Pasarela de pagos de la Administración Pública Vasca

V2

(Especificaciones Técnicas para las Entidades Financieras)

14 de Febrero de 2016

Indice

1 Introducción

El presente documento recoge las Especificaciones Técnicas para la versión 2 de la Pasarela de Pagos y está destinado a los responsables técnicos de las Entidades Financieras que prestan servicios de pago en la misma.

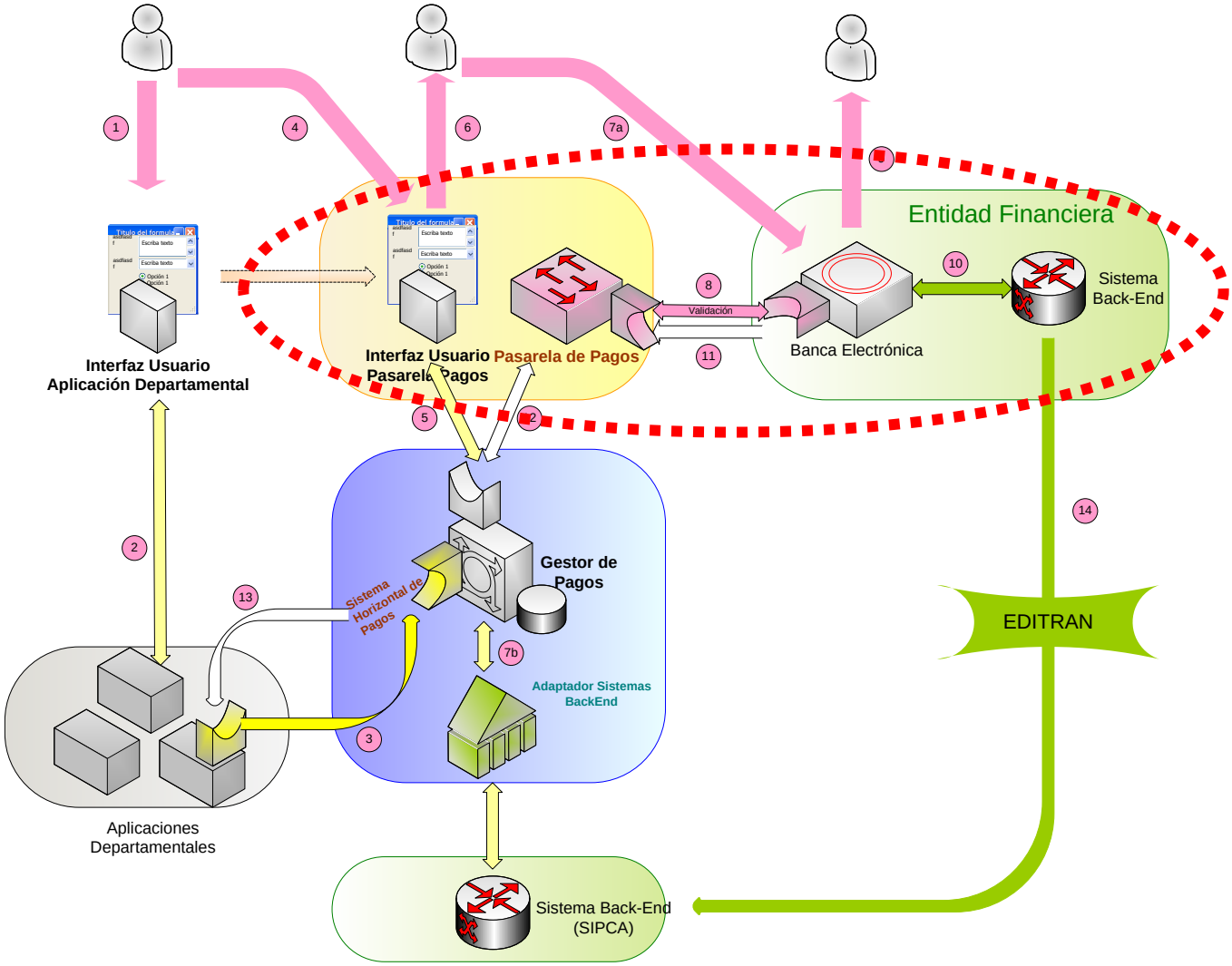
Específicamente se describirán:

1. Interacción de la Pasarela de Pagos con las Entidades Financieras.
2. Definición de datos a intercambiar.
3. Entorno tecnológico.
4. Procedimiento de prueba de la Pasarela de Pagos.

Este documento está vivo en el sentido que recogerá todas las modificaciones que se puedan hacer a los XML intercambiados o a la operativa del istema. Sin embargo, hay que señalar que se intentarán eliminar o minimizar en la medida de lo posible los cambios o modificaciones.

2 Proceso de Pago

Internamente, la Pasarela de Pagos se estructura según el siguiente esquema general:



En el esquema aparecen todas las piezas que componen la Pasarela de Pagos, sin embargo, de cara a la Integración con las Entidades Financieras, únicamente entran en juego las señaladas en un círculo **rojo**.

En la siguiente tabla, se describe cada uno de los pasos del pago señalados con un número en el gráfico.

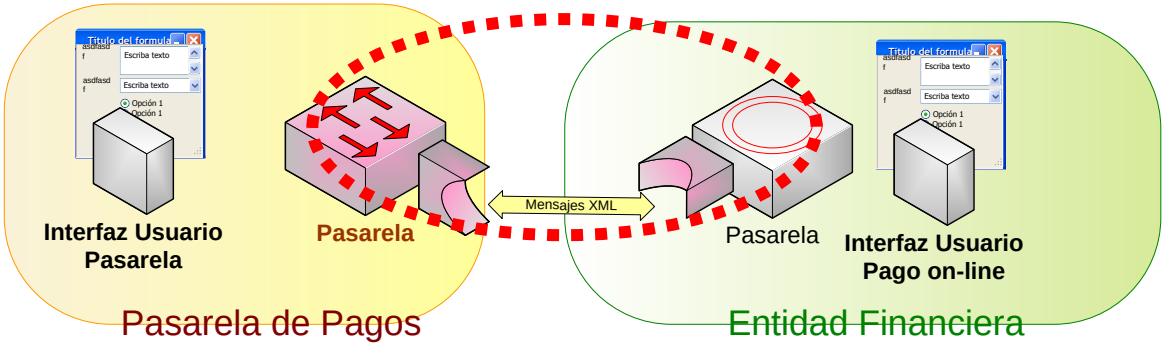
NOTA: *Se está tomando el caso más general del pago desde una aplicación departamental. El pago directo en la Pasarela, introduciendo los datos de la liquidación es una simplificación del anterior*.

Paso	Fase	Descripción
1	Tramitación	Un ciudadano/a tramita utilizando una aplicación departamental, en uno de cuyos pasos es necesario un pago.
2	Generación del Pago (*Petición de Pago*)	La lógica interna de la aplicación **genera un pago** componiendo un mensaje XML (*Petición de* Pago) con todos los datos necesarios.
3	Envío de la Petición de Pago	La aplicación departamental envía el pago en forma de ***Petición de Pago*** al ***Gestor de Pagos*** que se encarga de validarla.
4	Redirección al interfaz de usuario de pago	Con la Petición de Pago validada (es correcta), la aplicación departamental redirige el navegador del ciudadano/a al interfaz de usuario de la Pasarela de Pagos, indicando como parámetro únicamente el identificador del pago que se va a realizar.
5	Recuperación de la Petición de Pago y configuración del interfaz	En base al identificador del pago recibido, el *Gestor de Pagos* recupera todos los datos que se enviaron en el ***paso 3*** y compone el interfaz de usuario de pago. El interfaz de usuario que se muestra depende en gran medida de los datos de la Petición de Pago (paso 3) ya que en esta se indican datos como: → Formas de pago permitidas (on-line / off-line) → Entidades Financieras en las cuales es posible realizar este pago concreto
6	Interacción del ciudadano/a con la Pasarela de Pagos	En función de las opciones disponibles, utilizando la interfaz de usuario, el ciudadano podrá: → Obtener una liquidación para realizar el pago en una ventanilla o cajero de una Entidad Financiera. → Seleccionar una Entidad Financiera para realizar el pago on-line.
7a	Redirección del Ciudadano a la Banca Electrónica de la Entidad Financiera	Si se decide por el pago on-line en una determinada Entidad Financiera, la Pasarela de Pagos redirige al ciudadano/a a la web de pago on-line de dicha Entidad Financiera. La redirección se hace utilizando el navegador de Internet que envía un **POST** a la web de pago on-line con un mensaje XML que contiene los datos del pago.
7b	Consolidación de la Petición de Pago en el sistema back-end de la Administración correspondiente	Tanto si el usuario decide imprimir una liquidación para su pago off-line (ventanilla) como si decide pagar on-line, hay que consolidar la petición de pago en el sistema back-end de pagos para que posteriormente pueda ser "casado" con los cobros recibidos de la Entidad Financiera (*paso 14*) NOTA: Este paso únicamente se realiza si se indica así en la Petición de Pago. En el caso del Gobierno Vasco el sistema back-end es SIPCA (*Sistema de Cobros y Pagos de la Administración*)
8	Validación del Pago	Ante la recepción de un mensaje XML de un ciudadano/a que desea realizar un pago on-line, la Entidad Financiera valida que dicho mensaje XML procede de la Pasarela de Pagos y que además **no** ha sido alterado. Para ello, envía un mensaje de validación a la Pasarela de Pagos.

9	Pago on-line	En función de los medios de pago ofrecidos en la web de la Entidad Financiera, el usuario realiza el pago.
10	Consolidación back-end Entidad Financiera	La Entidad Financiera consolida la transacción de pago en su sistema back-end. Posteriormente, y en función de los acuerdos individuales de intercambio de datos con cada una de las administraciones, los datos de los cobros recibidos se enviarán a la Administración correspondiente (*paso 14*). En este envío de cobros consolidados, la Pasarela de Pagos es transparente.
11	Resultado del Pago	Tanto si el pago ha ido correctamente como si no ha sido así, la Entidad Financiera enviará un mensaje a la Pasarela de Pagos indicando el resultado del pago.
12	Consolidación del Resultado de Pago	La Pasarela de Pagos envía el resultado al *Gestor de Pagos,* que *en* función de dicho resultado actualiza el estado del pago, aunque **no lo pasa al sistema back-end** ya que los pagos realizados son enviados por las Entidades Financieras periódicamente en función de acuerdos individuales con las Administraciones (*paso 14*)
13	Envío a la aplicación departamental del resultado del pago	El Gestor de Pagos, si la aplicación así lo solicitó en el mensaje de Petición de Pago, le envía la respuesta del pago.
14	Consolidación de la Petición de Pago en el sistema back-end de la Administración correspondiente	Las Entidades Financieras periódicamente y en función de acuerdos individuales con las Administraciones reciben los cobros recibidos que se consolidan en el sistema back-end con las liquidaciones emitidas (**paso 7b**)

3 Intercambio de Mensajes

Haciendo zoom en la comunicación entre Pasarela de Pagos y Entidades Financieras, el esquema es:



Como se observa en la figura, funcionalmente, en ambos lados existe un módulo que actúa de **pasarela** (señalado en rojo en la figura). Abstrayendo del medio de comunicación, protocolos, etc, se puede decir que en ambos lados se expone un interfaz programático que se resume en las siguientes tablas:

Del lado de la perteneciente a las **Entidades Financieras**, la Pasarela de la Administración ve las siguientes funciones lógicas, o lo que es lo mismo, las Entidades Financieras "exponen":

Interfaz Entidades Financieras	
Operación	**Descripción**
Recibir Pago	Recibe un mensaje XML con los datos de un pago para su realización on-line.
Consultar Estado Pago	Permite a la Pasarela de Pagos de la Administración enviar un mensaje XML con los datos necesarios para consultar el estado de un pago (si ha sido realizado o no). La consulta de un pago se puede hacer a través del CSB o del NRC del mismo.

Del lado perteneciente a la **Pasarela de Pagos**, las Entidades Financieras ven las siguientes funciones lógicas, o lo que es lo mismo, la Pasarela de la Administración "expone":

Interfaz Pasarela de Pagos	
Operación	**Descripción**
Obtener Pago	Permite a las Entidades Financieras obtener el XML correspondiente a un determinado identificador de pago
Validar Pago	Permite a las Entidades Financieras enviar un mensaje XML con los datos de un pago para validar su procedencia y validez.
Validar Petición de Estado de un Pago	Permite a las Entidades Financieras enviar un mensaje XML con los datos de una solicitud del estado de un pago para validar su procedencia y validez.
Recibir Resultado de Pago	Permite a las Entidades Financieras enviar un mensaje XML con el resultado de un pago on-line

IMPORTANTE

- Todas las operaciones anteriores son **síncronas** (petición – respuesta inmediata).
- Estos interfaces son expuestos por la Administración y Entidades Financieras en base a **servicios accesibles vía web**, aunque en un futuro serán también ofrecidos como servicios web (WebServices).
- Para facilitar esta convergencia futura hacia WebServices se fijará una nomenclatura de acceso a los servicios actuales, accesibles únicamente por web (ver siguiente punto). Esta nomenclatura de acceso a los servicios **no es de obligado cumplimiento**, aunque si muy recomendable su utilización por parte de todas las Entidades Financieras.
- Los servicios intercambiarán XML que **son de obligado cumplimiento** por todas las Entidades Financieras y Administraciones.

3.1 Nomenclatura de las URLs de Acceso a los Servicios

Para facilitar futuros desarrollos tanto por parte de las Entidades Financieras como por parte de las Administraciones, es importante estandarizar una nomenclatura común para el acceso a los servicios vía web.

De esta forma, como criterio general a la hora de hacer accesibles los interfaces vía web, las URLs seguirán la siguiente **nomenclatura:**

http(s)://sitio:puerto/[URL del servicio]	URL de acceso al servicio de Pasarela.

Ejemplos:
https://www.ef.com/xWar/yServlet
https://www.ef.com/pasarela/x.asp
https://www.ef.com/x.cgi

A estos servicios web se les pasarán una serie de **parámetros normalizados** que identifican la **función** requerida en la forma:

http://[urlServicio]?module=X&function=Y&[resto de parametros]

En todos los casos, se devolverá el resultado en formato XML.

En la siguiente tabla se detallan cada uno de estos parámetros:

module	Módulo de la pasarela

Descripción	→ Permite dividir la pasarela de la Administración o Entidades Financieras en módulos (si se considera necesario) para agrupar funciones lógicas.
Valor	→ Un nombre identificador del módulo de la pasarela donde se encuentran las funciones.
Ejemplo	→ *module="pasarela"*

function

Función/procedimiento específico dentro del módulo correspondiente

Descripción	→ Identifica una funcionalidad invocable remotamente
Valor	→ Nombre identificador de la función. En base a las funciones de las interfaces para la Pasarela de la Administración y Entidades Financieras expuestas justo antes, los nombres serán: **Pasarela de la Administración** ▪ ***getPayment***: Obtener los datos de un pago (XML) a partir de su id ▪ ***validatePaymentData***: Validar un XML de pago ▪ ***validatePaymentQuery***: Validar una solicitud de estado de un pago ▪ ***setPaymentResult***: Establecer el resultado de un pago **Pasarela de la Entidad Financiera** ▪ ***setPaymentData***: Crear un pago (comenzar un pago) ▪ ***getPaymentStateDataByCSB***: Obtener los datos (estado) de un pago identificado por su CSB ▪ ***getPaymentStateDataByNRC***: Obtener los datos (estado) de un pago identificado por su NRC
Ejemplo	→ *function="getPaymentData"*

Otros parámetros

Los nombres de los parámetros enviados al servicio accesible vía web pueden ser:

Parámetro	Descripción	Valor
paymentData	XML del pago	
paymentResult	XML del resultado del pago	
presentationData	XML con datos de presentación en pantalla	
protocolData	XML con datos de protocolo (intercambiados entre servidores)	
csb	Ráfaga bancaria del pago	
nrc	Número asignado por le Entidad Financiera en la que se realiza un pago	

Importante:

Adicionalmente y en base a las necesidades específicas de cada pasarela (Administración o Entidad Financiera), podrán existir aquellos parámetros que cada uno considere necesarios.

Los métodos anteriores envían parámetros de entrada y devuelven parámetros de salida en forma de **estructuras XML** que se definen en el punto 5 (Estructuras XML).

A continuación, en la siguiente tabla se detallan cada una de las funciones:

4 Funciones de los Interfaces de Pago

4.1 Resumen del Interfaz Expuesto por la Pasarela de la Administración

Expone funciones llamables por las Pasarelas de las Entidades Financieras.

Funcionalidad	**Módulo**	**Función**	**Parámetros**	**Respuesta**
Obtener los datos de un pago desde la Pasarela de Pagos de la Administración	*IF*	*getPayment*	paymentId: Identificador del pago (**opcional**) protocolData: Datos de protocolo de la llamada a función.	Devuelve un mensaje XML con los datos del pago: *estructura* ***PaymentData*** dentro de la *estructura* ***OperationResult***
Validar los datos de un pago recibidos en una Entidad Financiera contra la Pasarela de Pagos de la Administración	*IF*	*validatePaymentData*	paymentData: Datos del pago en formato XML protocolData: Datos de protocolo de la llamada función.	Devuelve un mensaje XML con el resultado de la validación: *estructura* ***OperationResult***
Validar los datos de una solicitud de estado de un pago recibida por una Entidad Financiera contra la Pasarela de Pagos de la Administración	*IF*	*validatePaymentQuery*	paymentQueryData: Datos que permiten identificar el pago protocolData: Datos de protocolo de la llamada función.	Devuelve un mensaje XML con los datos del Pago si el resultado de la validación es correcto: estructura ***Pago*** dentro de la *estructura* ***OperationResult***
Establece el resultado de un pago en la Pasarela de Pagos de la Administración	*IF*	*setPaymentResult*	paymentResult: Datos del resultado de los pagos en formato XML protocolData: Datos de protocolo de la llamada a función.	Devuelve un mensaje XML con el resultado de la operación: estructura ***OperationResult***.

4.2 Resumen del Interfaz Expuesto por la Pasarela de las Entidades Financieras

Expone funciones llamables por la Pasarela de la Administración.

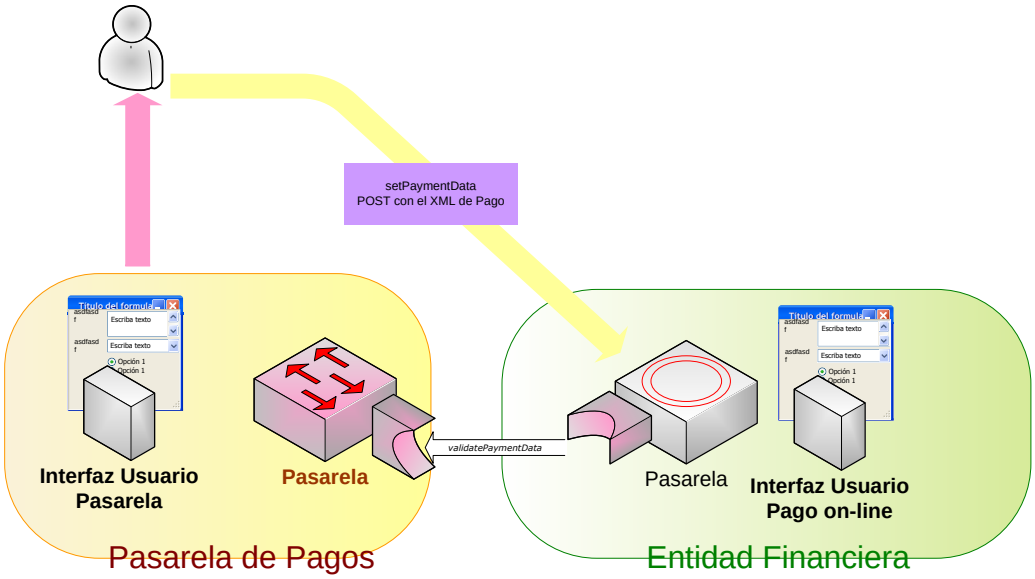
Funcionalidad	**Módulo**	**Función**	**Parámetros**		**Respuesta**
Establecer los datos de un pago y comenzar el pago on-line	*EF*	*setPaymentData*	paymentData: presentationData: protocolData:	Datos del pago en formato XML Datos de presentación en pantalla de los pagos Datos de protocolo de la llamada función	Devuelve un XML con el resultado de la operación: *estructura* ***operationResult****.*
Obtener los datos de un pago (estado)	*EF*	*getPaymentStateDataByCSB*	csb:	Ráfaga en el cuadernillo 57 o 60 del pago	Devuelve un mensaje XML con el estado de pago: *la estructura* ***paymentStateData*** dentro de la *estructura* ***operationResult***
Obtener los datos de un pago (estado)	*EF*	*getPaymentStateDataByNRC*	nrc:	NRC del pago asignado por la Entidad Financiera donde se ha realizado	Devuelve un mensaje XML con el estado de pago: *la estructura* ***paymentStateData*** dentro de la *estructura* ***operationResult***

A continuación se detalla cada una de las funciones de los interfaces anteriores:

4.3 Interfaz Expuesto por las Entidades Financieras

4.3.1 setPaymentData: Recepción de un Pago

Cuando un ciudadano/a desea realizar un pago de forma on-line en una Entidad Financiera, la Pasarela de Pagos utilizará el navegador como intermediario para enviar los datos del pago. Para esto, redirigirá el navegador a la web de la Entidad Financiera haciendo POST del mensaje XML con los datos a una URL concreta que espera la recepción de estos mensajes de pago:



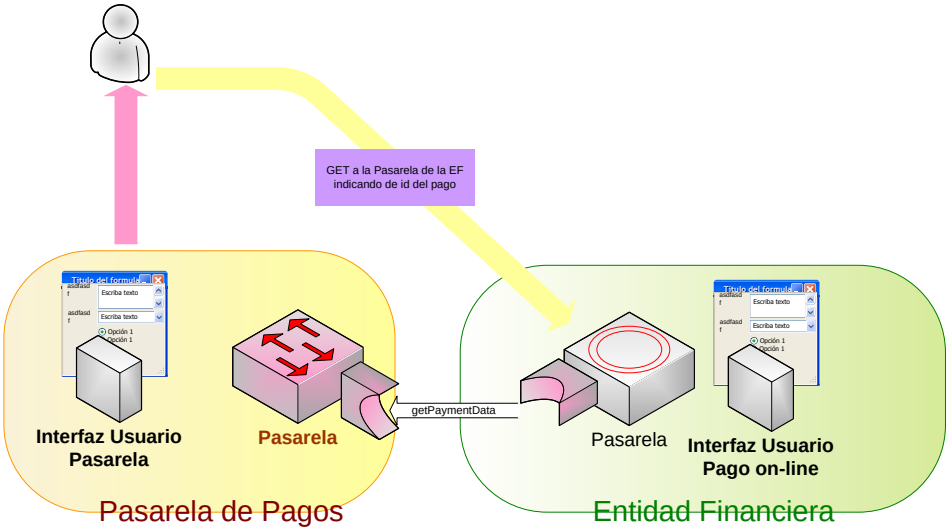
Origen	**Navegador** del Usuario
Destino	**Servicio** de la Pasarela de la Entidad Financiera
Método HTTP	POST
Función	module=EF Function=*setPaymentData* *Parámetros:* paymentData: Datos del pago en formato XML presentationData: Datos de presentación del pago en pantalla o justificantes de pago protocolData Datos de protocolo
Respuesta	En la llamada a este método desde el navegador de usuario la Entidad Financiera **no devolverá un XML con el resultado de la operación**, sino que **redirigirá al usuario a la pantalla de pago on-line**.

Ejemplo	Llamada vía **POST**: http://www.ef.com/[urlServicio]?**module**=EF& **function**= *setPaymentData*& **paymentData**=[XML del pago]& **presentationData**=[XMLpresent]& **protocolData**=[XML protocolo]& [otros parametros requeridos por la EF]

Después de la recepción de un nuevo pago, la Entidad Financiera, tal y como se ha descrito en el punto 2 -Proceso de Pago-, debe solicitar una validación del pago (ver punto 4.4.2) **antes** de mostrar al usuario el interfaz de usuario de pago on-line.

NOTA:

Para **pagos sencillos**, un funcionamiento alternativo y opcional al descrito anteriormente es el que se indica en la siguiente figura:



Como se ve en la figura, en lugar de enviar **directamente** el XML del pago desde el navegador del usuario, únicamente se pasa el identificador del pago a la Pasarela de la Entidad Financiera y esta se encarga de preguntar a la Pasarela de la Administración por los datos del pago (XML): función *getPaymentData*

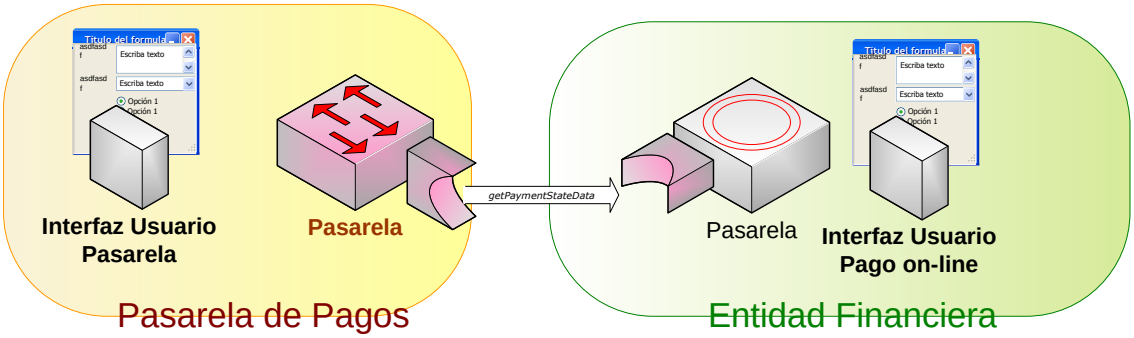
En este caso, **no es necesaria la validación** ya que los datos devueltos por la pasarela seguro que son "buenos".

Por el contrario, este procedimiento tiene el "inconveniente" de que las Entidades Financieras han de mantener las URLs de la Pasarela de Pagos en ficheros de configuración, en lugar de obtenerlas desde el XML de Pago recibido (datos de protocolo).

Para **pagos múltiples** esta solución es difícil de implementar ya que la Pasarela de la Administración **no conserva internamente los lotes**, así que habría que validar cada uno de los pagos del lote **independientemente**.

4.3.2 getPaymentStateDataByXXX: Consulta del Estado de un Pago

La consulta del estado de un pago es solicitada por la Pasarela de la Administración a la Pasarela de la Entidad Financiera utilizando un mensaje XML de consulta de pago que es enviado vía POST:

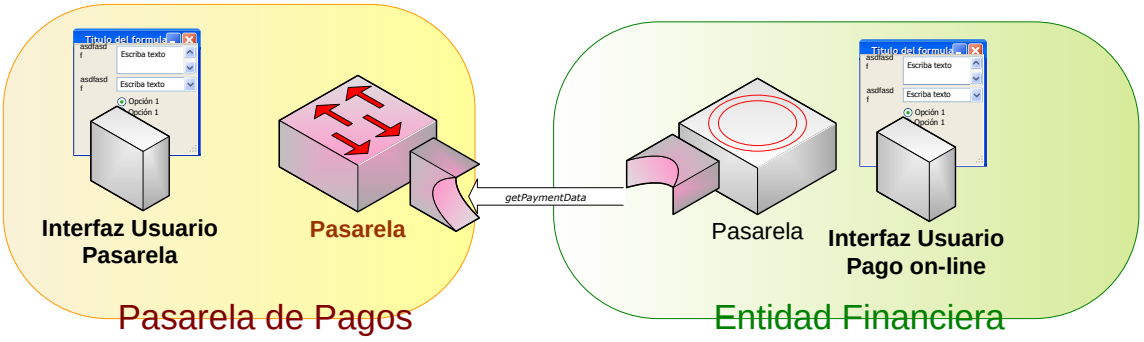


Origen	**Servicio** Pasarela de Pagos de la Administración
Destino	**Servicio** Pasarela de la Entidad Financiera
Método HTTP	POST
Función	module=EF function=*getPaymentStateDataByCSB* *Parámetros:* csb: Código que identifica el pago. function=*getPaymentStateDataByNRC* *Parámetros:* nrc: NRC del pago asignado por el banco.
Respuesta	La Entidad Financiera devuelve un mensaje XML con el resultado de la operación: estructura ***PaymentStateData*** dentro de la estructura ***OperationResult***.
Ejemplo	Llamada vía **POST**: http://www.ef.com/[urlServicio]?**module**=EF& **function**= *getPaymentStateDataByXXX*& **csb**=[CSB]& **nrc**=[NRC]& **protocolData**=[XML protocolo]& [otros parametros requeridos por la EF]

4.4 *Interfaz Expuesto por la Pasarela de Pagos*

4.4.1 *getPayment:* Consulta de los Datos de un Pago

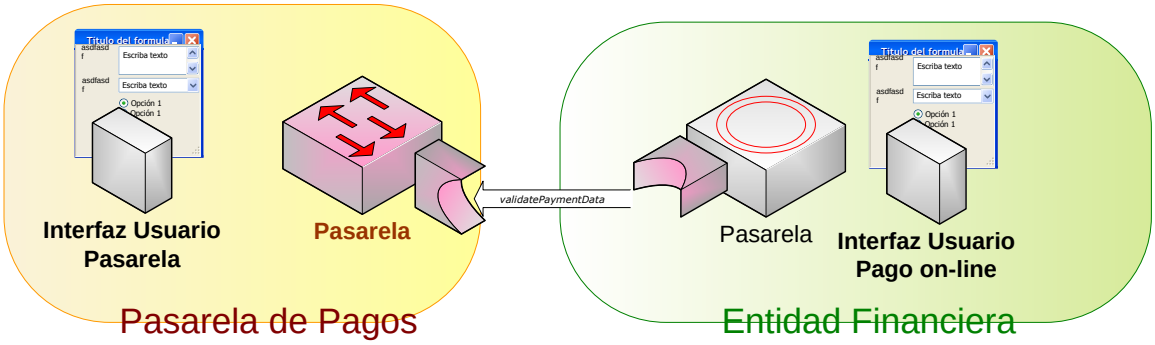
La consulta de los datos de un pago es solicitada por una Entidad Financiera a la Pasarela de Pagos de la Administración utilizando una llamada vía POST o GET en la que se identifica el pago del cual se requieren los datos:



Origen	**Servicio** Pasarela de la Entidad Financiera
Destino	**Servicio** Pasarela de la Administración
Método HTTP	POST / GET
Función	module=IF Function=*getPayment* *Parámetros:* paymentId: Identificador del pago (**opcional**) protocolData Datos de protocolo
Respuesta	La Pasarela de Pagos de la Administración devuelve un mensaje XML con el resultado de la operación: *estructura* ***Pago*** dentro de la *estructura* ***OperationResult***
Ejemplo	Llamada vía **POST**: http://www.ef.com/[urlServicio]?**module**=IF& **function**= *getPaymentData*& **paymentId**=[paymentId]& **protocolData**=[XML protocolo]

4.4.2 *validatePaymentData:* Validación de un Pago

La validación de un pago es solicitada por una Entidad Financiera a la Pasarela de Pagos de la Administración utilizando un mensaje XML de petición de validación que es enviado vía POST:



Origen	**Servicio** Pasarela de la Entidad Financiera
Destino	**Servicio** Pasarela de Pagos de la Administración
Método HTTP	POST
Función	module=IF Function=*validatePaymentData* *Parámetros:* paymentData: Datos del pago en formato XML protocolData Datos de protocolo
Respuesta	La Pasarela de Pagos responderá con un mensaje XML con el resultado de la operación: cadena con el resultado (*"true"/"false"*) dentro de la *estructura* ***OperationResult***
Ejemplo	Llamada vía **POST**: http://www.ef.com/[urlServicio]?**module**=if& **function**= *validatePaymentData*& **paymentData**=[XML de Pago]& **protocolData**=[XML protocolo]

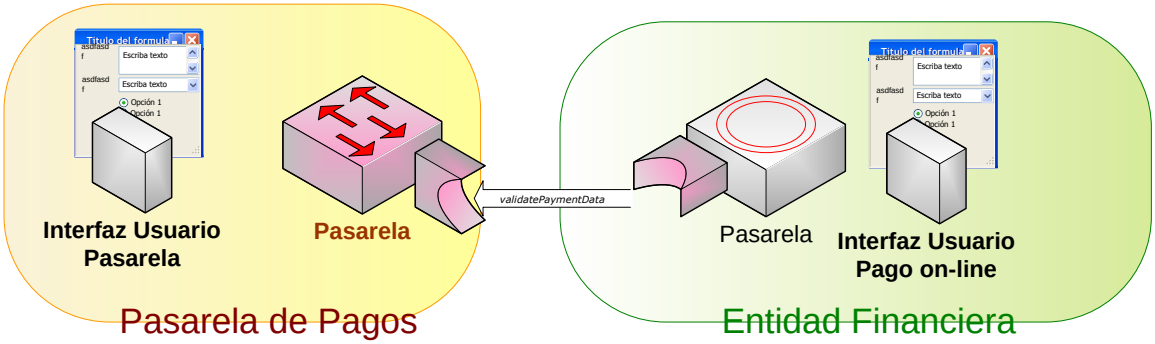
4.4.3 *validatePaymentQuery:* Validación de la solicitud de estado de un Pago

La operativa de la Pasarela de Pagos ofrece la posibilidad de que la Pasarela consulte a las Entidades Financieras por el estado de un pago, actuando como intermediaria para las Aplicaciones Departamentales. Los pagos objeto de estas consultas pueden venir identificados por su CSB o su NRC.

Cuando una Entidad Financiera recibe un consulta acerca del estado de un pago, lo primero que debe hacer es comprobar que esa consulta proviene de la pasarela, de forma que el estado de los pagos no pueda ser consultado por cualquiera.

Además, la Entidad Financiera, podría no disponer de toda la información necesaria para llevar a cabo la recuperación del estado del pago en el momento que recibe la consulta. Por ello, la validación de la consulta devuelve todos los datos del pago en caso de que la validación haya sido correcta.

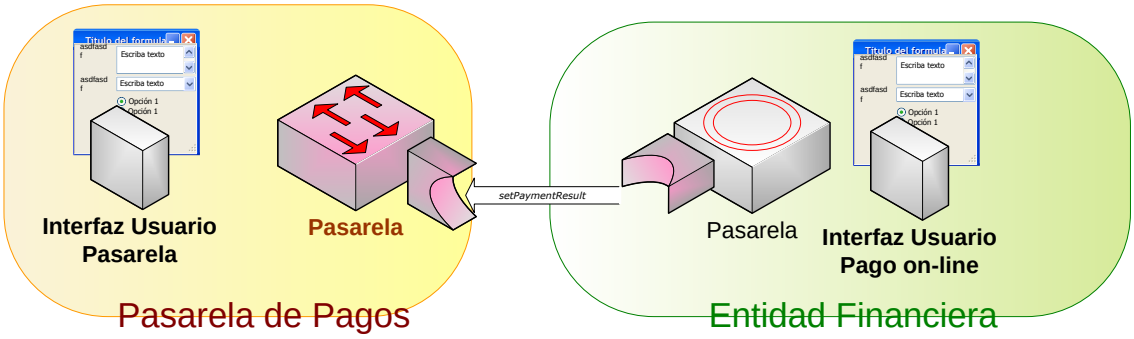
La validación de un pago es solicitada por una Entidad Financiera a la Pasarela de Pagos de la Administración utilizando un mensaje XML de petición de validación de solicitud de estado, que es enviado vía POST:



Origen	**Servicio** Pasarela de la Entidad Financiera
Destino	**Servicio** Pasarela de Pagos de la Administración
Método HTTP	POST
Función	module=IF Function=*validatePaymentQuery* *Parámetros:* paymentQueryData: Datos que identifican el pago del que se consulta el estado protocolData Datos de protocolo
Respuesta	La Pasarela de Pagos responderá con un mensaje XML con el resultado de la operación: *estructura* ***Pago*** dentro de la estructura ***OperationResult*** en caso de que la validación sea correcta***.***
Ejemplo	Llamada vía **POST**: http://www.ef.com/[urlServicio]?**module**=if& **function**= *validatePaymentData*& **paymentQueryData**=[XML de Pago]& **protocolData**=[XML protocolo]

4.4.4 *setPaymentResult:* Recepción del Resultado de un Pago

El resultado de un pago es enviado por la Entidad Financiera a la Pasarela de Pagos de la Administración:



Origen	**Servicio** Pasarela de la Entidad Financiera
Destino	**Servicio** Pasarela de Pagos de la Administración
Método http	POST
Función	module=IF Function=*setPaymentResult* *Parámetros:* paymentStateData: Datos del pago en formato XML protocolData: Datos de protocolo
Respuesta	La Pasarela de Pagos responderá con un mensaje XML de resultado de operación: *estructura* ***OperationResult***.
Ejemplo	Llamada vía **POST**: http://www.ef.com/[urlServicio]?**module**=if& **function**= *setPaymentResult*& **paymentResult**=[XML de Resultado]& **requiresResponse**=true

5 Estructuras XML

En este punto se detallan las estructuras de datos en formato XML que se intercambian entre la Pasarela de la Administración y la Pasarela de las Entidades Financieras.

Las estructuras de datos en formato XML son las siguientes:

paymentData *XML de Pago*	XML enviado por la Pasarela de Pagos y que contiene los datos de uno o varias liquidaciones a realizar on-line en una Entidad Financiera
presentationData *XML con datos de presentación en pantalla*	XML que contiene datos que permiten "pintar" las pantallas y justificantes de pago
protocolData *Datos de protocolo en la llamada a función*	Contiene datos de protocolo en las llamadas a función: urls, identificadores de sesión, etc.
paymentResult *XML de Resultado de Pago*	XML que devuelve la Entidad Financiera y que contiene el resultado de la operación de pago de una o varias liquidaciones.
operationResult *XML de Resultado de una operación genérica de la Pasarela de Pagos*	XML genérico de resultado de una operación.
validationResult *XML de Resultado de una petición de validación de la Pasarela de Pagos*	XML que contiene el resultado de una operación de validación.

El paquete ***p12f.exe.pasarelapagos.objects*** proporciona un conjunto de clases que encapsulan la información que van a manejar las Entidades Financieras.

Estas clases abstraen al programador del manejo de datos XML. Contienen métodos que permiten construir un objeto a partir de un XML pasado como ***String*** (método ***getObject(String XML)***) y obtener el XML que representa la información de uno de estos objetos (método ***toXML()***).

A continuación se presenta la relación entre las estructuras XML y las clases del paquete ***p12f.exe.pasarelapagos.objects***.

5.1 PaymentData.

paymentData	Objeto ***PaymentData***. Esta estructura encapsula uno o varios Pagos para las Entidades Financieras.
pagos	Mapa que contiene uno o varios Pagos.
pago	Objeto ***Pago***.

	Esta estructura contiene todos los datos relacionados con un Pago.
id	Identificador del Pago (***String***).
datosPago	Objeto ***DatosPago***. Esta estructura contiene los datos del Pago.
formato	Formato del Pago (***String***).
validar	Indica si hay que validar o no los dígitos de control 0 = no validar 1 = validación estandar 2 = validación no estandar
cpr	Código de Procedimiento Recaudatorio (***String***).
codigo	Código completo del Pago en el formato 57 ó 60 (***String***).
emisor	Código del emisor (***String***).
referencia	Referencia del Pago (***String***).
tipo	Tipo del Pago o código del tributo (***String***).
periodosPago	Mapa que contiene uno o varios Periodos de Pago.
periodoPago	Objeto ***PeriodoPago***. Esta estructura contiene los datos de un Periodo de Pago.
id	Identificador del Periodo de Pago (***String***). En el objeto ***PeriodoPago*** hay definidas las siguientes constantes: ▪ ***PERIODO_NORMAL***: Periodo normal de formalizacion del pago. ▪ ***PERIODO_VOLUNTARIO***: Periodo voluntario de formalizacion del pago. ▪ ***PERIODO_CON_RECARGO***: Periodo con recargo de formalizacion del pago.
identificacion	Identificación del Pago en este Periodo de Pago (***String***).
importe	Importe del Pago en centeuros en este Periodo de Pago (***long***).
descripcion	Descripción del Periodo de Pago en diferentes idiomas. Se trata de un mapa de ***String*** en el que el identificador de los objetos indica el idioma: ▪ es: castellano ▪ eu: euskera
fechaInicio	Fecha de inicio del Periodo de Pago con formato dd/mm/aa (***String***).
fechaFin	Fecha de Fin del Periodo de Pago con formato dd/mm/aa (***String***).
validarFechaFin	Indica si hay que validar la fecha de fin del Periodo de Pago (***boolean***).
activo	Indica si es el periodo activo (***boolean***).
descripcion	Descripción del Pago en diferentes idiomas.

	Se trata de un mapa de ***String*** en el que el identificador de los objetos indica el idioma: ▪ es: castellano ▪ eu: euskera
conceptos	Mapa que contiene uno o varios Conceptos del Pago.
concepto	Objeto ***Concepto***. Esta estructura contiene los datos de un Concepto de Pago.
numeroLinea	Número de línea del Concepto (***int***).
importe	Importe del Concepto (***long***).
descripcion	Descripción del Concepto de Pago en diferentes idiomas. Se trata de un mapa de ***String*** en el que el identificador de los objetos indica el idioma: ▪ es: castellano ▪ eu: euskera
unidades	Cantidad de elementos iguales que componen el concepto (***int***).
tieneIVARepercutido	Flag que indica si el concepto tiene IVA repercutido (***boolean***).
IVARepercutido	Flag que indica si el IVA es o no repercutido (***boolean***).
baseImponible	Base imponible del precio del concepto (***long***).
importeIVA	Importe del IVA (***long***).
tipoIVA	Tipo del IVA (***long***).
emisor	Objeto ***Emisor.*** Esta estructura contiene los datos del Emisor del Pago.
code	Código identificativo del Emisor (***String***).
cif	CIF del Emisor (***String***).
nombre	Nombre del Emisor en diferentes idiomas. Se trata de un mapa de ***String*** en el que el identificador de los objetos indica el idioma: ▪ es: castellano ▪ eu: euskera
calle	Calle del Emisor (***String***).
municipio	Municipio del Emisor (***String***).
territorio	Territorio del Emisor (***String***).
pais	País del Emisor (***String***).
codigoPostal	Código Postal del Emisor (***String***).
entidadTesorera	Entidad Tesorera del Emisor (***String***).
expediente	Objeto ***Expediente.*** Esta estructura contiene los datos del Expediente al que pertenece el Pago.
codigo	Código del Expediente (***String***).
familia	Familia del Expediente (***String***).
descripcion	Descripción del Expediente del Pago en diferentes idiomas. Se trata de un mapa de ***String*** en el que el

	identificador de los objetos indica el idioma: ▪ es: castellano ▪ eu: euskera
tercero	Objeto ***Tercero.*** Esta estructura contiene los datos del Tercero al que imputar el Pago.
dniNif	DNI/NIF del Tercero (***String***).
razonSocial	Razón Social del Tercero (***String***).
calle	Calle del Tercero (***String***).
municipio	Municipio del Tercero (***String***).
territorio	Territorio del Tercero (***String***).
pais	País del Tercero (***String***).
codigoPostal	CódigoPostal del Tercero (***String***).
imagenes	Mapa que contiene una o varias Imágenes del Pago.
imagen	Objeto ***Imagen.*** Esta estructura contiene los datos de una Imagen del Pago.
id	Identificador de la imagen (***String***).
alt	Texto de la imagen en diferentes idiomas. Se trata de un mapa de ***String*** en el que el identificador de los objetos indica el idioma: ▪ es: castellano ▪ eu: euskera
url	URL de la imagen (***String***).
bin	Imagen en formato binario codificada en base 64 (***String***).
mensajes	Mapa que contiene uno o varios Mensajes del Pago.
mensaje	Objeto ***Mensaje.*** Esta estructura contiene los datos de un Mensaje del Pago.
id	Identificador del Mensaje (***String***).
texto	Texto del Mensaje en diferentes idiomas. Se trata de un mapa de ***String*** en el que el identificador de los objetos indica el idioma: ▪ es: castellano ▪ eu: euskera
domiciliacion	Objeto ***Domiciliacion.*** Esta estructura contiene los datos domiciliación del Pago.
permitir	Indica a las Entidades Financieras si se permite la domiciliación de este pago (***boolean***).
tpvVirtual	Objeto ***TPVVirtual.*** Esta estructura contiene información para el TPV.
codigoComercio	Codigo de comercio del cliente del TPV (***String***).

5.2 PresentationData.

presentationData	Objeto ***PresentationData***. Esta estructura contiene los datos de presentación para la Entidad Financiera.
idioma	Idioma en que se quiere presentar la información (***String***): ▪ es: castellano ▪ eu: euskera
imagenes	Mapa que contiene una o varias Imágenes.
imagen	Objeto ***Imagen.*** Esta estructura contiene los datos de una Imagen.
id	Identificador de la imagen (***String***).
alt	Texto de la imagen en diferentes idiomas. Se trata de un mapa de ***String*** en el que el identificador de los objetos indica el idioma: ▪ es: castellano ▪ eu: euskera
url	URL de la imagen (***String***).
bin	Imagen en formato binario codificada en base 64 (***String***).
mensajes	Mapa que contiene uno o varios Mensajes del Pago.
mensaje	Objeto ***Mensaje.*** Esta estructura contiene los datos de un Mensaje.
id	Identificador del Mensaje (***String***).
texto	Texto del Mensaje en diferentes idiomas. Se trata de un mapa de ***String*** en el que el identificador de los objetos indica el idioma: es: castellano eu: euskera

5.3 ProtocolData.

protocolData	Objeto ***ProtocolData***. Esta estructura permite intercambiar datos de contexto relativos a la llamada a función.
token	Token de seguridad para autenticar el origen del mensaje (***String***).
responseURL	URL a la que hay que enviar la respuesta a la petición (***String***). **Caso habitual:** En el caso de que este valor no llegue, la respuesta se devolverá como *response* de la *request* original. **Otros casos:** Se puede indicar un valor en este

	parámetro, lo cual hará que la respuesta a la llamada a función (*operationResponse*) se envíe a la URL que se indica.
sourceSessionId	Idntificador de la sesión el el servidor que inicia la llamada (***String***).
destinationSessionId	Identificador de la sesión en el servidor que recibe la llamada (***String***).
timeStamp	Marca de tiempo del envío del mensaje (***String***).
sourceOperationNumber	Numero de operación para el servicio que inicia la llamada (***String***).
urls	Mapa de ***String*** con diferentes URLs necesarias para el procesado de la petición, como: ▪ ***vueltaAdmin***: Utilizada por el banco para devolver al usuario a la Administración que originó el pago. ▪ ***validacionAdmin***: Utilizada por el banco para enviar peticiones de validación de los pagos recibidos. ▪ ***resultadoAdmin***: Utilizada por el banco para devolver a la pasarela el resultado de los pagos solicitados.
url	Objeto ***Url.*** Esta estructura contiene la URL.
id	Identificador de la URL (***String***).
url	URL (***String***).

5.4 PaymentResult.

paymentResult	Objeto ***PaymentResult***. Esta estructura contiene información sobre el resultado de una operación de pago.
paymentStateDatas	Mapa con los indicadores del estado de cada uno de los pagos.
paymentStateData	Objeto ***PaymentStateData***. Esta estructura contiene información sobre el estado de un pago.
id	Identificador del pago (***String***).
datosPago	Objeto ***DatosPago***. Esta estructura contiene los datos del Pago.
formato	Formato del Pago (***String***).
validar	Indica si hay que validar o no los dígitos de control (***boolean***).
cpr	Código de Procedimiento Recaudatorio (***String***).
codigo	Código completo del Pago en el formato 57 ó 60 (***String***).

emisor	Código del emisor (***String***).
referencia	Referencia del Pago (***String***).
estado	Objeto ***Estado***. Esta estructura contiene los datos del Estado del Pago.
codigo	Codigo que indica el estado del pago (***String***). Puede ser: ▪ **REGISTRADO**: Se ha introducido el pago en el sistema. ▪ **EMITIDA_LIQUIDACION**: Se ha impreso una orden de pago correspondiente a este pago. ▪ **ENVIADO_ENTIDAD**: Se ha dirigido el pago a una entidad financiera. ▪ **ANULADO**: Se ha anulado el pago. ▪ **PAGADO**: El pago ha sido pagado correctamente en una entidad financiera. ▪ **ERROR_PAGO**: Se ha producido un error al tratar de realizar el pago en una entidad financiera.
fechaPago	Forma parte del NRC devuelto por la Entidad Financiera. Indica la fecha en que se realizo el pago en el formato **ddmmyyyy** (***String***).
horaPago	Forma parte del NRC devuelto por la Entidad Financiera. Indica la hora en que se realizo el pago en el formato **hhmmss** (***String***).
razonError	Código que indica la razón por el cual el pago no pudo ser realizado (***String***).
importe	Importe del pago realizado (***String***).
entidad	Entidad en la que se ha realizado el pago (***String***).
oficina	Oficina en la que se ha realizado el pago (***String***).
numeroOperacion	Código que la Entidad Financiera asigna para formar parte del NRC. Es el número de operación para la Entidad Financiera (***String***).
nrc	NRC completo del pago tal y como lo genera la Entidad Financiera: - Identificador de la liquidación + Firma (22 caracteres en total) (***String***).
mensajes	Mapa que contiene uno o varios Mensajes de estado del Pago.
mensaje	Objeto ***Mensaje.*** Esta estructura contiene los datos de un

	Mensaje.
id	Identificador del Mensaje (***String***).
texto	Texto del Mensaje en diferentes idiomas. Se trata de un mapa de ***String*** en el que el identificador de los objetos indica el idioma: ▪ es: castellano ▪ eu: euskera

5.5 OperationResult.

operationResult	Objeto ***OperationResult***. Esta estructura contiene información sobre el resultado de una llamada.
resultado	Objeto ***Resultado.*** Esta estructura contiene los datos del Resultado de una llamada.
resultadoOK	Indica si la operación se ha realizado con éxito o ha habido algún error durante la validación (***boolean***).
returnValue	Contiene el resultado de la operación. Puede ser cualquier valor, incluido un XML. Por ejemplo, el XML del ***ProtocolData*** (***String***).
returnCode	Código numérico del resultado de la operación (***String***).
mensajes	Mapa que contiene uno o varios Mensajes.
mensaje	Objeto ***Mensaje.*** Esta estructura contiene los datos de un Mensaje.
id	Identificador del Mensaje (***String***).
texto	Texto del Mensaje en diferentes idiomas. Se trata de un mapa de ***String*** en el que el identificador de los objetos indica el idioma: ▪ es: castellano ▪ eu: euskera
operationData	Objeto ***OperationData.*** Esta estructura contiene los datos sobre la llamada.
module	Módulo llamado (***String***).
function	Función u operación llamada (***String***).
parameters	Mapa que contiene los Parámetros de la función llamada.
parameter	Objeto ***Parameter.*** Esta estructura contiene los datos de un Parámetro de la llamada.
name	Nombre del Parámetro llamado (***String***).
value	Valor del Parámetro. Puede ser cualquier tipo de datos, incluido un XML (***String***).

5.6 ValidationResult.

validationResult	Objeto ***ValidationResult***. Esta estructura contiene información sobre el resultado de una operación de validación.
resultado	Indica si la validación se ha realizado con éxito o no (***boolean***).
pago	Objeto ***Pago***. Esta estructura contiene todos los datos relacionados con un Pago (Ver su contenido en anteriores apartados).
mensajes	Mapa que contiene uno o varios Mensajes.
mensaje	Objeto ***Mensaje.*** Esta estructura contiene los datos de un Mensaje.
id	Identificador del Mensaje (***String***).
texto	Texto del Mensaje en diferentes idiomas. Se trata de un mapa de ***String*** en el que el identificador de los objetos indica el idioma: ▪ es: castellano ▪ eu: euskera

6 Ejemplos de los XML

6.1 PaymentData: Datos de Pago

```
<paymentData>
    <pagos>
        <pago id='90507048330015141111111111111111111100000090500'>
            <domiciliacion>
                <permitir>true</permitir>
            </domiciliacion>
            <mensajes>
                <mensaje id='2'>
                    <texto>
                        <es>Mensaje Configuracion.....</es>
                        <eu>Beste leku batzutan.....</eu>
                    </texto>
                </mensaje>
                <mensaje id='1'>
                    <texto>
                        <es>Mensaje Configuracion.....</es>
                        <eu>beste leku batzutan.....</eu>
                    </texto>
                </mensaje>
            </mensajes>
            <imagenes>
                <imagen id='logoEmisor'>
                    <url>http://desaunix01.ej-
gv/testpago/newPaymentGateway/imagenes/euskadi.gif</url>
                    <alt>
                        <es>Euskadi</es>
                        <eu>Euskadi_eu</eu>
                    </alt>
                </imagen>
            </imagenes>
            <datosPago>
                <cpr>9050794</cpr>
                <referencia>1111111111111</referencia>
                <formato>507</formato>
                <emisor>04833001-514</emisor>
                <periodosPago>
                    <periodoPago id='periodoNormal'>
                        <fechaFin>111111</fechaFin>
                        <validarFechaFin>false</validarFechaFin>
                        <descripcion>
                            <es>Periodo de pago normal.</es>
                        </descripcion>
                        <identificacion>111111</identificacion>
                        <importe>9050</importe>
                        <activo>true</activo>
                    </periodoPago>
                </periodosPago>
                <validar>true</validar>
                <codigo>90507048330015141111111111111111111100000090500</codigo>
            </datosPago>
            <descripcion>
                <es>Tipo de pago 0483300514507</es>
                <eu>Ordainketa mota 0483300514507</eu>
            </descripcion>
            <emisor>
                <cif>04833001</cif>
                <code>04833001</code>
                <nombre>
                    <es>Administracion de prueba 2</es>
                    <eu>Probazko Administrazioa 2</eu>
                </nombre>
                <territorio>Alava</territorio>
                <municipio>Vitoria</municipio>
                <codigoPostal>30000</codigoPostal>
```

```
                <pais>Spain</pais>
                <calle>Avda. Mediterraneo</calle>
            </emisor>
        </pago>
    </pagos>
</paymentData>
```

6.2 PresentationData: Datos de Presentacion

```
<presentationData>
    <idioma>es</idioma>
</presentationData>
```

6.3 ProtocolData: Datos de Protocolo

```
<protocolData>
    <token>h45456kj34456g4ywfs879fsfg</token>
    <responseURL></responseURL>
    <sourceSessionId>QnsGctGNyXtWsLiZb37MiKedLYTZWdQeNHqib42QOLWVjh6B2HWX|-
3679191607778368118/-1062722781/6/7022/7022/7023/7023/7022/-
1|1115358971045</sourceSessionId>
    <destinationSessionId>QGFfh43nsf67df456Gcg5fgt454fgr355654GN67yXt877WsL5df|-
4556576788789890990/-4345343434/4/7022/7022/7023/7023/7022/-
1|4545565767676</destinationSessionId>
    <timeStamp>06/05/2005 [07:55:47:338]</timeStamp>
    <sourceOperationNumber>0001</sourceOperationNumber>
    <urls>
        <url
id="vueltaAdmin">http://www.adminX.com/m18/solicitudes/m18ConfirmacionPago.jsp?MODO=
MODO_ALTA&pagoID=048330015071001120051186160505182 2</url>
        <url
id="validacionPago">https://www.pasarelaAdmin.net/p72/p72ppaServlet?module=if</url>
        <url
id="resultadoPago">https://www.pasarelaAdmin.net/p72/p72ppaServlet?module=if</url>
    </urls>
</protocolData>
```

6.4 PaymentResult: Resultado de un Pago

```
<paymentResult>
    <paymentStateDatas>
        <paymentStateData id='90507048330015141111111111111111111100000090500'>
            <datosPago>
                <cpr>9050794</cpr>
                <referencia>1111111111111</referencia>
                <formato>507</formato>
                <emisor>04833001-514</emisor>
                <periodosPago>
                    <periodoPago id='periodoNormal'>
                        <fechaFin>111111</fechaFin>
                        <validarFechaFin>false</validarFechaFin>
                        <descripcion>
                            <es>Periodo de pago normal.</es>
                        </descripcion>
                        <identificacion>111111</identificacion>
                        <importe>9050</importe>
                        <activo>true</activo>
                    </periodoPago>
                </periodosPago>
                <validar>true</validar>
                <codigo>90507048330015141111111111111111111100000090500</codigo>
            </datosPago>
            <estado>
```

```
                <horaPago>100718</horaPago>
                <fechaPago>051005</fechaPago>
                <codigo>04</codigo>
                <numeroOperacion>3475694534657</numeroOperacion>
                <nrc>111111111111165618815</nrc>
            </estado>
        </paymentStateData>
    </paymentStateDatas>
</paymentResult>
```

6.5 OperationResult: Resultado de una operación de Pasarela

```
<operationResult>
    <resultado>
        <resultadoOK>true</resultadoOK>
        <returnValue>true</returnValue>
        <returnCode>1</returnCode>
        <mensajes>
            <mensaje id="1">
                <es>Error al comprobar la existencia del sufijo en el host</es>
                <eu>Errore bat sortu da atzizkiren existentzia hosten
begiratzean</eu>
            </mensaje>
            <mensaje id="2">
                <es>No se puede realizar el pago</es>
                <eu>Ordainketa burutu ezin da</eu>
            </mensaje>
        </mensajes>
    </resultado>
    <operationData>
        <module>if</module>
        <function>validatePayment</function>
        <parameters>
            <parameter>
                <name>paymentId</name>
                <value>90507940483300150710011200511861605050000018220</value>
            </parameter>
            <parameter>
                <name>cpr</name>
                <value>9050794</value>
            </parameter>
            <parameter>
                <name>paymentCode</name>
                <value>90507940483300150710011200511861605050000018220</value>
            </parameter>
            <parameter>
                <name>paymentData</name>
                <value><!-- Aqui se incluye la estructura PaymentData--></value>
            </parameter>
        </parameters>
    </operationData>
</operationResult>
```

6.6 ValidationResult: Resultado de una petición de validación

```
<validationResult>
    <resultado>true</resultado>
    <pago id='90507111111111111111111111111111100000656340'>
        <domiciliacion>
            <permitir>false</permitir>
        </domiciliacion>
        <mensajes>
            <mensaje id='cuerpoJustificante'>
                <texto>
                    <es>Este documento constituye justificante de pago y libera al
deudor de su obligación de pago desde la fecha y por el importe señalados.</es>
                    <eu>Este documento constituye justificante de pago y libera al
deudor de su obligación de pago desde la fecha y por el importe señalados._eu</eu>
                </texto>
            </mensaje>
        </mensajes>
        <imagenes>
            <imagen id='logoEmisor'>
                <url><![CDATA[http://desaunix01.ej-
gv/testpago/newPaymentGateway/imagenes/logo_gv.gif]]></url>
                <alt>
                    <es>Euskadi</es>
                    <eu>Euskadi_eu</eu>
                </alt>
            </imagen>
        </imagenes>
        <datosPago>
            <cpr>9050794</cpr>
            <referencia>1111111111111</referencia>
            <formato>507</formato>
            <emisor>11111111-111</emisor>
            <periodosPago>
                <periodoPago id='periodoNormal'>
                    <validarFechaFin>false</validarFechaFin>
                    <identificacion>111111</identificacion>
                    <importe>65634</importe>
                    <activo>true</activo>
                </periodoPago>
            </periodosPago>
            <validar>true</validar>
            <tipo>111</tipo>
            <codigo>90507111111111111111111111111111100000656340</codigo>
        </datosPago>
        <descripcion>
            <es>Aplicacion de Prueba  CSB 507</es>
            <eu>Aplicacion de Prueba _EU  CSB 507</eu>
        </descripcion>
        <emisor>
            <cif>11111111</cif>
            <nombre>
                <es>Administracion de prueba 3</es>
                <eu>Probazko Administrazioa 3</eu>
            </nombre>
            <territorio>Alava</territorio>
            <municipio>Vitoria</municipio>
            <codigoPostal>30000</codigoPostal>
            <pais>Spain</pais>
            <calle>Avda. Mediterraneo</calle>
        </emisor>
    </pago>
</validationResult>
```

7 SDK de la Pasarela de Pagos.

El SDK de la Pasarela de Pagos son una serie de librerías instalables en los servidores de las Entidades Financieras y que abstraen a las aplicaciones de Banca Electrónica de las complejidades de la Pasarela de Pagos.

En concreto, utilizando estas librerías es posible:

1. Invocar las funciones expuestas por la Pasarela de Pagos desde la aplicación de Banca Electrónica de una Entidad Financiera.
2. Interpretar las estructuras de datos XML.
3. "Pintar" estructuras que muestran la información de los pagos.

Inicialmente estas librerías solamente están disponibles para entornos Java con JDK 1.3/1.4, estando prevista su implementación en entornos Microsoft en un futuro.

En este apartado se pretende mostrar el proceso de desarrollo a seguir en una aplicación de Banca Electrónica para llevar a cabo la integración con la Pasarela de Pagos. Los pasos a seguir son:

1. Construcción de una clase que implemente la interfaz ***p12e.pasarelapagos.base.FinantialOrgFunctions*** que encapsula los métodos con las funciones específicas de acceso al host financiero de la aplicación de Banca electrónica. Esta clase se encargan de:

    - Realizar las validaciones internas sobre los pagos que la Entidad Financiera considere oportunas.
    - Recuperar el estado de un pago en la Entidad Financiera.
    - Redirigir al cliente a la página de inicio de la aplicación de Banca Electrónica de la Entidad Financiera.

2. Implementación del servicio de escucha de peticiones en la Entidad Financiera. Este componente, que puede ser un servlet o una JSP, hará uso de las siguientes partes del API:

    2.1. Clase ***p12e.pasarelapagos.base.EFClass,*** que encapsula la comunicación con la Pasarela de Pagos e implementa parcialmente la lógica de los métodos de la interfaz expuesta por las Entidades Financieras. Los métodos de esta clase se encargan de:

        - Solicitar a la Pasarela de Pagos la validación de los pagos recibidos por la Entidad Financiera.
        - Llamar a los métodos que realizan las validaciones en la Entidad Financiera y redirigen al cliente a la página de inicio de la aplicación de Banca Electrónica de la Entidad Financiera.
        - Devolver a la Pasarela de Pagos el resultado de los pagos realizados en la entidad Financiera.

- Llamar al método que recupera el estado de un pago en la Entidad Financiera y devolver éste estado a la Pasarela de Pagos.

2.2. Clases del paquete ***p12e.exe.pasarelapagos.html.helpers***, que permiten la generación sencilla de estructuras HTML que muestran información sobre los pagos. Estas clases pueden ser utilizadas en cualquier lugar de la aplicación de Banca Electrónica.

Se irán siguiendo uno a uno estos pasos, dando ejemplos de utilización de las librerías del SDK de la Pasarela e Pagos.

7.1 Interfaz FinantialOrgsFunction.

Los métodos definidos en este interfaz se encargarán de añadir la lógica propia a la operación en la Entidad Financiera a los métodos de la interfaz expuesta por las Entidades Financieras.

Esta interfaz define los siguientes métodos, cuya implementación queda a manos de la Entidad Financiera:

7.1.1 doInternalValidations.

Este método permite que la Entidad Financiera haga las validaciones internas que estime necesarias con los pagos recibidos. La signatura de este método es:

```
public boolean doInternalValidations(
    PaymentData paymentData)
    throws FinantialOrgException;
```

Este método recibe como parámetro una estructura *PaymentData* con los pagos recibidos en la Entidad Financiera y devuelve un *boolean* con el resultado de las validaciones realizadas, con valor verdadero si el resultado de la validación es satisfactorio y falso si no lo es.

El tipo de validaciones a realizar son:

1. Comprobación de datos:
    - Dígitos de control.
    - Datos mínimos.
    - Etc.
2. Comprobación de si el emisor es cliente de la Entidad Financiera.
3. Etc.

7.1.2 doGetPaymentStateDataByCSB.

Este método permite que la Entidad Financiera haga las operaciones internas necesarias para recuperar el estado del pago solicitado. La signatura de este método es:

```
public Estado doGetPaymentStateDataByCSB(
    String csb)
    throws GatewayException, FinantialOrgException;
```

Este método recibe una cadena con el CSB del pago, que permitirá identificarlo en los sistemas de la Entidad Financiera.

7.1.3 doGetPaymentStateDataByNRC.

Este método permite que la Entidad Financiera haga las operaciones internas necesarias para recuperar el estado del pago solicitado. La signatura de este método es:

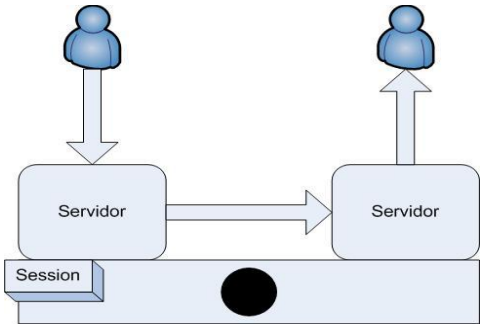
```
public Estado doGetPaymentStateDataByCSB(
    String nrc)
    throws GatewayException, FinantialOrgException;
```

Este método recibe una cadena con el NRC del pago, que permitirá identificarlo en los sistemas de la Entidad Financiera.

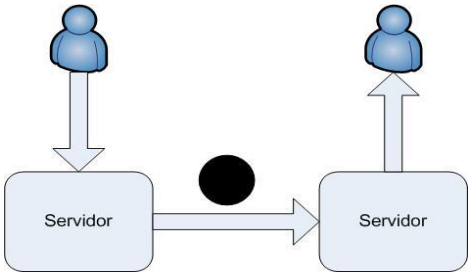
7.1.4 doRedirectClient.

Este método permite que la Entidad Financiera redirija al cliente desde el servlet/JSP de escucha de peticiones a la página de inicio de la aplicación de Banca Electrónica de cualquiera de las tres formas siguientes:

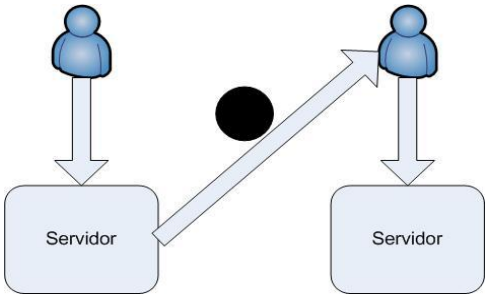
- Redirección de **servidor con sesión en el servidor**, almacenando los objetos de pago como atributos de sesión.



- Redirección de **servidor sin sesión en el servidor**, enviando los objetos de pago como atributos dentro de la petición.



- Redirección de **cliente sin sesión**, enviando los XML de los objetos de pago como parámetros.



La signatura de este método es:

```
public void doRedirectClient(
    HttpServletRequest request,
    HttpServletResponse response,
    PaymentData paymentData,
    PresentationData presentationData);
```

Para la implementación de este método se puede utilizar la clase ***p12e.exe.pasarelaPagos.redirection.ClientRedirector***. Esta clase contiene métodos estáticos que permiten hacer la redirección de cualquiera de las tres formas anteriormente presentadas, respectivamente:

```
public static void doServerSessionRedirection(
    HttpServletRequest request,
    HttpServletResponse response,
    String url,
    PaymentData paymentData,
    PresentationData presentationData;

public static void doServerRequestRedirection(
    HttpServletRequest request,
    HttpServletResponse response,
    String url,
    PaymentData paymentData,
    PresentationData presentationData)

public static void doClientRedirection(
    HttpServletRequest request,
    HttpServletResponse response,
    String url,
    PaymentData paymentData,
    PresentationData presentationData)
```

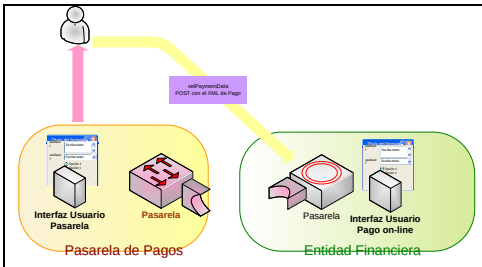
El parámetro *url* contiene la URL de la página de login de la aplicación de Banca Electrónica a la que se redirige al cliente en el inicio.

7.2 Servicio de escucha de peticiones de la Entidad Financiera.

Este servicio, que puede ser implementado como un servlet, JSP, etc…, es el encargado de recibir las peticiones provenientes de la Pasarela de Pagos. Del lado perteneciente a las **Entidades Financieras**, la Pasarela de la Administración ve las siguientes funciones lógicas, o lo que es lo mismo, las Entidades Financieras "exponen":

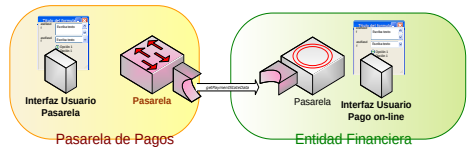
Interfaz Entidades Financieras	
Operación	**Descripción**
Recibir Pago	Recibe un mensaje XML con los datos de un pago para su realización on-line.
Consultar Estado Pago	Permite a la Pasarela de Pagos de la Administración enviar un mensaje XML con los datos necesarios para consultar el estado de un pago (si ha sido realizado o no)

Una petición de pago en una Entidad Financiera se hará mediante una llamada con el siguiente formato:



http://www.ef.com/[**urlServletPasarelaEF**]?
module=EF&
function= *setPaymentData*&
paymentData=[XML del pago]&
presentationData=[XMLpresent]&
protocolData=[XML protocolo]

Una consulta del estado de un pago en una Entidad Financiera se hará mediante una llamada con el siguiente formato:



http://www.ef.com/[**urlServletPasarelaEF**]?
module=EF&
function=[*getPaymentStateDataByCSB* *ó* *getPaymentStateDataByNRC*] &
paymentData=[XML del pago]&
protocolData=[XML protocolo]

El servicio debe recoger los parámetros y verificar si son correctos. En base a los parámetros *module* y *function* llamará al método que implementa la lógica de la función llamada pasándole los parámetros necesarios.

La lógica de esta función se implementa en la clase base ***p12e.exe.pasarelapagos.base.EFClass*** proporcionada en el SDK de la pasarela y que se ve en el siguiente apartado.

7.3 Clase base EFClass.

Esta clase encapsula la comunicación con la Pasarela de Pagos e implementa parcialmente la lógica de los métodos de la interfaz expuesta por las Entidades Financieras.

Esta clase debe ser instanciada en el servicio de escucha de peticiones de la Entidad Financiera. Al constructor se le pasa una instancia de la clase que implementa la interfaz ***FinantialOrgsFunction*** definida anteriormente y cuya construcción es responsabilidad de la Entidad financiera. La signatura del constructor es:

```
public EFClass(FinantialOrgFunctions impl);
```

Existe otro constructor que permite instanciar la clase ***EFClass*** a partir de los XML pasados como parámetros en la redirección de cliente. La signatura de este otro constructor es:

```
public EFClass(String paymentDataXML,
               String presentationDataXML,
               String protocolDataXML);
```

Esta clase dispone de métodos ***getXXX()*** que permiten acceder a sus propiedades, que en este caso se trata de las estructuras XML intercambiadas entre la Pasarela de Pagos y la Entidad Financiera:

```
public PaymentData getPaymentData();
public PresentationData getPresentationData();
public ProtocolData getProtocolData();
public PaymentResult getPaymentResult();
```

El servicio, en función de los parámetros recibidos, llamará al método de esta clase correspondiente a la función requerida a la Entidad Financiera:

- El método ***setPaymentData*** será llamado cuando se solicita un pago a través de la Entidad Financiera. Este método realiza las siguientes operaciones:
  - Validación contra la Pasarela de Pagos de los pagos recibidos por la Entidad Financiera.
  - Validaciones internas de la Entidad Financiera, utilizando los métodos de la clase cuya instancia ha sido proporcionada en el constructor.
  - En caso que todas las validaciones sean correctas, se redirige al cliente a la página de login de la Entidad Financiera. La signatura de este método es:

```
public void setPaymentData(
    String paymentDataXML,
    String presentationDataXML,
    String protocolDataXML,
    HttpServletRequest request,
    HttpServletResponse response)
    throws GatewayException, FinantialOrgException;
```

- En el caso de pagos múltiples, a medida que se van realizando transacciones en el host financiero, se va informando con el resultado de dichas transacciones a la clase ***EFClass***, invocando el método ***setSinglePaymentResult***. La signatura de este método es:

```
public void setSinglePaymentResult(
    String idPago,
    DatosPago datosPago,
    Estado estado);
```

- Una vez finalizado el proceso de pago, los resultados de los pagos serán enviados a la Pasarela de Pagos utilizando el método ***returnPaymentResult***:

```
public void returnPaymentResult()
    throws GatewayException;
```

- Los resultados de los pagos también pueden ser enviados de uno en uno a la Pasarela de Pagos utilizando el método ***returnSinglePaymentResult***, el cual no precisa que los pagos se vayan almacenando mediante el método ***setSinglePaymentResult***:

```
public void returnSinglePaymentResult(
    String idPago,
    DatosPago datosPago,
    Estado estado);
```

- El método ***getPaymentStateDataByCSB*** será llamado cuando se solicita el estado de un pago identificado por su CSB a la Entidad Financiera. Este método, en primer lugar, valida la solicitud del estado del pago contra la Pasarela y, en caso de que ésta sea correcta, devuelve la información del pago en una estructura ***Pago***. Posteriormente hace una solicitud del estado del pago utilizando un método de la clase cuya instancia ha sido proporcionada en el constructor. La signatura de este método es:

```
public void getPaymentStateDataByCSB(
    String csb,
    String protocolDataXML)
    throws FinantialOrgException;
```

- El método ***getPaymentStateDataByNRC*** será llamado cuando se solicita el estado de un pago identificado por su NRC a la Entidad Financiera. Este método, en primer lugar, valida la solicitud del estado del pago contra la Pasarela y, en caso de que ésta sea correcta, devuelve la información del pago en una estructura ***Pago***. Posteriormente hace una solicitud del estado del pago utilizando un método de la clase cuya instancia ha sido proporcionada en el constructor. La signatura de este método es:

```
public void getPaymentStateDataByNRC(
    String nrc,
    String protocolDataXML)
    throws FinantialOrgException;
```

Como se ha visto, los métodos presentados se comunican con la Pasarela de Pagos para realizar validaciones o devolver resultados de las operaciones

solicitadas a la Entidad Financiera. Para que esto sea posible, se deben habilitar las comunicaciones a través de *http(s)*:

Origen	**Destino**	**Puerto**
Servicio de Banca Electrónica	Pasarela de Pagos	443 / 80

7.4 HTML Helpers.

El paquete ***p12e.exe.pasarelapagos.html.helpers*** proporciona un conjunto de clases que permiten a las Entidades Financieras generar de forma inmediata las estructuras HTML RESPONSIVAS con la información relativa a los datos a presentar en pantalla.

Estas clases abstraen al programador de la generación del código HTML. Contienen métodos que tienen como parámetros los objetos Java cuya información se quiere presentar y devuelven un objeto *String* con el código HTML correspondiente, que deberá ser insertado en la correspondiente página JSP.

Estas clases tienen un nombre *XXXComposer* donde *XXX* hace referencia al nombre de la estructura de datos cuya información se quiere presentar. Estas clases a su vez contienen un método *composeXXX* que devuelve un *String* con la estructura HTML que presenta la información requerida.

A continuación se presentan las clases del paquete ***p12e.exe.pasarelapagos.html.helpers*** junto con una vista de las estructuras HTML que generan.

7.4.1 PaymentListComposer.

Esta clase permite, a través del método *composePaymentList,* generar la tabla que presenta el listado de pagos recibidos:

```
public static String composePaymentList(
      final PaymentData paymentData
      final String language);
```

Este método recibe como parámetros:

- Un objeto *PaymentData* que encapsula la información relativa a los pagos.
- Un *String* con el idioma en el que debe presentarse la información.

Este mismo método, llamando con distintos parámetros, permite generar la tabla que presenta la confirmación de un lote de pagos. La confirmación muestra qué pagos se han realizado correctamente y qué pagos no:

```
public static String composePaymentList(
      final PaymentData paymentData
      final PaymentResult paymentResult
      final String language);
```

Este método recibe como parámetros:

- Un objeto *PaymentData* que encapsula la información relativa a los pagos.
- Un objeto *PaymentResult* que encapsula el resultado correspondiente a los pagos realizados.
- Un *String* con el idioma en el que debe presentarse la información.

Por un lado, el siguiente trozo de código muestra un ejemplo de utilización de este método para generar el listado de pagos recibidos:

```
PaymentData paymentData = new PaymentData();
// Rellenar la estructura Pago con los datos de los pagos…

String codigoHTML =
PaymentListComposer.composePaymentList(paymentData, "es");
```

A continuación se muestra la tabla de pagos recibidos generada por el método *composePaymentList*:

Datos de los recibos a pagar

Emisor	Recibo	Importe	Detalle
Eusko Jaurlaritzak / Gobierno Vasco	Cuadernillo 57 - estándar	2,33 €	

Por otro lado, el siguiente trozo de código muestra un ejemplo de utilización de este método para generar la confirmacion de un lote de pagos:

```
PaymentData paymentData = new PaymentData();
// Rellenar la estructura Pago con los datos de los pagos…

PaymentResult paymentResult = new PaymentResult ();
// Rellenar la estructura con los resultados de los pagos…

String codigoHTML =
PaymentListComposer.composePaymentList(paymentData, paymentResult,
"es");
```

A continuación se muestra la tabla de pagos confirmados generada por el método *composePaymentList*:

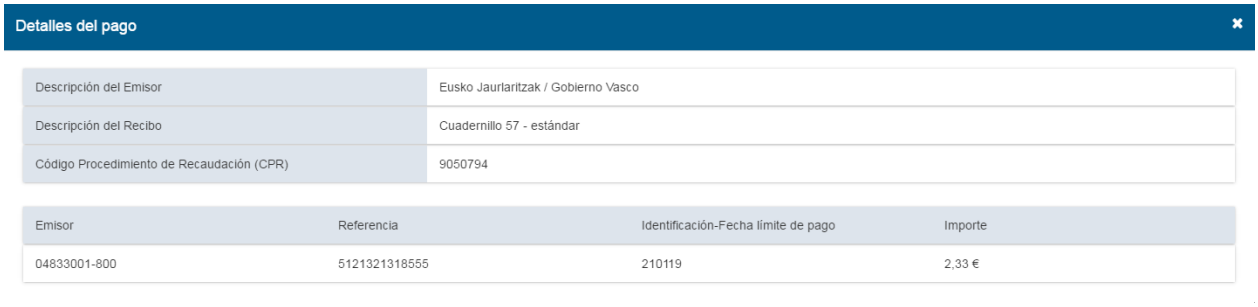
DATOS DE LOS RECIBOS PAGADOS

Emisor	Recibo	Importe	Detalle
Eusko Jaurlaritzak / Gobierno Vasco	Cuadernillo 57 - estándar	2,33 €	

DATOS DE LOS RECIBOS NO PAGADOS

Emisor	Recibo	Importe	Detalle
Eusko Jaurlaritzak / Gobierno Vasco	Cuadernillo 57 - estándar	1,25 €	

El botón que tiene cada pago en la columna "Detalle" llama a una función javascript, que abre una capa en la que se muestra el detalle del pago:



7.4.2 PaymentReceiptComposer.

Esta clase permite, a través del método *composePaymentReceipt,* generar el justificante de los pagos realizados. Puede ser llamado con distintos parámetros:

```
public static String composePaymentReceipt(
       final PaymentData paymentData,
       final PaymentResult paymentResult,
       final String language,
       final String urlFinantialOrgLogo);
```

Este método recibe como parámetros:

- Un objeto *PaymentData* que encapsula la información relativa a los pagos realizados.
- Un objeto *PaymentResult* que encapsula el resultado correspondiente a los pagos realizados.
- Una cadena con el idioma en que se debe mostrar la información.
- Una cadena con la URL de la que obtener el logo de la entidad financiera.

```
public static String composePaymentReceipt(
       final PaymentData paymentData,
       final PaymentResult paymentResult,
       final String language,
       final String urlFinantialOrgLogo,
       final boolean hasSign);
```

Este método recibe como parámetros:

- Un objeto *PaymentData* que encapsula la información relativa a los pagos realizados.
- Un objeto *PaymentResult* que encapsula el resultado correspondiente a los pagos realizados.
- Una cadena con el idioma en que se debe mostrar la información.
- Una cadena con la URL de la que obtener el logo de la entidad financiera.
- Un booleano que indica si se tiene que dejar espacio o no para la firma.

```
public static String composePaymentReceipt(
        final PaymentData paymentData,
        final PaymentResult paymentResult,
        final String language,
        final String urlFinantialOrgLogo,
        final boolean hasSign
        final String backURL);
```

Este método recibe como parámetros:

- Un objeto *PaymentData* que encapsula la información relativa a los pagos realizados.
- Un objeto *PaymentResult* que encapsula el resultado correspondiente a los pagos realizados.
- Una cadena con el idioma en que se debe mostrar la información.
- Una cadena con la URL de la que obtener el logo de la entidad financiera.
- Un booleano que indica si se tiene que dejar espacio o no para la firma.
- Una cadena con la URL de vuelta una vez completado el pago.

El siguiente trozo de código muestra un ejemplo de utilización de este método:

```
String codigoHTML =
PaymentReceiptComposer.composePaymentReceipt(paymentData,
paymentresult, "es", logo);
```

A continuación, se muestra la información generada por el método *composePaymentReceipt* para el caso de un pago simple:

bbk

Internet bidez ordaindu izanaren agiria
Justificante de pago de recibo por Internet

Igorlearen deskribapena Descripción del Emisor	Eusko Jaurlaritzak / Gobierno Vasco Eusko Jaurlaritzak / Gobierno Vasco
Ordainagiriaren deskribapena Descripción del Recibo	Cuadernillo 57 - estándar [EUS] Cuadernillo 57 - estándar

EZO - Erreferentzia Zenbaki Osoa NRC - Número de Referencia Completo		1496218033745359042F5	
Diru-sarreraren eguna eta ordua Fecha y hora del ingreso		31/05/2017 [10:07:13]	
Erakunde Kodea Código de Entidad	9999	**Bulego Kodea** Codigo de Oficina	0001

Dokumentu hau ordainketaren egiaztagiria da eta zordunari ordaintzeko betekizuna kentzen dio, bertan adierazitako data eta kopuruarengatik.

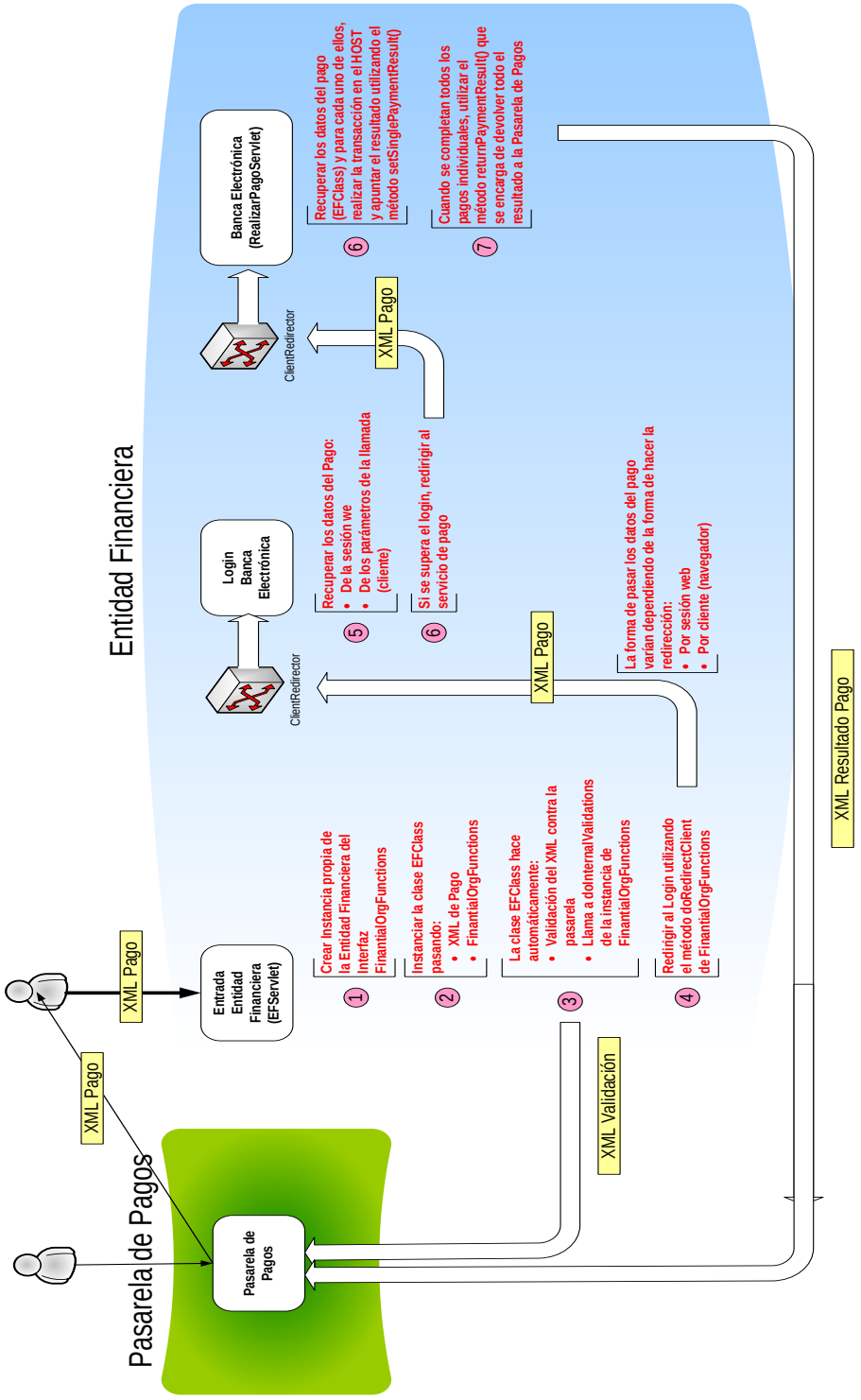
Este documento constituye justificante de pago y libera al deudor de su obligación de pago desde la fecha y por el importe señalados.

Ordainketa-gutuna Carta de Pago		**Bilketa Prozedura Kodea (BPK)** Código Procedimiento de Recaudación (CPR)				9050794	
Igortze-entitatea Entidad emisora	04833001-800	**Erreferentzia** Referencia	5121321318555	**Identifikazioa** Identificación	210119	**Zenbatekoa** Importe	EUR **** 2,33

En el caso de un pago múltiple, se irían sucediendo estructuras como ésta para cada uno de los pagos.

7.5 Ejemplos.

En este apartado se pretende mostrar a través de ejemplos el proceso de desarrollo a llevar a cabo en la Entidad Financiera, utilizando las clases provistas por el SDK de la Pasarela de Pagos. Se deberán seguir los siguientes pasos:



7.5.1 Implementación de la interfaz FinantialOrgFunctions.

El primer paso que debe llevar a cabo la Entidad Financiera es la construcción de una clase que implemente la interfaz ***FinantialOrgsFunction***, que como se ha visto encapsula la lógica propia a la operación en la Entidad Financiera.

```
package p12e.exe.pasarelapagos.test;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import p12e.exe.pasarelapagos.base.*;
import p12e.exe.pasarelapagos.redirection.*;
import p12f.exe.pasarelapagos.exceptions.*;
import p12f.exe.pasarelapagos.helpers.*;
import p12f.exe.pasarelapagos.objects.*;

/**
 * Clase de implementación de ejemplo de los métodos de la Entidad Financiera.
 */
public class FinantialOrgFunctionsImpl implements FinantialOrgFunctions, java.io.Serializable
{

    /**
     * Método que realiza las validaciones internas de la Entidad financiera.
     */
    public void doInternalValidations(PaymentData paymentData)
            throws FinantialOrgException {
    }

    /**
     * Método que obtiene el estado de un pago identificado por su CSB.
     */
    public Estado doGetPaymentStateDataByCSB(String csb) throws FinantialOrgException {

        HostData hostData = null;
        hostData = HostEntidadFinanciera.consultarPorRafaga(csb);

        Estado estado = new Estado();
        if(hostData.estadoPago.equals("1")){
            estado.codigo = Estado.PAGADO;
            estado.numeroOperacion = hostData.numeroOperacion;
            estado.fechaPago = hostData.fechaPago;
            estado.horaPago = hostData.horaPago;
            estado.entidad = "0001";
            estado.importe = hostData.importe;
            estado.nrc = "976376567495696976";
        }else if(hostData.estadoPago.equals("0")){
            estado.codigo = Estado.ERROR_PAGO;
        }
        return estado;
    }

    /**
     * Método que obtiene el estado de un pago identificado por su NRC.
     */
    public Estado doGetPaymentStateDataByNRC(String nrc) throws FinantialOrgException {

        HostData hostData = null;
        hostData = HostEntidadFinanciera.consultarPorNRC(nrc);

        Estado estado = new Estado();
        if(hostData.estadoPago.equals("1")){
            estado.codigo = Estado.PAGADO;
            estado.numeroOperacion = hostData.numeroOperacion;
            estado.fechaPago = hostData.fechaPago;
            estado.horaPago = hostData.horaPago;
            estado.entidad = "0001";
            estado.importe = hostData.importe;
            estado.nrc = "976376567495696976";
```

```
        }else if(hostData.estadoPago.equals("0")){
            estado.codigo = Estado.ERROR_PAGO;
        }
        return estado;
    }

    /**
     * Método encargado de redirigir al cliente a la pantalla de login de la Entidad
Financiera.
     */
    public void doRedirectClient(HttpServletRequest request,
            HttpServletResponse response,
            String targetUrl,
            EFClass ef) {
        ClientRedirector.doServerSessionRedirection(request, response, targetUrl, ef);
    }

}
```

7.5.2 Servlet de escucha de peticiones EFServlet.

Este servlet es un ejemplo de cómo implementar el punto de escucha de la aplicación de Banca Electrónica que recibirá las peticiones de realización de pagos.

```
package p12e.exe.pasarelapagos.test;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.*;
import javax.servlet.http.*;

import p12e.exe.pasarelapagos.base.*;
import p12f.exe.pasarelapagos.exceptions.*;
import p12f.exe.pasarelapagos.objects.*;

import com.ejie.r01f.log.R01FLog;
import com.ejie.r01f.xml.marshalling.XOMarshallerException;

public class EFServlet extends HttpServlet {

    /**
     * Constructor of the object.
     */
    public EFServlet() {
        super();
    }

    /**
     * The doGet method of the servlet.
     */
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doExec(request, response);
    }

    /**
     * The doPost method of the servlet.
     */
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doExec(request, response);
    }

    /**
     * The doExec method of the servlet.
     */
    public void doExec(HttpServletRequest request, HttpServletResponse response) throws
```

```
ServletException, IOException {

        // Se recogen los parámetros de la petición
        String module = request.getParameter("module");
        String function = request.getParameter("function");
        String paymentDataXML = request.getParameter("paymentData");
        String presentationDataXML = request.getParameter("presentationData");
        String protocolDataXML = request.getParameter("protocolData");
        String csb = request.getParameter("csb");
        String nrc = request.getParameter("nrc");

        // Se instancia la clase que implementa la lógica propia de la Entidad Financiera.
        FinantialOrgFunctions fof = new FinantialOrgFunctionsImpl();
        // Se instancia un objeto de la clase EFClass
        EFClass ef = new EFClass(fof);

        // Se comprueba que método remoto se ha invocado
        if(function!=null && function.equals("setPaymentData")){
            try{
                // Se llama al método setPaymentData()
                ef.setPaymentData(paymentDataXML,
                        presentationDataXML,
                        protocolDataXML,
                        request,
                        response);
                // Se redirige al cliente a la página de login de la Entidad Financiera
                fof.doRedirectClient(request,response,"login.jsp",ef);
            }catch(Exception e){
                e.printStackTrace();
            }
        }else if(function!=null && function.equals("getPaymentStateDataByCSB")){
            R01FLog.to("p12e.getPaymentStateDataByCSB").info("Se ha recibido la peticion del
estado del pago con CSB: " + csb);
            R01FLog.to("p12e.getPaymentStateDataByCSB").info("ProtocolData: " +
protocolDataXML);

            try {
                // Respuesta en forma de XML
                String returnXML =  null;
                PrintWriter out = response.getWriter();
                Estado estado = ef.getPaymentStateDataByCSB(csb, protocolDataXML);
                // Se envía de vuelta el XML del resultado de la operación
                returnXML = estado.toXML();
                out.println(returnXML);
                out.flush();
                out.close();
            } catch (Exception e) {
                R01FLog.to("p12e.getPaymentStateDataByCSB").info("Error en la operación: NO se
devuelve respuesta.");
                e.printStackTrace();
            }
        }else if(function!=null && function.equals("getPaymentStateDataByNRC")){
            R01FLog.to("p12e.getPaymentStateDataByNRC").info("Se ha recibido la peticion del
estado del pago con NRC: " + nrc);
            R01FLog.to("p12e.getPaymentStateDataByNRC").info("ProtocolData: " +
protocolDataXML);

            try {
                // Respuesta en forma de XML
                String returnXML =  null;
                PrintWriter out = response.getWriter();
                Estado estado = ef.getPaymentStateDataByNRC(nrc, protocolDataXML);
                // Se envía de vuelta el XML del resultado de la operación
                returnXML = estado.toXML();
                out.println(returnXML);
                out.flush();
                out.close();
            } catch (Exception e) {
                R01FLog.to("p12e.getPaymentStateDataByNRC").info("Error en la operación: NO se
devuelve respuesta.");
                e.printStackTrace();
            }
```

```
        }
    }
}
```

7.5.3 Página de login de la aplicación de Banca Electrónica.

Esta página será la página de inicio del interfaz de usuario de la aplicación de Banca Electrónica. Se pedirá al usuario los datos necesarios para logearse en el servicio de Banca Electrónica y se mostrará un listado con información de los pagos recibidos.

Para realizar la implementación, se proporcionarán los componentes que se encargan de pintar tanto el listado de pagos, como el justificante. Estos componentes permiten la responsividad de los elementos según el dispositivo desde el que se visualizan.

Para usarlos correctamente, es necesario importarlos en el *jsp* de la siguiente manera:

En la parte **head**, hay que añadir:

```
<meta http-equiv="X-UA-Compatible" content="IE=edge"/>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
<meta name="viewport" content="width=device-width, initial-scale=1"/>

<script src='/rutaContenidoEstatico/jquery.min.js'></script>
<script src='/rutaContenidoEstatico/p12Components.js'></script>
<link rel='stylesheet' href='/rutaContenidoEstatico/p12Components.css'>
```

Y en la parte del **body**, hay que añadir lo siguiente:

```
<div id="paymentListDiv" class="p12Container"></div>
<script src="/rutaContenidoEstatico/bundle.js" type="text/javascript"></script>
```

Un ejemplo implementación es la siguiente:

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="X-UA-Compatible" content="IE=edge" >
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <script src='/rutaContenidoEstatico/jquery.min.js'></script>
        <script src='/rutaContenidoEstatico/p12Components.js'></script>
        <link rel='stylesheet' href='/rutaContenidoEstatico/p12Components.css'>
    </head>
    <body>
        <%@ page import="p12e.exe.pasarelapagos.base.EFClass" %>
        <%@ page import="p12e.exe.pasarelapagos.html.helpers.PaymentListComposer" %>
        <%@ page import="p12f.exe.pasarelapagos.objects.PaymentData" %>
        <%@ page import="p12f.exe.pasarelapagos.objects.PaymentResult"%>
        <%@ page import="p12f.exe.pasarelapagos.objects.PresentationData" %>
        <%@ page import="p12f.exe.pasarelapagos.objects.ProtocolData" %>
        <%@ page import="p12f.exe.pasarelapagos.redirection.SessionAttrs" %>
        <%@ page import="com.ejie.r01f.xmlproperties.XMLProperties" %>
        <%@ page import="java.util.Iterator" %>

        <%
        //
        // Recogemos los datos del pago en funcion del modo en que hayamos redirigido al
        // cliente a la aplicación de banca electrónica.
```

```
        //

        //  Método 1: Redirección de servidor utilizando session.
        //      En la llamada a setPaymentData se puso en session el objeto EFClass
        //
        EFClass ef = (EFClass)session.getAttribute(SessionAttrs.EFCLASS);

        //  Método 2: Redirección de servidor sin utilizar session.
        //      En la llamada a setPaymentData se puso en la request el objeto EFClass
        //EFClass ef = (EFClass)request.getAttribute(RequestAttrs.EFCLASS);

        //  Método 3: Redirección de servidor a través del cliente.
        //      Se utilizó una redirección de cliente para llegar a esta página.
        //  En la redirección se pasaron como parámetro los XML y ahora hay
        //  que obtener de nuevo los objetos.
        //String paymentDataXML = request.getParameter(RequestParams.PAYMENT_DATA);
        //String presentationDataXML = request.getParameter(RequestParams.PRESENTATION_DATA);
        //String protocolDataXML = request.getParameter(RequestParams.PROTOCOL_DATA);
        //EFClass ef = new EFClass(paymentDataXML, presentationDataXML, protocolDataXML);

        PaymentData paymentData = ef.getPaymentData();
        ProtocolData protocolData = ef.getProtocolData();
        PresentationData presentationData = ef.getPresentationData();
        PaymentResult paymentResult = ef.getPaymentResult();
        %>

        <div class="container-fluid" style="max-width: 990px; min-height: 500px; margin-
bottom: 20px;">
            <div class="page-header">
                <div class="row" style="padding: 1em; vertical-align: middle;">
                    <div class="col-xs-12 col-md-6" style="text-align:center; background-
color: #CCCCCC; border-radius: 8px 8px 8px 8px;"><h3 style="margin:0px; padding:0.8em;"><span
style="">Entidad Financiera de Simulaci&oacute;n</span></h3></div>
                    <div class="col-xs-12 col-md-6" style="text-align:center"><img
alt="Brand"
src="https://www6.euskadi.net/appcont/newPaymentGateway2/images/logo_es.jpg"></div>
                </div>
            </div>

            <div id="paymentListDiv" class="p12Container"></div>
            <script src="/appcont/p12EEFFComponents/src/client/public/bundle.js"
type="text/javascript"></script>

            <%=PaymentListComposer.composePaymentList(paymentData,presentationData.idioma)%>

            <div>
                - Pago a traves de banca electronica:
                <br>
                <br>
                <table cellpadding="0" cellspacing="0" align="center">
                    <tr>
                        <td>Usuario</td>
                        <td><input type="text"></td>
                    </tr>
                    <tr>
                        <td>Password</td>
                        <td><input type="text"></td>
                    </tr>
                </table>
            </div>
            <div>
            <a href="RealizarPagoServlet?R01HNoPortal=true">Entrar en Banca Electronica</a>
        </div>
    </div>
    </body>
</html>
```

Cuando el usuario se loguea se procede a la realización de los pagos.

7.5.4 Servlet que maneja los pagos en la Entidad Financiera.

El cliente de Banca Electrónica es redirigido a este servlet una vez que se loguea. Se procede al pago de los pagos recibidos contra el host de la Entidad Financiera. Los pagos se realizan sobre el host de uno en uno, almacenándose su resultado en la propiedad ***PaymentResult*** de ***EFClass***, a través del método ***setSinglePaymentResult().*** Al acabar de realizar los pagos se procede a la comunicación de los resultados de los pagos a la Pasarela de Pagos mediante el método ***setSinglePaymentResult()***de ***EFClass***.

```
package p12e.exe.pasarelapagos.test;

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Iterator;

import javax.servlet.*;
import javax.servlet.http.*;

import org.bouncycastle.util.encoders.Hex;

import p12e.exe.pasarelapagos.base.EFClass;
import p12e.exe.pasarelapagos.redirection.SessionAttrs;
import p12f.exe.pasarelapagos.objects.*;

/**
 * Servlet encargado de la realización de un pago en la Entidad Financiera.
 */
public class RealizarPagoServlet extends HttpServlet {

    /**
     * Constructor of the object.
     */
    public RealizarPagoServlet() {
        super();
    }

    /**
     * The doGet method of the servlet. <br>
     */
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doExec(request, response);
    }

    /**
     * The doPost method of the servlet. <br>
     */
    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doExec(request, response);
    }

    /**
     * The doExec method of the servlet. <br>
     */
    public void doExec(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        /*
         * Recogemos los datos del pago en funcion del modo en que hayamos redirigido al
         * cliente a la aplicación de banca electrónica.
         */

        /*  Método 1: Redirección de servidor utilizando session.
         *      En la llamada a setPaymentData se puso en session el objeto EFClass
         */
        HttpSession session = request.getSession();
```

```
        EFClass ef = (EFClass)session.getAttribute(SessionAttrs.EFCLASS);

        /*  Método 2: Redirección de servidor sin utilizar session.
         *      En la llamada a setPaymentData se puso en la request el objeto EFClass
         */
        //EFClass ef = (EFClass)request.getAttribute(RequestAttrs.EFCLASS);

        /*  Método 3: Redirección de servidor a través del cliente.
         *      Se utilizó una redirección de cliente para llegar a esta página.
         *  En la redirección se pasaron como parámetro los XML y ahora hay
         *  que obtener de nuevo los objetos.
         */
        //String paymentDataXML = request.getParameter(RequestParams.PAYMENT_DATA);
        //String presentationDataXML = request.getParameter(RequestParams.PRESENTATION_DATA);
        //String protocolDataXML = request.getParameter(RequestParams.PROTOCOL_DATA);
        //EFClass ef = new EFClass(paymentDataXML, presentationDataXML, protocolDataXML);

        // Realizar los pagos
        try{
            PaymentData paymentData = ef.getPaymentData();
            /*
             * Realizar cada uno de los pagos en una transacción INDEPENDIENTE contra el HOST.
             * En la petición puede llegar un sólo pago o un pago múltiple, hay que realizar
             * una transacción de pago para cada uno de los pagos del lote.
             * El objeto PaymentData tiene un mapa con todos los pagos del lote.
             * Si se trata de un pago sencillo unicamente hay un elemento.
             * Basta con recorrer el objeto y realizar una transacción para cada uno.
             */
            Pago currPago = null;
            int hostResponse = -1;
            /*
             * Transacción por cada pago del lote.
             */
            for(Iterator it = paymentData.pagos.keySet().iterator(); it.hasNext();){
                currPago = (Pago)paymentData.pagos.get(it.next());


                // Si se trata de un lote y es el ultimo pago hacemos que falle
                boolean error = false;
                if(paymentData.pagos.size()>1&&!it.hasNext()){
                error = true;
                }

                /*
                 * Aquí se tienen todos los datos del pago que son accesibles
                 * como un objeto Java normal, se puede acceder a los datos de
                 * cualquier miembro de los señalados en la documentación.
                 */
                String cpr = currPago.datosPago.cpr; // CPR: Código de Procedimiento
Recaudatorio
                String rafaga = currPago.datosPago.codigo; // Código de barras
                // ... acceder a cualquier dato del pago .. //
                try{
                    /*
                     * Aquí la entidad financiera realizaria una transacción contra su host
                     * con los datos que considere necesarios de los que están disponibles
                     * en el objeto de pago.
                     * Haciendo una suposición de la llamada, suponiendo que el host devuelve
                     * únicamente un número indicativo del resultado (seguramente no sea
                     * un caso real... es un ejemplo)
                     */
                    hostResponse = HostEntidadFinanciera.transaccionPago(cpr, rafaga);
                    if(error){
                    hostResponse = -30;
                    }
                }catch(Exception ex){
                    hostResponse = -30;  // Para poder marcar el error en el justificante
                    ex.printStackTrace();
                    System.out.println(">>>Error transaccion HOST " + ex.toString());
                    throw new Exception();
                }
                // Componer un objeto PaymentStateData con el resultado del pago actual.
```

```
                Estado estado = new Estado();
                if(hostResponse==0){
                    estado.codigo = Estado.PAGADO;
                    estado.numeroOperacion = "12345678901234";
                    Date fecha = new Date();
                    SimpleDateFormat dateFormatter = new SimpleDateFormat("ddMMyy");
                    SimpleDateFormat hourFormatter = new SimpleDateFormat("hhmmss");
                    estado.fechaPago = dateFormatter.format(fecha);
                    estado.horaPago = hourFormatter.format(fecha);

                    estado.importe = new Long(currPago.getImporte()).toString();
                    estado.entidad = "0001";

                    String emisor = currPago.datosPago.emisor;
                    if(currPago.datosPago.tipo.equals("507")){
                    emisor = emisor.substring(0,8)+emisor.substring(9,12);
                    }
                    String referencia = currPago.datosPago.referencia;
                    String importe = estado.importe;
                    String justificante = estado.numeroOperacion;

                    String entrada = emisor +  referencia +  importe +  justificante;

                    String clave = "ABABABABABABABAB";
                    byte[] keyBytes = Hex.decode(clave);
                    byte[] keyData = entrada.getBytes();
                    String mac = NRCHelper.calcularMAC(keyData, keyBytes);

                    estado.nrc = estado.numeroOperacion + mac;

                }else{
                    estado.codigo = Estado.ERROR_PAGO;
                    Mensaje mensaje1 = new Mensaje();
                    mensaje1.id = "error1";
                    mensaje1.texto.put("es", "Error en la realizacion del pago");
                    mensaje1.texto.put("eu", "Error en la realizacion del pago_eu");
                    estado.mensajes.put(mensaje1.id, mensaje1);

                    Mensaje mensaje2 = new Mensaje();
                    mensaje2.id = "error2";
                    mensaje2.texto.put("es", "El error se debe a que no dispone de saldo
suficiente");
                    mensaje2.texto.put("eu", "El error se debe a que no dispone de saldo
suficiente_eu");
                    estado.mensajes.put(mensaje2.id, mensaje2);

                    /*
                     *  Indicar la razón del error en el pago, ya que este dato se
                     *  utiliza para componer el justificante de pago e indicar que
                     *  pagos han fallado y por qué motivo.
                     */
                    if(hostResponse == -20){
                        // El emisor/sufijo no esta dado de alta en el HOST...
                        estado.razonError = "No existe el sufijo";
                    }else if(hostResponse == -30){
                        // Error técnico general...
                        estado.razonError = "Error general";
                    }
                }
                /*
                 * El resultado de cada pago individual se va almacenando en un objeto que
"acumula"
                 * los resultados de cada transacción para devolver el resultado a la Pasarela
de
                 * la Administración.
                 */
                ef.returnSinglePaymentResult(currPago.id, currPago.datosPago ,estado);
            } // Cada uno de los pagos
            /*
             * Una vez realizadas las transacciones individuales de pago, devolver el
resultado
             * de pago a la pasarela de la Administración.
```

```
             */
            //PaymentResult paymentResult = ef.returnPaymentResult();
            // Si no hay error, redirigimos al cliente a la página de emisión del
justificante.
            RequestDispatcher rd = request.getRequestDispatcher("/confirmacion.jsp");
            rd.forward(request, response);
        }catch(Exception ex){
            ex.printStackTrace();
            System.out.println(">>> Ha ocurrido un error en la tramitación contra el HOST " +
ex.toString());
            // Si hay error, redirigimos al cliente a la página de error.
            RequestDispatcher rd = request.getRequestDispatcher("/error.jsp");
            rd.forward(request, response);
        }
    }
}
```

7.5.5 Página de justificante de la aplicación de Banca Electrónica.

En esta página se mostrará el justificante de los pagos realizados en la aplicación de Banca Electrónica.

Para realizar la implementación, se proporcionarán los componentes que se encargan de pintar tanto el listado de pagos, como el justificante. Estos componentes permiten la responsividad de los elementos según el dispositivo desde el que se visualizan.

Para usarlos correctamente, es necesario importarlos de la siguiente manera:

En la parte **head**, hay que añadir:

```
<meta http-equiv="X-UA-Compatible" content="IE=edge"/>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
<meta name="viewport" content="width=device-width, initial-scale=1"/>

<script src='/rutaContenidoEstatico/jquery.min.js'></script>
<script src='/rutaContenidoEstatico/p12Components.js'></script>
<link rel='stylesheet' href='/rutaContenidoEstatico/p12Components.css'>
```

Y en la parte del **body**, hay que añadir lo siguiente:

```
<div id="paymentReceiptDiv" class="p12Container"></div>
<script src="/rutaContenidoEstatico/bundle.js" type="text/javascript"></script>
```

Un ejemplo implementación es la siguiente:

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="X-UA-Compatible" content="IE=edge" >
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <script src='/rutaContenidoEstatico/jquery.min.js'></script>
        <script src='/rutaContenidoEstatico/p12Components.js'></script>
        <link rel='stylesheet' href='/rutaContenidoEstatico/p12Components.css'>
    </head>
    <body>
        <%@ page import="p12e.exe.pasarelapagos.html.helpers.PaymentReceiptComposer" %>
        <%@ page import="p12f.exe.pasarelapagos.objects.PaymentData" %>
        <%@ page import="p12f.exe.pasarelapagos.objects.PaymentResult"%>
        <%@ page import="p12f.exe.pasarelapagos.objects.PresentationData" %>
        <%@ page import="p12f.exe.pasarelapagos.objects.ProtocolData" %>
```

```
        <%@ page import="p12f.exe.pasarelapagos.objects.URL" %>
        <%@ page import="p12f.exe.pasarelapagos.redirection.RequestParams" %>
        <%@ page import="com.ejie.r01f.xmlproperties.XMLProperties" %>

        <%
        // Recogemos los datos del pago en funcion del modo en que hayamos redirigido al
        // cliente a la aplicación de banca electrónica.
        //  Método 1: Redirección de servidor utilizando session.
        //      En la llamada a setPaymentData se puso en session el objeto EFClass
        //
        //EFClass ef = (EFClass)session.getAttribute(SessionAttrs.EFCLASS);
        //PaymentData paymentData = ef.getPaymentData();
        //ProtocolData protocolData = ef.getProtocolData();
        //PresentationData presentationData = ef.getPresentationData();
        //PaymentResult paymentResult = ef.getPaymentResult();

        //  Método 2: Redirección de servidor sin utilizar session.
        //      En la llamada a setPaymentData se puso en la request el objeto EFClass
        //EFClass ef = (EFClass)request.getAttribute(RequestAttrs.EFCLASS);

        //  Método 3: Redirección de servidor a través del cliente.
        //      Se utilizó una redirección de cliente para llegar a esta página.
        //  En la redirección se pasaron como parámetro los XML y ahora hay
        //  que obtener de nuevo los objetos.
        String paymentDataXML = request.getParameter(RequestParams.PAYMENT_DATA);
        String presentationDataXML = request.getParameter(RequestParams.PRESENTATION_DATA);
        String protocolDataXML = request.getParameter(RequestParams.PROTOCOL_DATA);
        String paymentResultXML = request.getParameter(RequestParams.PAYMENT_RESULT);
        System.out.println(paymentDataXML);
        PaymentData paymentData = PaymentData.getObject(paymentDataXML);
        ProtocolData protocolData = ProtocolData.getObject(protocolDataXML);
        PresentationData presentationData = PresentationData.getObject(presentationDataXML);
        PaymentResult paymentResult = PaymentResult.getObject(paymentResultXML);
        Url urlVuelta = (Url)protocolData.urls.get("urlVuelta");

        String carpetaContenidosPaymentGateway =
XMLProperties.get("p12et","contentFolder/paymentGateway");
        %>

        <div>

            <div id="paymentReceiptDiv" class="p12Container"></div>
            <script src="/appcont/p12EEFFComponents/src/client/public/bundle.js"
type="text/javascript"></script>

            <%=PaymentReceiptComposer.composePaymentReceipt(paymentData, paymentResult,
presentationData.idioma, carpetaContenidosPaymentGateway+"images/logoentidades/bbk.gif",
false, urlVuelta.url)%>

        </div>
    </body>
</html>
```

7.6 Personalización de estilos de la nueva interfaz responsiva.

Los componentes se pueden personalizar para satisfacer las necesidades de la entidad financiera. Cada componente tiene unos IDs que permiten modificar los estilos a través de CSS.

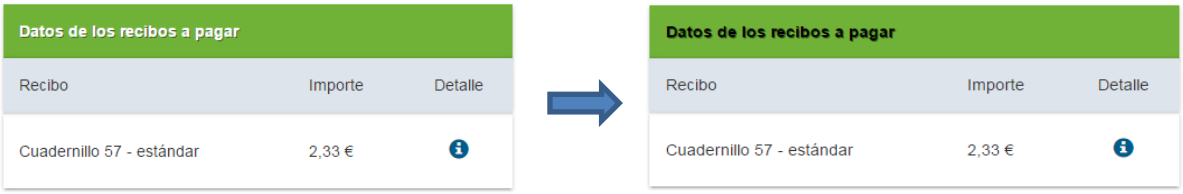
Estos IDs son los siguientes:

- **PaymentListHeader, PaymentListHeaderNotPaid**: Permite modificar el color de la cabecera del listado de pagos, tanto de los pagos pendientes, pagados, y no pagados. Ejemplo:

#paymentListHeader {background-color:black !important;}

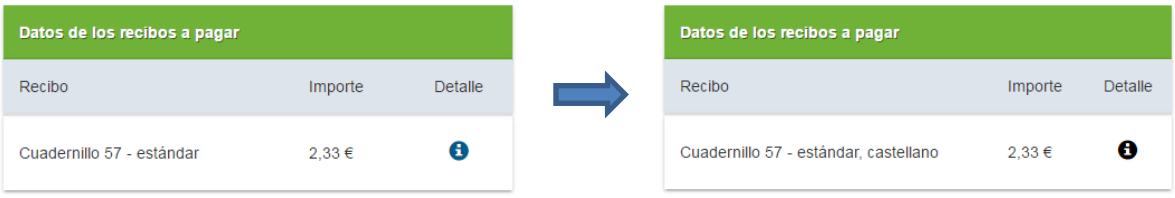
- **PaymentListHeaderTitle, PaymentListHeaderTitleNotPaid**: Permite modificar el color del texto de la cabecera del listado de pagos, tanto de los pagos pendientes, pagados, y no pagados. Ejemplo:

#paymentListHeaderTitle {color:black !important;}



- **PaymentListItemIcon**: Permite modificar el color del icono de "*Detalle*" del listado de pagos. Ejemplo:

#paymentListItemIcon {color:black !important;}

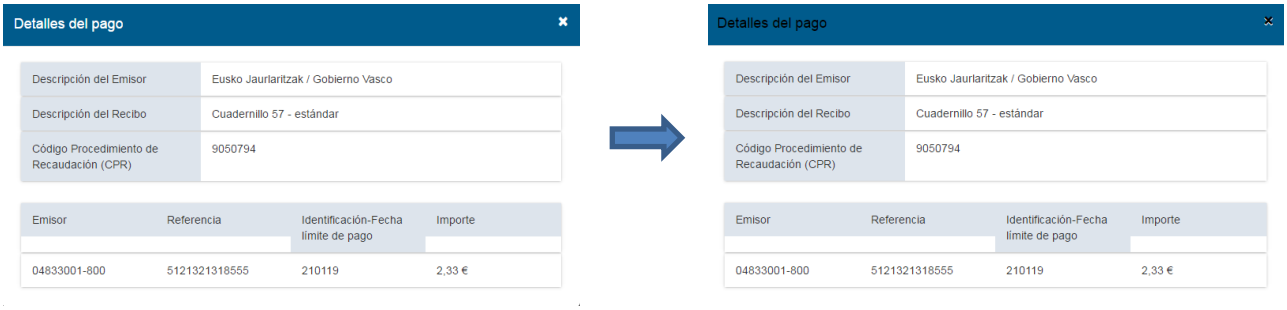


- **DialogBoxHeader**: Permite modificar el color de la cabecera de la ventana Popup de detalle de un pago. Ejemplo:

  #dialogBoxHeader {background-color:black !important;}

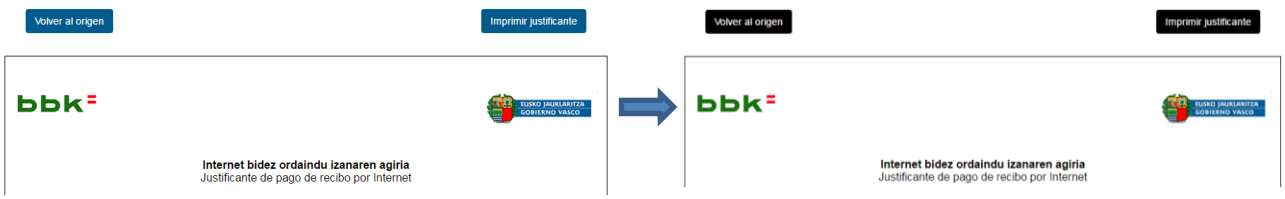
- **DialogBoxHeaderTitle, DialogBoxHeaderButton**: Permiten modificar el color del texto de la cabecera y del botón "X" (cerrar) de la ventana Popup de detalle de un pago. Ejemplo:

  #dialogBoxHeaderTitle {color:black !important;}
  #dialogBoxHeaderButton {color:black !important;}



- **ReceiptBackButton, ReceiptPrintButton**: Permiten modificar el color de los botones del justificante, tanto para volver al origen, como para imprimir el justificante. Ejemplo:

  #receiptBackButton {background-color:black !important;}
  #receiptPrintButton {background-color:black !important;}



8 Applet lector de tarjetas.

8.1 Introducción.

El lector de tarjetas permite capturar en la aplicación de la Entidad Financiera la información de la tarjeta del ciudadano que quiere realizar el pago en una Ventanilla de la Administración. Esta información consiste en el número de la tarjeta, el titular de ésta y la fecha de caducidad.

El lector de tarjetas opera de forma conjunta con un applet Java que recoge los datos enviados por el lector de tarjetas a través del puerto serie. Se trata de un applet firmado proporcionado por EJIE a la Entidades Financieras para que éstas lo incluyan en sus aplicaciones.

Es este capítulo se pretende dar una explicación detallada de los pasos a realizar para llevar a cabo la instalación del lector de tarjetas en los equipos de la Administración, suponiendo que estos equipos se ejecutan bajo sistema operativo Windows XP.

8.2 Configuración en el cliente.

Se va a utilizar un applet para recoger los datos que envía el lector de tarjetas del puerto serie del equipo. Para ello el equipo debe tener instalado un *Java Runtime Environement (JRE)* que permita la ejecución de applets en el navegador. Se debe comprobar que el navegador tiene habilitado el JRE en el menú *Herramientas -> Opciones de Internet -> Opciones avanzadas*.

Como se ha dicho anteriormente, el applet lee los datos que envía el lector de tarjetas del puerto serie del equipo. Para ello se utilizara una librería de comunicaciones de Java que permita la comunicación con el puerto serie.

El applet utiliza las clases del Java Communications API 2.0 de Sun, que ofrece clases Java de comunicación con los puertos serie y paralelo. Para el correcto funcionamiento del applet es necesaria la instalación de los archivos de esta librería en el equipo del cliente.

Los archivos de esta librería están contenidos en el fichero comprimido *javacomm20-win32.zip*. El primer paso es descomprimir este archivo, lo cual devuelve, entre otros, los siguientes archivos:

- comm.jar
- win32.dll
- javax.comm.properties

Se deben instalar estos archivos siguiendo en las siguientes rutas a partir del path donde este instalado el JRE que utiliza el navegador, por ejemplo, en el caso del *JRE 1.5.0_05*:

- C:\Archivos de programa\Java\jre1.5.0_05\lib\ext\comm.jar
- C:\Archivos de programa\Java\jre1.5.0_05\bin\win32.dll
- C:\Archivos de programa\Java\jre1.5.0_05\lib\javax.comm.properties

Se deben instalar estos archivos siguiendo en las siguientes rutas a partir del path donde este instalado el JRE que utiliza el navegador, por ejemplo, en el caso del *JRE 1.5.0_05*.

Existe un archivo de configuración *C:\app_ejie\p12\config\p12.config* en el que se configura el puerto por defecto del que va a leer el applet los datos que envía el lector, a través del siguiente parámetro *defaultPort* (COM1 o COM2).

8.3 Inserción del applet.

A continuación se muestra un ejemplo del código HTML necesario para el funcionamiento del applet en una página. Se deben incluir unas funciones javascript que introducen los datos recogidos por el applet en los campos adecuados.

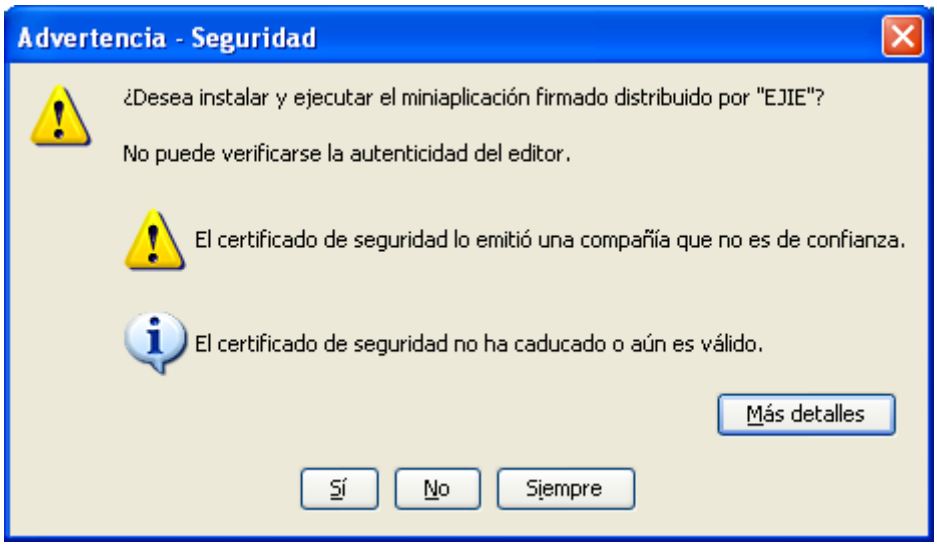
```
<html>
  <head>
    <title>Prueba Applet CreditCardReader</title>
    <script type="text/javascript">
        function resetCardData() {
            document.CardData.nombre.value="";
        document.CardData.numeroTarjeta.value="";
        }
        function printCardData(data1,data2,data3,data4) {
            Pos1 = data1.indexOf("^",0);
            Pos2 = data1.indexOf("^",Pos1+1);
            numeroTarjeta = data1.substring(1,Pos1);
            nombre = data1.substring(Pos1+1,Pos2);
            document.CardData.nombre.value=nombre;
            document.CardData.numeroTarjeta.value=numeroTarjeta;
            document.CardData.mesCaducidad.value=data3;
            document.CardData.anioCaducidad.value=data4;
        }
    </script>
  </head>

  <body onLoad="resetCardData()">
    <p><h1>Pase la tarjeta por el lector:</h1></p>
    <applet name="CreditCardReader" code="p12.exe.client.applet.CreditCardReader.class"
archive="CreditCardReader.jar" codebase="." height="0" width="0" MAYSCRIPT="true"></applet>
    <form name="CardData" >
        <table width="80%" align="center" border="0" cellspacing="0" cellpadding="0" >
            <tr>
                <td nowrap>Número de tarjeta</td>
                <td>
                    <input type="text" name='numeroTarjeta' tabindex='2' maxlength='16'
size='50' readOnly="true"/>
                </td>
            </tr>
            <tr>
                <td nowrap>Nombre Titular</td>
                <td>
                    <input type="text" name='nombre' tabindex='3' maxlength='30' size='50'
readOnly="true"/>
                </td>
```

```
            </tr>
            <tr>
                <td nowrap>Fecha de caducidad</td>
                <td>
                    <input type="text" name='mesCaducidad' maxlength='2' size='2'
readOnly="true"/>/<input type="text" name='anioCaducidad' maxlength='2' size='2'
readOnly="true"/>
                </td>
            </tr>
        </table>
    </form>
  </body>
</html
```

8.4 Ejecución del applet.

Una vez instalados estos archivos ya se puede ejecutar el applet sobre el equipo. Se trata de un applet firmado, por lo si el navegador no tiene instalado el certificado con el que se ha firmado el applet, al cargarlo mostrara la siguiente advertencia.



Pulsando en “Siempre”, el certificado se instalará en el navegador y la advertencia no volverá a aparecer.

9 FICHEROS DE CONFIGURACIÓN Y PROPIEDADES.

Para el correcto funcionamiento de las librerías es necesaria la instalación en los equipos de las Entidades Financieras de una serie de ficheros. Existen dos tipos de ficheros que deben ser instalados:

- Ficheros de configuración (*p12x.properties.xml*).
- Ficheros de ClassMap (*classMap.xml*).

9.1 Ficheros de configuración.

Indican propiedades y configuraciones de los módulos usados por las Entidades Financieras. Son los siguientes:

- *p12f.properties.xml*
- *p12e.properties.xml*
- *r01f.properties.xml*

Se colocarán bajo un directorio que esté en el classpath, dondese crearán los directorios *p12f*, *p12e* y *r01f*. Por ejemplo:

[directorio en classpath]/p12f/p12f.properties.xml

9.2 Fichero de ClassMap.

Este fichero sirve para transformar las estructuras XML a objetos y viceversa. Lleva el nombre de *classMap.xml*.

La ruta donde se debe colocar el fichero classMap se especificará en el fichero p12f.properties.xml, en la etiqueta *objectMapPath*:

<objectMapPath>/datosSoft/p12f/file/classMap.xml</objectMapPath>

10 Personas de Contacto

Por parte de la Administración Vasca intervienen en este proyecto las siguientes Entidades:

Entidad	Persona de Contacto
Dirección de Finanzas	Jose Antonio Ceciaga Aguirre e-mail:tesoreria@euskadi.eus Teléfono: 945018964
DACIMA (Dirección de Atención a la Ciudadanía e Innovación y Mejora de la Administración) Dirección y Coordinación del Proyecto	Juan Luis Ronco Rodrigo e-mail: ronco@euskadi.eus Telefono: 945062016
EJIE (Eusko Jaularitzaren Informatika Elkartea) Dirección Técnica del Proyecto	Alex Lara Garachana e-mail: a-lara@ejie.eus Teléfono: 945017473 Iker Olabarria Ozaeta e-mail : i-olabarria@ejie.eus Teléfono: 945017403
Correo de consulta	mipago@euskadi.eus