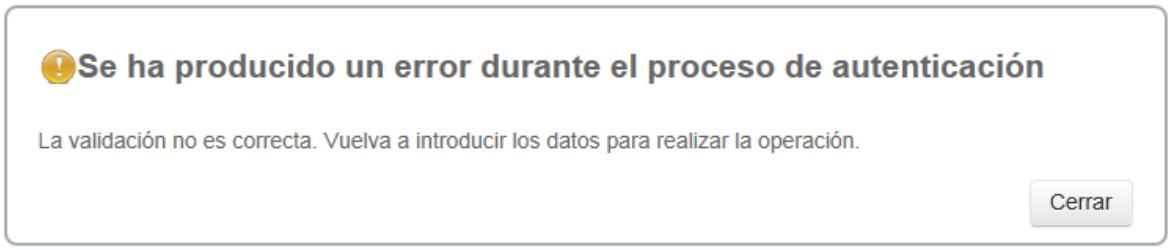


Ilustración 13: OTP incorrecta

Si se alcanza el número máximo de intentos permitidos antes de cancelar la OTP, se finalizará la transacción redirigiendo a la URL de error indicando que la OTP ha sido cancelada mediante el código de error **AUTHE00112** si la operación es una autenticación o mediante **MSIGN00112** si es de firma.

Las contraseñas (OTP) de segundo factor de autenticación pueden configurarse para que expiren en un determinado periodo de tiempo, si la OTP caducó antes de realizar la validación se finalizará la transacción redirigiendo a la URL de error indicando que la OTP ha expirado mediante el código de error **AUTHE00115** si la operación es una autenticación o mediante **MSIGN00115** si es de firma.

5.7.8 Caducidad de certificados

En la operación de autenticación o de firma si se detecta que el certificado ha caducado se le solicitará al usuario que renueve su certificado antes de poder continuar con la operación.

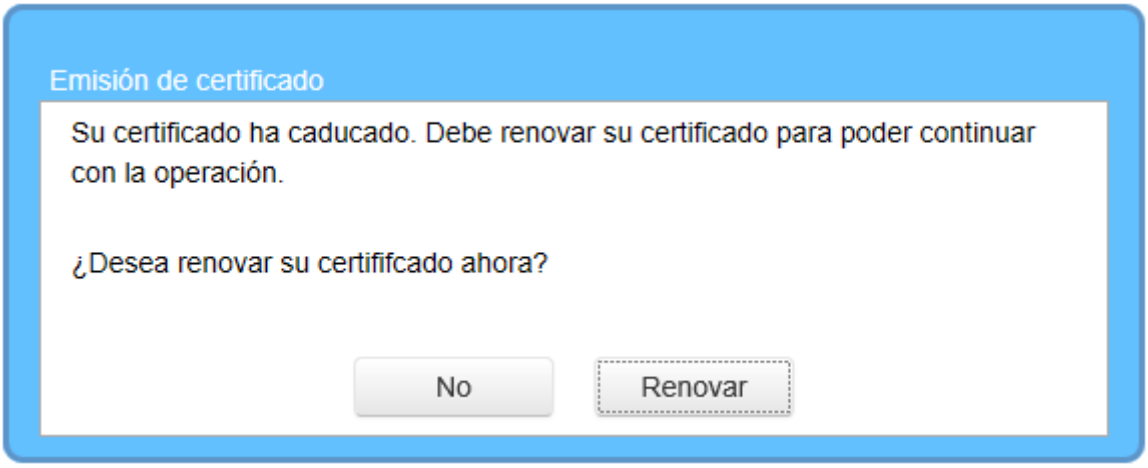


Ilustración 14: Certificado caducado

Igualmente a la caducidad de contraseñas, se le informará al usuario de la proximidad de la caducidad de su certificado. En este caso, tras completar la operación se le ofrecerá la posibilidad de renovar su certificado.

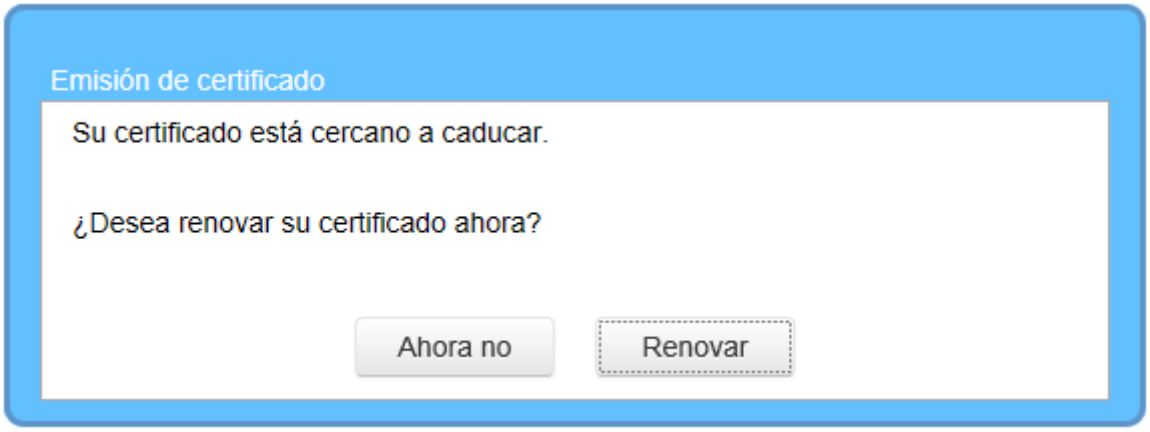


Ilustración 15: Certificado próximo a caducar

5.7.9 Emisión del certificado de firma

Durante una operación de firma, si el titular no dispone de ningún certificado para realizar la operación de firma se le ofrecerá la opción de poder emitir el certificado en ese momento. Si su certificado de firma se encuentra en un estado no operativo pero si dispone de certificado, no aparecerá la opción de emitir el certificado.

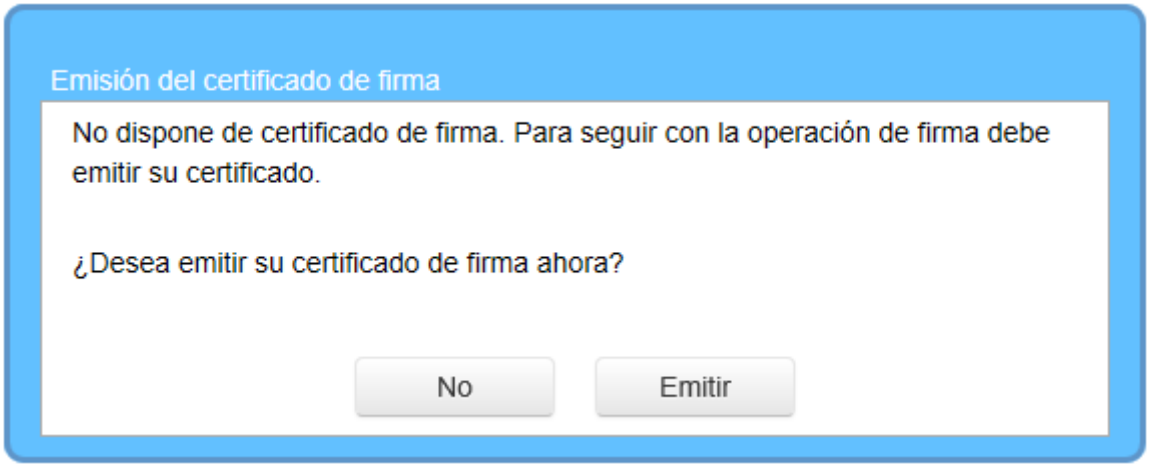


Ilustración 16: Emisión certificado de firma

5.7.10 Transacción caducada

Cada transacción tiene un tiempo de vida, superado este tiempo la transacción se finalizará informando a través de la URL de error especificada en el inicio de la transacción y con el código de error **TRANS00011.**

5.8 Errores

A continuación se muestra una lista de los errores más comunes que se pueden dar al utilizar el sistema de Pasarela:

ERRORES	
Código	**Descripción**
WSAPI00001	Error genérico en los servicios Gateway.
WSAPI00006	Error al inicializar los servicios Gateway, no se encuentra el fichero con el certificado de autenticación.
	SERVICIO QUERYCERTIFICATES
OPQUE00001	Se ha producido un error en la operación QueryCertificates.
SERVH00003	No se ha podido recuperar los datos XML de la operación. Verifique el esquema de datos de la operación solicitada.

OPQUE00003	El identificador del titular no existe en SafeCert.
	SERVICIO STARTTRANSACTION
OPSTR00006	No se permite forzar la solicitud del PIN.
OPSTR00007	No se permite forzar la solicitud del segundo factor de autenticación.
OPSTR00008	El usuario no dispone de certificados preparados para firmar.
OPSTR00009	El usuario dispone de más de un certificado. Es necesario indicar un certificado.
OPSTR00011	El identificador del titular no existe en SafeCert.
OPSTR00012	El certificado especificado no es correcto o ha dejado de ser operativo.
OPSTR00013	El identificador de titular es necesario cuando la operación solamente requiera segundo factor de autenticación
OPSTR00014	El certificado no dispone de segundo factor de autenticación
OPSTR00015	El nivel de autenticación no está soportado
OPSTR00016	El certificado está bloqueado para su uso
OPSTR00017	El certificado está bloqueado por exceder el número máximo de intentos
OPSTR00018	El certificado no está activo
	SERVICIO STARTOPTRANSACTION
OPSTR00011	El identificador del titular no existe en SafeCert.
OPSTR00019	El nombre de la operación para la transacción no está permitido.
OPSTR00020	El uso del certificado a emitir no es correcto.
OPSTR00021	El titular ya dispone de un certificado para el uso indicado.
OPSTR00022	El certificado de firma del titular está caducado.

	OPERACIÓN DE FIRMA Y AUTENTICACIÓN SIN INDICAR NIVEL
MSIGN00104	Error de PIN y certificado bloqueado.
MSIGN00105	Error de SFDA y usuario bloqueado en el sistema de segundo factor de autenticación.
MSIGN00107	Error por Certificado inactivo.
MSIGN00108	Error de autenticación.
MSIGN00112	Reto de segundo factor anulado. Debe iniciar la operación nuevamente.
MSIGN00113	El certificado no dispone de segundo factor de autenticación
MSIGN00114	El usuario no dispone de certificado.
MSIGN00115	El valor del segundo factor de autenticación expiró.
MSIGN00116	La contraseña ha caducado
MSIGN00117	El certificado ha caducado
MSIGN00118	Error de autenticación en el proceso de firma
WEBCT00016	Operación cancelada por el usuario.
	OPERACIÓN DE AUTENTICACIÓN INDICANDO NIVEL
AUTHE00104	Error de PIN y certificado bloqueado.
AUTHE00105	Error de SFDA y usuario bloqueado en el sistema de segundo factor de autenticación.
AUTHE00107	Error por Certificado inactivo.
AUTHE00108	Error usuario no existe.
AUTHE00112	Reto de segundo factor anulado. Debe iniciar la operación nuevamente.
AUTHE00113	El certificado no dispone de segundo factor de autenticación

AUTHE00114	El usuario no dispone de certificado.
AUTHE00115	El valor del segundo factor de autenticación expiró.
AUTHE00116	La contraseña ha caducado
AUTHE00117	El certificado ha caducado
WEBCT00016	Operación cancelada por el usuario.
	SERVICIO DATATRANSACTION
DTAPI00003	El parámetro identificador de transacción es nulo para la operación DataTransaction.
OPDTR00001	Se ha producido un error en la operación DataTransaction.
OPDTR00003	La transacción ha caducado.
OPDTR00005	La transacción no ha finalizado y aún no se dispone de la información de la operación.
	SERVICIO ENDTRANSACTION
ETAPI00003	El parámetro identificador de transacción es nulo para la operación EndTransaction.
OPETR00004	La transacción no existe.
OPETR00005	La transacción todavía no ha finalizado.
	CÓDIGOS DE CONSOLA
WEBCT00015	Se ha producido un error al realizar la operación
WEBCT00016	Operación cancelada por el usuario
WEBCT00017	El usuario indicado no ha sido dado de alta en el sistema
WEBCT00018	La transacción ya ha finalizado

WEBCT00019	El usuario no dispone de certificados
WEBCT00020	El certificado se encuentra bloqueado por razones administrativas
WEBCT00021	El certificado se encuentra bloqueado por superar el número máximo de contraseñas incorrectas
WEBCT00022	El certificado no se encuentra activo
WEBCT00023	La transacción se encuentra en un estado incorrecto
WEBCT00024	La transacción ha caducado
WEBCT00025	La emisión del certificado ha fallado
WEBCT00026	El cambio de contraseña ha fallado
WEBCT00027	No se ha podido realizar la firma porque el certificado ha sido renovado
TRANS00011	Transacción caducada

Tabla 34: Códigos de error

6. PERSONALIZACIÓN DE PLANTILLAS XSL

Pasarela permite la personalización de las ventanas que interactúan con el usuario. Para ello se utilizan plantillas XSL que permiten generar una página HTML a partir de un XML de entrada y cuya estructura se detalla en los siguientes esquemas: ***transactionInfo.xsd*** (esquema utilizado para la generación del XML en las operaciones de firma y autenticación) y ***decisionInfo.xsd*** (utilizado para la generación del XML en la ventana de toma de decisiones por parte del usuario).

Es aconsejable que bajo ningún concepto se modifique la lógica de negocio de las plantillas que se proporcionan con el producto, dejando configurable el aspecto visual de las ventanas. Dichas plantillas son:

- **pasarela.xsl:** plantilla utilizada para la operación de firma.
- **ownerAuth.xsl:** plantilla utilizada para la operación de autenticación.
- **decision.xsl:** plantilla utilizada para la ventana de toma de decisiones por parte del usuario.

En caso de requerir la personalización de las plantillas es necesario modificar el fichero de configuración de la pasarela (modificando cada sección por aplicación), configurando los siguientes parámetros:

XSLT_APP: indica la ruta completa de la plantilla utilizada para una operación de firma.

XSLT_APP_OWNER: indica la ruta completa de la plantilla utilizada en una operación de autenticación.

XSLT_APP_DECISION: indica la ruta completa de la plantilla utilizada para operaciones en la que se requiere la toma de decisiones por parte del usuario. Por ejemplo, renovar un certificado o no que está a punto de caducar.

I18N_APP: ruta completa del fichero de textos utilizado por las plantillas.

CSS_APP: ruta completa del fichero hoja de estilos utilizado por las plantillas.

6.1 Plantilla utilizada en la operación de firma

El aspecto de la ventana presentada para las operaciones de firma viene marcada por dos nodos del xml utilizado por la plantilla de firma:

RequestPwd: en la operación de firma, para un titular concreto, el valor "true" en este nodo indicará que la operación de firma requiere de la introducción de la contraseña del certificado por parte del usuario. Por tanto, la plantilla debe presentar un cuadro de texto en el que el usuario podrá introducir la contraseña.

RequestSFDA: indica si la operación de firma requiere (con valor "true") o no (con valor "false") la introducción de una contraseña para validar un segundo factor de autenticación. En la actualidad existen varios segundos factores de autenticación que se podrían clasificar en dos tipos:

1. Aquellos que requieren una contraseña, por lo que solo sería necesario mostrar una caja de texto para su introducción.
2. Aquellos que necesitan que el usuario introduzca uno o más datos por parte del usuario. Por ejemplo, el Segundo Factor de Autenticación "Tarjeta de Coordenadas" necesita presentar al usuario las celdas que el usuario debe completar para autenticarse. Esto implica que es necesario mostrar más de una caja de texto para que el usuario complete la operación. En la plantilla de ejemplo que se proporciona con el producto para las operaciones de firma, se permite la integración con un segundo factor de autenticación de tipo "GRID", que ya presenta las celdas para el usuario.

La información necesaria que se pudiera requerir para la presentación o petición de datos al usuario estará contenida en el nodo xml ***SfdaInfo***. En la plantilla que se proporciona con el producto se sigue la siguiente lógica para la personalización del aspecto con respecto al segundo factor de autenticación:

1. En primer lugar se evalúa si es necesario o no la introducción de un valor por parte del usuario, o lo que es lo mismo, si la operación requiere o no de autenticación mediante segundo factor. Para ello se inspecciona el nodo ***RequestSFDA.*** Si dicho nodo tiene un valor "true", entonces la operación requiere de segundo factor de autenticación.
2. Se inspecciona el nodo ***SfdaInfo*** y más concretamente un nodo hijo de éste y de nombre ***SfdaTypeAuth.*** Si el tipo de factor de autenticación es de tipo "*GRID*" (es el único que en la actualidad requiere la introducción de más de un dato por parte del usuario), entonces se consulta qué casillas de la tarjeta de coordenadas debe introducir el usuario. Para ello se recorre la lista del nodo xml ***SfdaValues*** y se presenta el valor de cada una de las celdas al usuario, junto con una caja de texto para su introducción. En caso de que el segundo factor de autenticación fuera de otro tipo, se presenta una simple caja de texto.

La representación visual del Segundo Factor de Autenticación depende de los datos que requiera, por lo que es necesario consultar el modo de funcionamiento de cada uno de los segundos factores.

6.1.1 Evento Submit

Con independencia de la representación visual de la ventana en la operación de firma, el formulario que se envía al servidor de aplicaciones de pasarela debe contener los siguientes parámetros:

id_transaction: Identificador de la transacción en curso. Este parámetro viene indicado en el XML de datos, más concretamente en el atributo *Id* del nodo *TransactionDetails.*

pin: contraseña del certificado del usuario.

sfdaValue0: valor para el segundo factor de autenticación. En caso de que el segundo factor de autenticación requiera de más de un valor, habría que mostrar más de una caja de texto para que el usuario pueda introducir cada valor y los identificadores de cada una de esas cajas deben ser sfdaValue1, sfdaValue2…

6.1.2 Presentación de textos y posibles errores

Para encontrar posibles errores que se puedan producir durante la operación de autenticación, es necesario inspeccionar el nodo ***errorCode***, hijo del nodo ***TransactionResponse***. En la plantilla que se proporciona con el producto, el mensaje que describe cada uno de los errores se encuentra en el fichero de textos que se proporciona en la entrada *I18N_APP.* Con el producto se proporciona un fichero de textos de nombre *GateWayTexts.properties* que incluye los textos y mensajes asociados a posibles errores que se puedan encontrar. Los códigos de error en la operación de firma comienzan por ***MSIGN*.**

6.2 Plantilla utilizada en la operación de autenticación

Al igual que en la plantilla utilizada para la operación de firma, la de autenticación guía la presentación de la ventana de autenticación en pasarela a través de una serie de nodos del XML de datos. En este caso son tres los nodos (en lugar de dos) que son necesarios evaluar para realizar la presentación de la ventana. A los ya mencionados ***RequestPwd*** y ***RequestSFDA*** (y cuyo significado y funcionamiento se puede aplicar del mismo modo para la ventana de autenticación), hay que sumar la evaluación del nodo ***RequestDatoContraste.*** En caso de que este nodo contenga el valor "true", es necesario mostrar el dato de contraste al usuario. Existen cuatro tipos de datos de contraste y por tanto es necesario inspeccionar el XML de datos para saber qué tipo de método de introducción hay que mostrar al usuario. El nodo a inspeccionar en cuestión es ***DatoContrasteDataType*** y más concretamente un nodo hijo de este y de nombre ***tipo***. El valor de este nodo nos proporciona el tipo de dato de contraste a mostrar. Los cuatro tipos de datos de contraste son los que se enumeran a continuación:

- En caso que el valor del nodo ***tipo*** sea "**1**", el dato de contraste a mostrar debe posibilitar la introducción de un dato de tipo alfanumérico.
- En caso que el valor del nodo ***tipo*** sea "**2**", el dato de contraste a mostrar debe posibilitar la introducción de un dato de tipo numérico.
- En caso que el valor del nodo ***tipo*** sea "**3**", el dato de contraste a mostrar debe posibilitar la introducción de un dato de tipo fecha.
- En caso que el valor del nodo ***tipo*** sea "**4**", el dato a mostrar será de tipo imagen, por lo que será necesario consultar el nodo ***url,*** hijo de ***DatoContrasteDataType*** para mostrar la imagen.

La representación visual del dato de contraste depende de los datos que requiera, por lo que es necesario consultar el modo de funcionamiento de cada dato de contraste.

6.2.1 Evento Submit

Con independencia de la representación visual de la ventana en la operación de autenticación, el formulario que se envía al servidor de aplicaciones de pasarela debe contener los siguientes parámetros:

id_transaction: Identificador de la transacción en curso. Este parámetro viene indicado en el XML de datos, más concretamente en el atributo *Id* del nodo *TransactionDetails.*

pin: contraseña del certificado del usuario.

sfdaValue0: valor para el segundo factor de autenticación. En caso de que el segundo factor de autenticación requiera de más de un valor, habría que mostrar más de una caja de texto para que el usuario pueda introducir cada valor y los identificadores de cada una de esas cajas deben ser sfdaValue1, sfdaValue2…

datocontraste: valor introducido por el usuario para el dato de contraste.

6.2.2 Presentación de textos y posibles errores

Para encontrar posibles errores que se puedan producir durante la operación de autenticación, es necesario inspeccionar el nodo ***errorCode***, hijo del nodo ***TransactionResponse***. En la plantilla que se proporciona con el producto, el mensaje que describe cada uno de los errores se encuentra en el fichero de textos que se proporciona en la entrada *I18N_APP.* Con el producto se proporciona un fichero de textos de nombre *GateWayTexts.properties* que incluye los textos utilizados y los mensajes asociados a los posibles errores que se puedan encontrar. Los códigos de error en la operación de autenticación comienzan por **AUTHEXXX.**

6.3 Plantilla utilizada en la operación de toma de decisiones

Para las operaciones de toma de decisiones se muestra una ventana preguntando al usuario si desea renovar un certificado que está cercano a caducar o preguntando si desea renovar la contraseña. Para ello la representación gráfica de la ventana que se muestra al usuario se guía por los siguientes nodos:

TextForUser: texto a mostrar al usuario en la ventana de decisión.

RequestForUser: pregunta que se realiza al usuario.

ShowYes: indica si se muestra un botón de aceptar.

ShowNo: indica si se muestra un botón de cancelar.