Pasarela de pagos de la Administración Pública Vasca

V2

Integración con Sistemas BackEnd de Administraciones

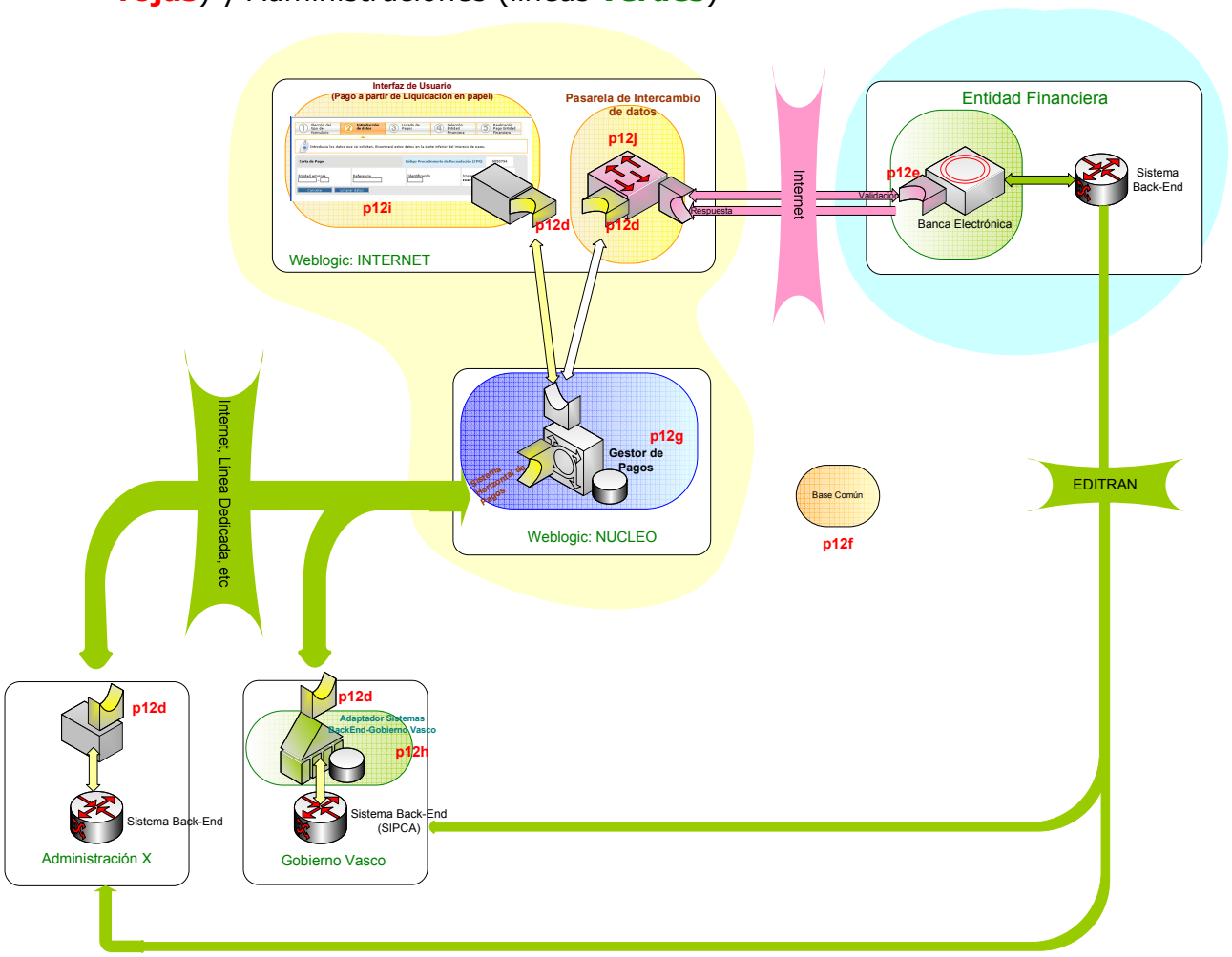
06 de Febrero de 2007

Indice

1 Introducción

El presente documento recoge las Especificaciones Funcionales para el modelo de integración de sistemas de las Administraciones con la Pasarela de Pagos (versión 2).

En la figura se observa la arquitectura de la Pasarela de Pagos y específicamente la integración de la misma con Entidades Financieras (líneas **rojas**) y Administraciones (líneas **verdes**)



La Pasarela de Pagos actúa de intermediario entre Administraciones y Entidades financieras para **operaciones on-line**:

- Pagos on-line
- Consulta del estado de Pagos on-line
- Cualquier otra operación on-line futura

De esta forma, las Administraciones envían peticiones de operaciones a la pasarela (pago, consulta, etc) y esta es la encargada de interactuar con las Entidades Financieras, devolviendo finalmente una respuesta a la Administración origen. En este documento se describe la integración de las Administraciones y la Pasarela de Pagos.

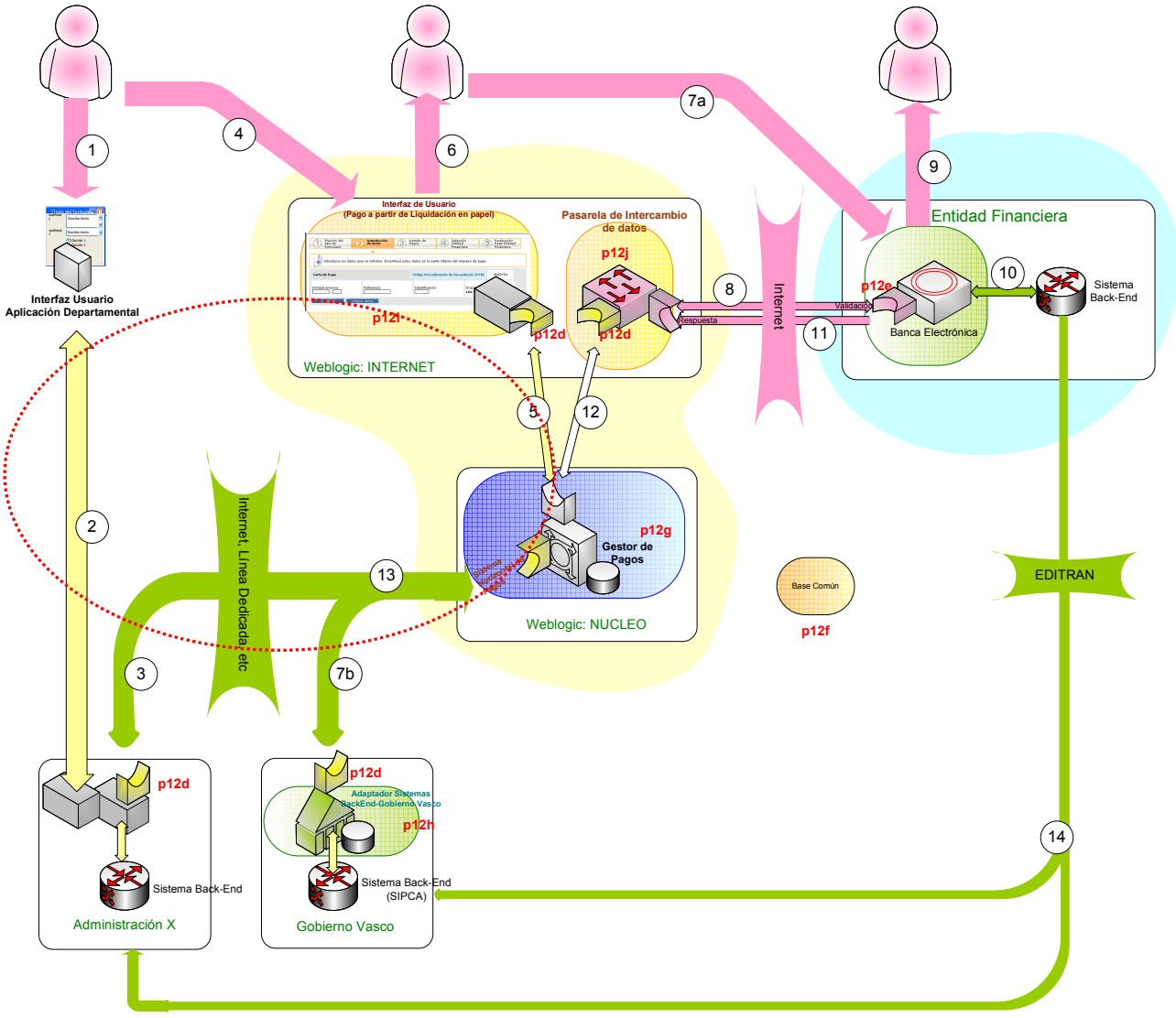
2 Módulos de la Pasarela de Pagos

p12d	**API para las Administraciones** Encapsula las funcionalidades que permiten a las Administraciones operar con **el núcleo** de pasarela de pagos: ▪ Enviar peticiones de pago ▪ Recibir notificaciones de los eventos ocurridos en la Pasarela acerca de los pagos. ▪ Enviar consultas de estado de pagos ▪ Cualquier otra operación que se implemente en un futro.
p12e	**API para las Entidades Financieras** Encapusula las funcionalidades que permiten a las Entidades Financieras operar con el módulo **de intercambio de datos** de la Pasarela de Pagos: ▪ Validar peticiones de pagos ▪ Enviar resultados de pagos tramitados en la Entidad Financiera ▪ Recibir y responder a consultas de estado de pagos ▪ Cualquier otra operación que se implemente en un futuro
p12f	**Framework común a todos los módulos de la Pasarela de Pagos** Contiene utilidades y funciones que se utilizan desde todos los módulos de la pasarela.
p12g	**Núcleo de la Pasarela de Pagos.** Lógica de negocio donde se controlan todas las operaciones que ocurren en la pasarela de pagos. Es el motor del sistema donde se encuentran implementadas las funcionalidades expuestas en los diferentes APIs.
p12h	**Módulo de integración de pagos del Gobierno Vasco**
p12i	**Interfaz gráfica de la Pasarela de Pagos** Interfaz de usuario (ciudadano) que permite identificar los pagos y seleccionar la Entidad Financiera

3 Proceso de Pago

Internamente, la Pasarela de Pagos se estructura según el siguiente esquema general en el que aparecen todas las piezas que componen la Pasarela de Pagos, sin embargo, de cara a la Integración con las Administraciones, únicamente entran en juego las señaladas en un círculo **rojo**.

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*.



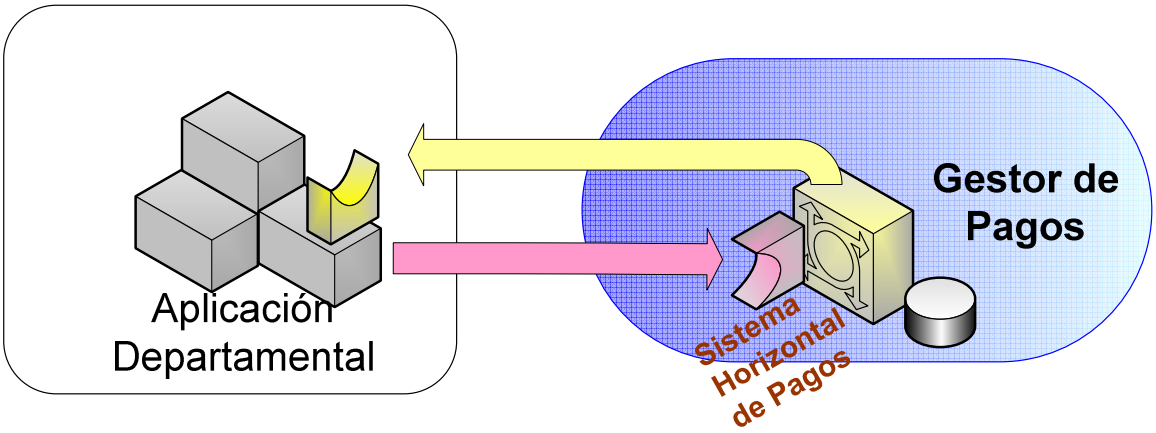
En la siguiente tabla, se describe cada uno de los pasos del pago señalados con un número en el gráfico.

Paso	Fase	Descripción
1	Tramitación	Un ciudadano/a tramita utilizando una aplicación departamental de un Administración X, 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. NOTA: 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, la Pasarela de Pagos informa al sistema origen. NOTA: Normalmente, el sistema origen utiliza esta notificación para 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*) 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 que responde a la Entidad Financiera diréctamente.
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*). **NOTA**: En este envío de cobros consolidados, la Pasarela de Pagos es transparente (no interviene en ningún caso)
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.
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 notifica 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**)

4 Intercambio de Mensajes

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



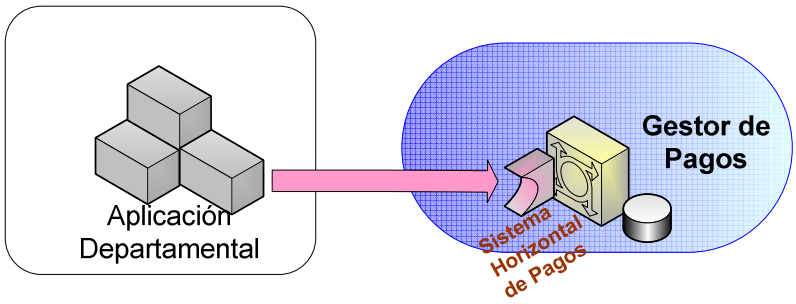
El **Gestor de Pagos** (núcleo de la pasarela de pagos) **expone** varias funciones para que las aplicaciones departamentales envíen nuevos pagos, consulten el estado de otros ya enviados, etc. (módulo rojo)

Por su parte, las **Aplicaciones Departamentales exponen** a su vez funciones para que el Gestor de Pagos **notifique** a la aplicación **eventos de pago** (módulo amarillo)

Como se observa en la figura, funcionalmente, en ambos lados existe un módulo que actúa de **conector**, su función es abstraer a la aplicación departamental y al gestor de pagos del medio de comunicación, protocolos, etc.

4.1 Funciones Expuestas por el Gestor de Pagos

Las Aplicaciones Departamentales "ven" las siguientes funciones lógicas del Gestor de Pagos, o lo que es lo mismo, el Gestor de Pagos "expone":



Interfaz Gestor de Pagos (núcleo)	
Operación	**Descripción**
Inicializar Pago(s)	Envía todos los datos de uno/varios pagos a la Pasarela de Pagos para que puedan ser tramitados
Comenzar Pago(s)	Indica que uno/varios pagos que han sido previamente inicializados van a ser pagados a través de la pasarela.
Cancelar Pago	Cancela un pago en la pasarela, siempre que este no se haya comenzado a tramitar on-line u off-line.
Consultar Estado Pago	Permite 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.

IMPORTANTE

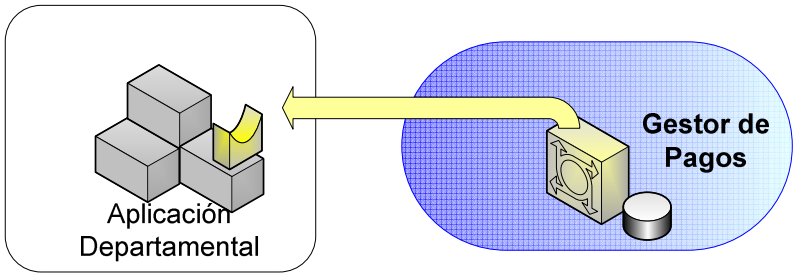
- Todas las operaciones anteriores son **síncronas** (petición – respuesta inmediata).

- Este interfaz es expuesto por el Gestor de Pagos en base a **servicios accesibles vía web**, aunque en un futuro serán también ofrecidos como servicios web (WebServices). Sin embargo, para abstraer al programador de la forma de acceder a los servicios se dispone de un API (P12D) que encapsula todas las llamadas, permitiendo por ejemplo en un futuro la llamada vía webServices de forma transparente y sin necesidad de retocar código.

- Los servicios intercambiarán estructuras XML que **son de obligado cumplimiento** por todas las Administraciones. Sin embargo, para evitar al programador de la complejidad de la construcción de XML se ofrecen objetos Java / Microsoft que facilitan la programación y minimizan los errores.

4.2 *Funciones expuestas por las Aplicaciones Departamentales*

El Gestor de Pagos "ve" las siguientes funciones lógicas de las Aplicaciones Departamentales, o lo que es lo mismo, las Aplicaciones Departamentales "exponen" **listeners a eventos de pagos**:



Interfaz Aplicaciones Departamentales	
Evento	**Descripción**
Pago Preinicializado (opción para el Ayto. de Vitoria)	Indica que se ha introducido **un pago** en el formulario del interfaz de usuario de la pasarela. **NOTA**: La entidad emisora que recibe el evento podría en este momento actualizar los datos del pago, teniendo en cuenta posibles modificaciones que este pago haya sufrido desde que fue emitida su orden de pago. Esta modificación no es algo habitual y debe consensuarse con las entidades financieras colaboradoras de esta entidad emisora.
Pago Inicializado	Indica que **un pago** ha sido inicializado en el Gestor de Pagos. **NOTA**: Así como una aplicación departamental puede inicializar un lote de pagos, los eventos de pagos inicializados se reciben **individualmente** para cada uno de los pagos.
Pago Comenzado	Indica que uno/varios pagos que han sido previamente inicializados van a ser pagados a través de la pasarela.
Pago tramitado on-line	Indica que un pago se va tramitar on-line en la banca electrónica de alguna de las Entidades Financieras Colaboradoras
Pago tramitado off-line	Indica que un pago se va a tramitar off-line: se ha impreso la liquidación para ser pagada en una ventanilla o cajero automático.
Validación de Pago OK Validación de Pago NOK	Indica que un pago ha sido recibido en la banca electrónica de una Entidad Financiera y este ha sido validado correctamente / incorrectamente contra la Pasarela de Pagos. **NOTA**: Este evento únicamente es indicativo de que el ciudadano ha entrado en la Banca Electrónica de la Entidad Financiera: aún no se ha procedido al pago.

Pago on-line OK Pago on-line NOK	Indica que un pago ha sido pagado correcta / incorrectamente en la banca Electrónica de una Entidad Financiera
Pago Cancelado	Indica que un pago ha sido cancelado en la pasarela, siempre que este no se haya comenzado a tramitar on-line u off-line.

IMPORTANTE

- Todas las operaciones anteriores son **asíncronas**: el Gestor de Pagos llamará (lanzará los eventos) cuando se produzcan las condiciones que los desencadenan.

- Este interfaz es expuesto por las Aplicaciones Departamentales de la forma / tecnología en que estas consideren necesario (http, colas, ficheros, etc). Por defecto se proporciona una implementación de llamada en base a http aunque en un futuro próximo se implementarán callbacks en base a colas, ficheros, etc.

- Los servicios intercambiarán XML que **son de obligado cumplimiento** por todas las Administraciones. Sin embargo, para evitar al programador de la complejidad de la construcción de XML se ofrecen objetos Java que facilitan la programación y minimizan los errores.

4.3 Acceso al Interfaz expuesto por el Gestor de Pagos

El Gestor de Pagos expone sus servicios a las aplicaciones departamentales en base a servicios accesibles vía web (en un futuro se soportarán webServices).

Para utilizar estos servicios, hay dos formas:

http	Cada una de las funciones expuestas por el gestor de pagos tiene una url con unos parámetros normalizados en forma de XML
API	Se dispone de un API Java / Microsoft que encapsula la llamada a los servicios, abstrayendo al programador de la implementación concreta (funciones web, servicios web, etc)

4.3.1 Acceso vía http

Todos los servicios expuestos por el Gestor de Pagos son accesibles invocando a unas URLs y pasando una serie de parámetros normalizados vía POST.

Para facilitar futuros desarrollos tanto por parte de las Pasarela 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 y a modo de **recomendación** 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.pasarela.com/xWar/yServlet
https://www.pasarela.com/pasarela/x.asp
https://www.pasarela.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 las aplicaciones departamentales, la Pasarela de Pagos o las 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 desde una aplicación departamental
Valor	Nombre identificador de la función. En base a las funciones de las interfaces para la Pasarela expuestas justo antes, los nombres serán: ▪ ***initializePayment***: Enviar los datos de uno/varios pagos (XML) al núcleo de la Pasarela de Pagos ▪ ***cancelPayment:*** Cancela un pago que aún no ha sido tramitado ni on-line, ni off-line ▪ ***getPaymentStateData:*** Obtiene el estado de un pago: pagado, emitida liquidación, etc.
Ejemplo	→ *function="getPaymentData"*

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

Parámetro	Descripción
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

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:

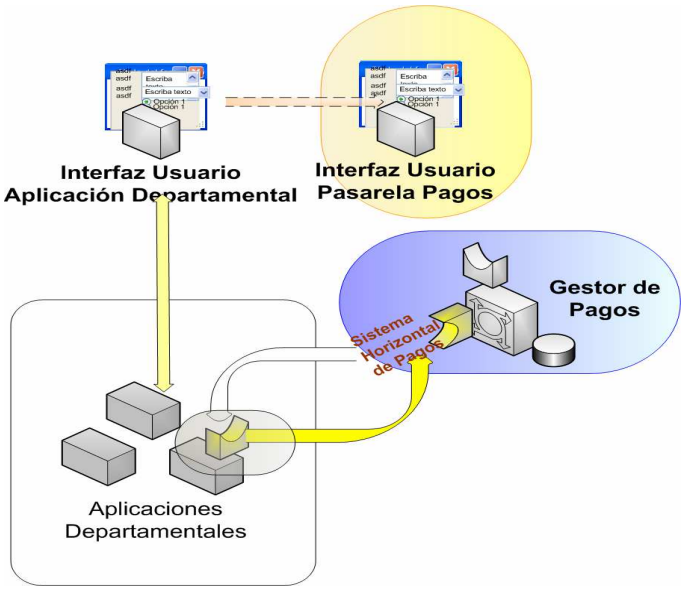
Funcionalidad	Servicio	Módulo	Función	Parámetros	Respuesta
Inicializa un pago en la Pasarela	*Gestor de Pagos*	*-nada-*	*initializePayment*	paymentRequestData: Datos de las peticiones de pago en formato XML protocolData: Datos de protocolo de la llamada	XML *la estructura* ***paymentRequestData***
Obtener el estado de un pago	*Gestor de Pagos*	*-nada-*	*getPaymentStateData*	paymentId: Identificador del pago cpr: Código de procedimiento recaudatorio paymentCode: Ráfaga en el cuadernillo 57 o 60 del pago nrc: NRC del pago protocolData: Datos de protocolo de la llamada	XML *la estructura* ***paymentStateData*** con el estado del pago
Cancelar un pago	*Gestor de Pagos*	*-nada-*	*cancelPayment*	paymentId: Identificador del pago	XML *la estructura* ***paymentStateData*** con el estado del pago

A continuación se detalla cada una de las funciones:

4.3.1.1 initializePayment: Inicialización de un Pago

Normalmente los pagos se generan en aplicaciones departamentales que para posibilitar su pago utilizando la Pasarela de Pagos han de ser registrados **previamente** en el Gestor de Pagos.

El método ***initializePayment*** realiza la validación de los pagos enviados por las aplicaciones y los registra en el Gestor de Pagos. A partir de este momento, el ciudadano puede entrar en cualquier momento al interfaz de usuario de la Pasarela de Pagos y finalizar el pago.



En la figura se observa como el usuario interactuando con una aplicación departamental genera un pago que es dado de alta en el Gestor de Pagos. Una vez inicializado el pago, el usuario es redirigido al interfaz de usuario de la Pasarela de Pagos donde finaliza el proceso.

Origen	**Servicio** de la Aplicación Departamental
Destino	**Servicio** del Gestor de Pagos de la Pasarela
Método HTTP	POST
Función	function=*initializePayment* *Parámetros:* paymentRequestData: Datos de las peticiones de pago en formato XML protocolData: Datos de protocolo
Respuesta	*estructura* ***operationResult*** que contiene la estructura ***paymentStateData*** en su campo ***returnValue***.
Ejemplo	Llamada vía **POST**: http://www.testpago.euskadi.net/p12gWar/p12gRPCDispatcherServlet? **function**= *initializePayment*& **paymentRequestData**=[XML de la petición de pago]& **protocolData**=[XML protocolo]

Después de la inicialización de un nuevo pago, la Aplicación Departamental debe recoger los identificadores de los pagos devueltos por el Gestor de Pagos para solicitar el comienzo del proceso de pago en el interfaz de usuario de la Pasarela de Pagos.

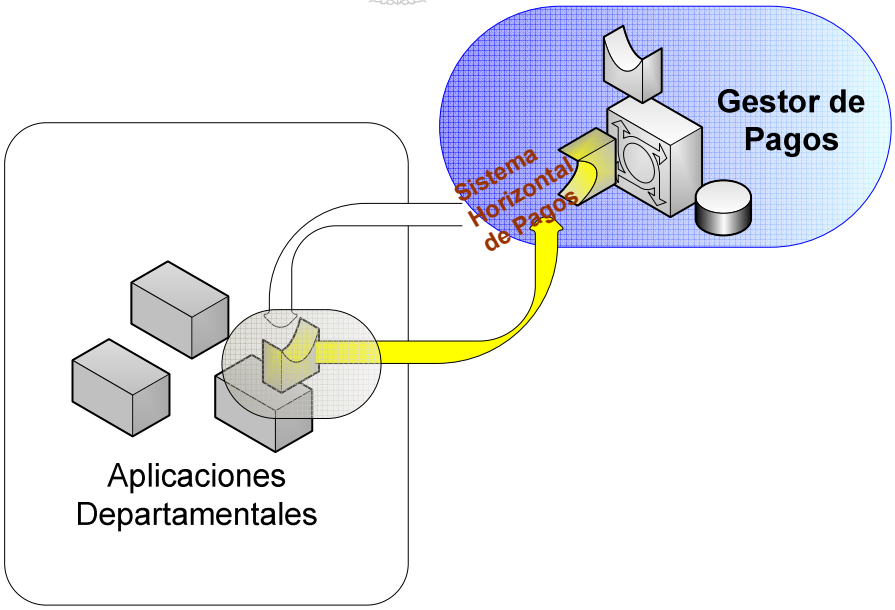
NOTA:

*Cuando la Aplicación Departamental construye el mensaje de petición de pago debe rellenar una serie de campos, entre ellos el del identificador de pago. Aunque **no** es capaz de construir identificadores en el formato que maneja la pasarela, debe por lo menos dar un valor distinto a los identificadores de cada uno de los pagos. Cuando los pagos sean validados, la pasarela le devolverá una estructura con los datos de los pagos, de donde la aplicación puede recoger los identificadores de los pagos que habrá asignado la pasarela con el formato correcto.*

4.3.1.2 getPaymentStateData: Consulta del Estado de un Pago

La función ***getPaymentStateData*** solicita el estado de un pago al Gestor de la Pasarela que puede ser uno de los siguientes:

preinicializado	Un usuario ha introducido los datos del pago en el formulario de introducción de datos del interfaz de usuario de la Pasarela.
inicializado	La aplicación departamental ha solicitado la inicialización del pago, sin embargo, el ciudadano **no** ha entrado en la interfaz de usuario de la Pasarela de Pagos para realizar el pago
emitida liquidación	El usuario ha utilizado la opción de pago offline imprimiéndose una liquidación con código de barras para ser pagada en una ventanilla de una Entidad Financiera.
acceso a banca electrónica	El usuario ha escogido pagar en la Banca Electrónica de una entidad financiera
pago validado OK / NOK	El ciudadano ha entrado en la banca electrónica de una entidad financiera y el pago ha sido validado contra la Pasarela de Pagos de forma satisfactoria o errónea si existe alguna diferencia entre el pago que hay en la Pasarela de Pagos y el que ha llegado a la Entidad Financiera
pago on-line OK / NOK	El ciudadano ha pagado en la banca electrónica de una Entidad Financiera y el proceso ha ido correcta o erróneamente.
pago cancelado	El pago ha sido cancelado antes de haber sido pagado o de que el ciudadano se haya impreso una liquidación con código de barras.



Origen	**Servicio** Aplicación Departamental
Destino	**Servicio** del Gestor de Pagos de la Pasarela
Método HTTP	POST
Función	function=*getPaymentStateData* *Parámetros:* paymentId: Identificador del pago
Respuesta	*estructura* ***operationResult*** que contiene la estructura ***paymentStateData*** en su campo ***returnValue***.
Ejemplo	Llamada vía **POST**: http://www.testpago.euskadi.net/p12gWar/p12gRPCDispatcherServlet? **function**= *getPaymentStateData*& **paymentId**=[paymentId]

4.3.2 Acceso utilizando el API

El API del Gestor de Pagos (**P12D y P12F**) abstrae de la complejidad comunicación http (o de cualquier otro método que se implemente en un futuro –webServices-), exponiendo una interfaz programática Java / Microsoft, muy sencilla de utilizar.
El API tiene dos partes:

P12D	▪ Acceso a las funciones del Gestor de Pagos para Aplicaciones Departamentales
P12F	▪ Objetos comunes que abstraen de la construcción de XML ▪ Utilidades comunes

NOTA: *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 concreto, utilizando estas librerías es posible:

1. Invocar las funciones expuestas por la Pasarela de Pagos desde una Aplicación Departamental (API de comunicación con la Pasarela contenido en el paquete ***p12d.exe.pasarelapagos.services***).

2. Interpretar las estructuras de datos XML (el paquete ***p12f.exe.pasarelapagos.objects*** *visto* el punto 5 proporciona un conjunto de clases que encapsulan la información que van a manejar las Aplicaciones Departamentales).

La utilización del API es extremadamente sencilla y se resume en los siguientes pasos:

1. Obtener una instancia del API a partir de una factoría de APIs
2. Componer los objetos que se pasan como parámetros
3. Invocar a la función correspondiente del API

Las funciones expuestas por el Gestor de Pagos están expuestas en la clase p12d.exe.pasarelapagos.services.P12DPaymentManagerAPI

Para obtener una instancia de esta clase basta con utilizar la factoría de APIs p12d.exe.pasarelapagos.services.P12DPaymentFactoryAPI

```
P12DPaymentManagerAPI paymentAPI  = P12DPaymentFactoryAPI.getPaymentManagerAPI();
```

Una vez instanciado el API se puede invocar a cualquiera de sus métodos:

initializePayment Inicializa un lote de pagos en el gestor de pagos	Parámetros de Entrada *PaymentRequestData* Parámetros de Salida *OperationResult*	paymentRequestData Datos de los pagos que se van a inicializar Objeto que indica el resultado de la operación contra el gestor de pagos (OK / NOK) y que encapsula en el miembro returnValue el valor devuelto por el Gestor de Pagos: ▪ Si el resultado es OK, el mismo objeto PaymentRequestData pero ya inicializado en el Gestor de Pagos ▪ Si el resultado es NOK, la razón del error.
cancelPayment Cancela un pago en el Gestor de Pagos, siempre que este no haya sido ya comenzado a tramitar (emitida liquidación o enviado a una Entidad Financiera)	Parámetros de Entrada *String* Parámetros de Salida *OperationResult*	paymentOid Identificador del pago a cancelar Objeto que indica el resultado de la operación contra el gestor de pagos (OK / NOK). En este caso en el miembro *returnValue* no viene nada a no ser que el resultado haya sido NOK en cuyo caso se indican las razones.
getPaymentStateData Obtiene el estado de un pago	Parámetros de Entrada *String* Parámetros de Salida *OperationResult*	paymentOid Identificador del pago cuyo estado se quiere conocer Objeto que indica el resultado de la operación contra el gestor de pagos (OK / NOK). ▪ Si el resultado es OK, un objeto Estado ▪ Si el resultado es NOK, la razón del error.

sendUsingPOST ***sendUsingGET*** Método de **utilidad para las aplicaciones Web** que envía a la pasarela un pago que previamente se ha inicializado mediante el método *initializePayment* y permite al ciudadano iniciar la operativa de pago	Parámetros de Entrada PaymentGatewayData Estructura que encapsula: **PaymentRequestData** Datos del pago **PresentationData** Datos de presentación como las entidades financieras permitidas o los modos de pago (off-line / on-line) permitidos **ProtocolData** Datos de protocolo como urls de vuelta, identificadores de sesión, etc Parámetros de Salida *OperationResult* Objeto que indica el resultado de la operación (OK / NOK).

4.3.3 Ejemplo: Solicitud de un pago a la Pasarela de Pagos desde la aplicación Departamental

En este apartado se pretende mostrar a través de un ejemplo el desarrollo a llevar a cabo por la Aplicación Departamental, utilizando las clases provistas por el SDK de la Pasarela de Pagos.

Los pasos a dar, en grandes líneas se resume en la siguiente tabla:

PASO 1	Instanciación de los objetos de pago, en especical, un objeto ***PaymentRequestData*** que encapsula los datos de las peticiones de pago que se quieren realizar.
PASO 2	Enviar la estructura ***PaymentRequestData*** al Gestor de Pagos para su validación y registro en el sistema de las peticiones de pago que encapsula.
PASO 3	Recoger los oid de las peticiones de pago de la respuesta del Gestor de Pagos redirigir al ciudadano al Interfaz de Usuario de la Pasarela para que realice el pago on-line. En esta redirección además de los datos del pago (*PaymentRequestData*) se envían: ProtocolData: Datos de protocolo: urls, identificadores de sesión web, etc PresentationData: Datos de presentación: ▪ Medios de Pago permitidos ▪ Entidades Financieras permitidas ▪ etc

El siguiente servlet es un ejemplo de todo el proceso:

```
package p12d.test;


import java.io.IOException;
import java.util.*;

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

import p12d.exe.pasarelapagos.exceptions.*;
import p12d.exe.pasarelapagos.services.*;
import p12f.exe.pasarelapagos.exceptions.*;
import p12f.exe.pasarelapagos.objects.*;
import p12f.exe.pasarelapagos.paymentrequest.*;

import com.ejie.r01f.xml.marshalling.XOMarshallerException;
import com.ejie.r01f.xmlproperties.XMLProperties;

/**
 * Servlet de ejemplo de introducción de un pago en la pasarela.
 */
public class p12dTestServlet extends HttpServlet {
////////////////////////////////////////////////////////////////////
// CONSTRUCTOR / DESTRUCTOR
////////////////////////////////////////////////////////////////////
    public p12dTestServlet() {
        super();
    }
    public void destroy() {
         super.destroy();
    }
////////////////////////////////////////////////////////////////////
// INTERFAZ HttpServlet
////////////////////////////////////////////////////////////////////
    public void init() throws ServletException {

    }
    public void doPost(HttpServletRequest request, HttpServletResponse response)
              throws ServletException, IOException {
         try {
           _doExec(request,response);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
              throws ServletException, IOException {
         try {
           _doExec(request,response);
        } catch (Exception e) {
           e.printStackTrace();
        }
    }
////////////////////////////////////////////////////////////////////
// METODOS PRIVADOS: Implementación
////////////////////////////////////////////////////////////////////
    private void _doExec ( HttpServletRequest request,
                           HttpServletResponse response) throws ServletException,
                                                                IOException,
                                                                ConfigLoadException,
                                                                P12DPaymentRequestException,
                                                                XOMarshallerException {
    // PAS0 1: ---- Componer los objetos de pago
        PaymentRequestData paymentRequestData = new PaymentRequestData();
        paymentRequestData.peticionesPago = new java.util.HashMap();

        PeticionPago peticionPago507 = _composePeticionPago507();
        paymentRequestData.peticionesPago.put(peticionPago507.id,peticionPago507);

    // PAS0 2: ---- Enviar la petición de pago al Gestor de Pagos utilizando el API
         // Instanciar el API y llamar al método initializePayment
         P12DPaymentManagerAPI paymentAPI  = P12DPaymentFactoryAPI.getPaymentManagerAPI();
         OperationResult opResult = paymentAPI.initializePayment(paymentRequestData);
         // Obtener la respuesta del Gestor de Pagos encapsulada en el objeto operationResult
        PaymentRequestData returnedPymntReqData = null;
        if(opResult.resultado.resultadoOK)
              returnedPymntReqData = PaymentRequestData.getObject(opResult.resultado.returnValue);

        if(returnedPymntReqData  == null
        || returnedPymntReqData.peticionesPago == null
        || returnedPymntReqData.peticionesPago.size()==0) {
            throw new ServletException();
        }
```

```
// PAS0 3: ---- Redirigir al ciudadano a la Pasarela de Pagos
//              Para ello componer un objeto PaymentGatewayData que engloba:
//                  - PaymentRequestData: Datos de Pago
//                  - ProtocolData: Datos de Protocolo
//                  - PresentationData: Datos de Presentación
    PaymentGatewayData dataToSend  =  new PaymentGatewayData();
    dataToSend.paymentRequestData  = returnedPymntReqData;
    dataToSend.protocolData =  _composeProtocolData().toXML();
    dataToSend.presentationData = _composePresentationData().toXML();
    operationResult = paymentAPI.sendUsingPost(dataToSend,request,response);
}

/**
 * Compone una petición de pago de ejemplo.
 */
private PeticionPago composePeticionPago507 () {

     PeticionPago peticionPago = new PeticionPago();

     /*****************  DATOS PETICION PAGO ************************ */
     DatosPago datosPeticionPago = new DatosPago();
     datosPeticionPago.formato = "507";
     datosPeticionPago.emisor =  "04833001-514";
     datosPeticionPago.cpr     = "9050794";
     datosPeticionPago.validar = 1;
     datosPeticionPago.tipo = "514";
     datosPeticionPago.referencia = "1234567890123";

     /***************** PERIODOS PETICION DE PAGO *****************/
     datosPeticionPago.periodosPago = new HashMap();
     PeriodoPago periodoPago = new PeriodoPago();
     periodoPago.id = PeriodoPago.PERIODO_NORMAL;
     periodoPago.fechaFin =    "010806";
     periodoPago.fechaInicio = "010606";
     periodoPago.importe = 1160;
     periodoPago.identificacion = "120307";
    periodoPago.activo = true;
     datosPeticionPago.periodosPago.put(periodoPago.id,periodoPago);

     /***************** CONCEPTOS PETICION DE PAGO *****************/
     peticionPago.conceptos = _getConceptosPeticionPago();

     /*************  DESCRIPCION PETICION DE PAGO **************************/
     Map descripciones = new HashMap();
     descripciones.put("es", "Aplicacion de Prueba  CSB 507");
     descripciones.put("eu", "Aplicacion de Prueba _EU  CSB 507");
     peticionPago.descripcion = descripciones;

     peticionPago.id = datosPeticionPago.cpr +
                       datosPeticionPago.emisor +
                       datosPeticionPago.referencia;
     peticionPago.datosPago = datosPeticionPago;

     return peticionPago;
}
```

```
    /**
     * Funcion _getConceptosPeticionPago ()
     * Funcion que devuelve un MAP de dos Conceptos de Pago para el siguiente ejemplo
     */
    private static java.util.List _getConceptosPeticionPago(){

         java.util.List conceptosPeticionPagoMap = new ArrayList();

         ConceptoPeticion conceptoPeticionMatricula  = new ConceptoPeticion();
         conceptoPeticionMatricula.numeroLinea =1;
         conceptoPeticionMatricula.descripcion.put("es","Matricula de seminario");
         conceptoPeticionMatricula.descripcion.put("eu","Matricula de seminarios");
         conceptoPeticionMatricula.unidades=1;
         conceptoPeticionMatricula.baseImponible=1000;
         conceptoPeticionMatricula.importeIVA=160;
         conceptoPeticionMatricula.importe =1160;
         conceptosPeticionPagoMap.add(conceptoPeticionMatricula);

         return    conceptosPeticionPagoMap;

    }
    /**
     * Obtiene la información de protocolo del pago
     */
    private static java.util.List _composeProtocolData(){
       // Componer la estructura con los datos de Protocolo como la url
        // a la que hay que volver después de pagar
         ProtocolData protocolData = new ProtocolData();
         protocolData.responseURL="";
         protocolData.sourceSessionId = (request.getSession()).getId();
         protocolData.timeStamp = new Long(System.currentTimeMillis()).toString();
         protocolData.sourceOperationNumber = "1";
         Url urlVuelta = new Url(Url.VUELTA,
                                 XMLProperties.get("p12d","urls/urlVuelta")); // url vuelta por defecto
         protocolData.urls.put(urlVuelta.id, urlVuelta);
         return protocolData;
    }
    /**
     * Obtiene la información de presentación de los pagos en el interfaz
     * de usuario de la Pasarela de Pagos
     */
    private static java.util.List _composePresentationData(){
         // Componer la estructura con los datos de Presentación
         PresentationRequestData presentationRequestData = new PresentationRequestData();
         presentationRequestData.idioma = "es";
         PaymentMode modeON = new  PaymentMode(PaymentMode.ONLINE);
         PaymentMode modeOFF = new  PaymentMode(PaymentMode.OFFLINE);
         presentationRequestData.paymentModes.put(modeON.oid, modeON);
         presentationRequestData.paymentModes.put(modeOFF.oid, modeOFF);
         return presentationData;
    }


}
```

4.4 Acceso al Interfaz Expuesto por las Aplicaciones Departamentales

El Gestor de Pagos **"lanza" eventos** ante determinadas situaciones ocurridas en el proceso de pago:

Evento	Descripción
Pago Preinicializado (opción para el Ayto. de Vitoria)	Indica que se ha introducido **un pago** en el formulario del interfaz de usuario de la pasarela. **NOTA**: La entidad emisora que recibe el evento podría en este momento actualizar los datos del pago, teniendo en cuenta posibles modificaciones que este pago haya sufrido desde que fue emitida su orden de pago. Esta modificación no es algo habitual y debe consensuarse con las entidades financieras colaboradoras de esta entidad emisora.
Pago Inicializado	Indica que **un pago** ha sido inicializado en el Gestor de Pagos. Este evento es lanzado cuando un pago enviado por una Aplicación Departamental se inicializa en el Gestor de Pagos posibilitando de esta forma su tramitación a través de la Pasarela de Pagos. **NOTA**: Así como una aplicación departamental puede inicializar un lote de pagos, los eventos de pagos inicializados se reciben **individualmente** para cada uno de los pagos.
Pago Comenzado	Indica que uno/varios pagos que han sido previamente inicializados van a ser pagados a través de la pasarela. Es el momento en el que el ciudadano accede al interfaz de usuario de la Pasarela de Pagos para comenzar su tramitación.
Pago tramitado on-line	Indica que un pago se va tramitar on-line en la banca electrónica de alguna de las Entidades Financieras Colaboradoras.
Pago tramitado off-line	Indica que un pago se va a tramitar off-line: se ha impreso la liquidación para ser pagada en una ventanilla o cajero automático.
Validación de Pago OK Validación de Pago NOK	Indica que un pago ha sido recibido en la banca electrónica de una Entidad Financiera y este ha sido validado correctamente / incorrectamente contra la Pasarela de Pagos. **NOTA**: Este evento únicamente es indicativo de que el ciudadano ha entrado en la Banca Electrónica de la Entidad Financiera: aún **no** se ha procedido al pago.
Pago on-line OK Pago on-line NOK	Indica que un pago ha sido pagado correcta / incorrectamente en la banca Electrónica de una Entidad Financiera
Pago Cancelado	Indica que un pago ha sido cancelado en la pasarela, siempre que este no se haya comenzado a tramitar on-line u off-line.

Las Aplicaciones Departamentales pueden "escuchar" estos eventos lanzados por el Gestor de Pagos para hacer cualquier operación que consideren necesaria.

Para que el Gestor de Pagos lance eventos a un listener, es necesario registrar el listener en la configuración del Gestor de Pagos. Esta configuración se hace a nivel de emisor-sufijo.

IMPORTANTE! El Gestor de Pagos lanza eventos a un ***listener*** (punto de escucha) para cada pareja emisor-sufijo. Esto implica que si varias aplicaciones distintas comparten un único sufijo para

un emisor, el punto de escucha (*listener*) será común, debiendo residir la lógica para informar a cada aplicación en el propio listener.

Adicionalmente, se puede filtrar aún más qué eventos desea la Aplicación Departamental escuchar, es decir, solamente se lanzarán los eventos a los cuales la aplicación departamental se ha "suscrito".

Todos los eventos devuelven una estructura con los siguientes datos:

Tipo de Evento	El tipo de evento lanzado (ver tabla anterior)
TimeStamp	Momento en el que se ha lanzado el evento en la Pasarela de Pagos
Mensajes	Mensajes asociados al evento
Identificador del Objeto	Identificador del objeto pago al que se refiere el evento
Datos del Pago	Datos del pago asociado al evento.
Estado del Pago	Estado del pago asociado al evento, enviado cuando el evento producido suponga un cambio en el estado del pago en la Pasarela.

Un listener es una clase que verifica un interfaz determinado: **GatewayEventListener**.

```
 /**
  * Preinicialización de un pago en el Gestor de Pagos. Permite que la
  * Aplicación Departamental que recibe el evento actualice los datos
  * del pago si fuese necesario.
  * @param evt datos del evento
  * @return Resultado de la operacion
  */
 public InitializeCSBPaymentResult onInitializeCSBPayment(GatewayEvent evt);
/**
 * Inicialización de un pago en el Gestor de Pagos desde una Aplicación Departamental
 * @param evt datos del evento
 */
public void onInitializePayment(GatewayEvent evt);
/**
 * Un ciudadano accede al interfaz de usuario de la Pasarela de Pagos
 * @param evt datos del evento
 */
public void onBeginPayment(GatewayEvent evt);
/**
 * Se cancela un pago que se había registrado en la pasarela y que aún no se
 * había comenzado a tramitar on-line / off-line
 * @param evt datos del evento
 */
public void onCancelPayment(GatewayEvent evt);
/**
 * El ciudadano se imprime la liquidación de un pago para poder pagarlo off-line
 * en una ventanilla / cajero de una Entidad Financiera
 * @param evt datos del evento
 */
public void onPayOFFLine(GatewayEvent evt);
/**
 * El ciudadano decide paga on-line en la banca electrónica de una entidad financiera
 * @param paymentOid Identificador de pago que se paga on-line
 */
public void onPayONLine(GatewayEvent evt);
/**
 * El ciudadano entra en la banca electrónica de una entidad financiera y esta envía
 * el pago a la Pasarela de Pagos para su validación resultando esta correcta.
 * @param evt datos del evento
 */
public void onValidatePaymentOK(GatewayEvent evt);
/**
 * El ciudadano entra en la banca electrónica de una entidad financiera y esta envía
 * el pago a la Pasarela de Pagos para su validación resultando esta incorrecta.
 * NOTA: No indica que se haya pagado todavía
 * @param evt datos del evento
 */
public void onValidatePaymentNOK(GatewayEvent evt);
```

```
/**
 * Un pago se ha realizado en la banca electrónica de una entidad financiera de forma correcta
 * @param paymentStateData los datos de estado del pago
 */
public void onPayONLineOK(GatewayEvent evt);
/**
 * Un pago se ha realizado en la banca electrónica de una entidad financiera de forma incorrecta
 * @param evt datos del evento
 */
public void onPayONLineNOK(GatewayEvent evt);
```

IMPORTANTE! Como se puede observar en el interfaz anterior, **todas las notificaciones** (*eventos*) **se lanzan a nivel de pago**, a pesar de que la pasarela soporta el tratamiento de lotes de pagos.

En resumen, la Pasarela de Pagos soporta lotes de pagos únicamente en el **interfaz de aplicaciones**, internamente la pasarela **no trata lotes de pagos**.

4.4.1 Implementacion del Interfaz GatewayEventListener

Para que el Gestor de Pagos sea capaz de invocar a un listener de eventos de una Aplicación Departamental, es necesario suministrar una clase java que implemente el interfaz ***ServerGatewayEventListener*** y que "conozca" la forma de hacer llegar el evento a la Aplicación Departamental:

El Gestor de Pagos "conoce" únicamente la clase y tiene la certeza de que esta implementa los métodos del interfaz *ServerGatewayEventListener* así que los invocará en los momentos adecuados. Será la clase proporcionada por la Aplicación Departamental la encargada de "propagar" el evento por los medios que sea: http, colas de mensajes, ficheros, etc.

El propio Gestor de Pagos tiene implementados varias formas de invocar a la Aplicación Departamental:

ClassGatewayEventListenerImpl Adecuado cuando el listener está en el mismo servidor de aplicaciones que el Gestor de Pagos. *Normalmente esta implementación es únicamente para test.*	Simplemente invoca a una clase java plana (POJO). En la configuración del listener en el emisor-sufijo se indica el nombre de la clase a invocar
RPCServletGatewayEventListenerImpl Adecuado cuando el listener es remoto y solo puede ser accedido vía http	Invoca a un módulo RPC Servlet vía http En la configuración del listener en el emisor-sufijo se indica la url del servlet RPC a invocar
EJBGatewayEventListenerImpl Adecuado cuando el listener es un EJB NO implementado	Invoca a la interfaz remota de un EJB
MessageQueueGatewayEventListenerImpl NO implementado	Envía un mensaje a una cola. La clase implementadota será la encargada de componer el mensaje y enviarlo a la cola

A continuación se describe la invocación de cada una de las formas anteriores:

4.4.2 Implementación en base a una clase Java Plana

```
package p12d.test.eventlistener;

import p12d.exe.pasarelapagos.eventlistener.GatewayEventListener;
import p12f.exe.pasarelapagos.objects.DatosPago;
import p12f.exe.pasarelapagos.objects.GatewayEvent;
import p12f.exe.pasarelapagos.objects.Mensaje;
import p12f.exe.pasarelapagos.paymentrequest.InitializeCSBPaymentResult;
import p12f.exe.pasarelapagos.paymentrequest.PeticionPago;

public class ClassGatewayEventListenerImpl implements ClassGatewayEventListener{

 /**
 * Preinicialización de un pago en el Gestor de Pagos. Permite que la Aplicación Departamental que
 * recibe el evento actualice los datos del pago si fuese necesario.
 * @param evt datos del evento
 * @return Resultado de la operacion
 */
 public InitializeCSBPaymentResult onInitializeCSBPayment(GatewayEvent evt){

      // Se recuperan los datos del pago que ha producido el evento
      DatosPago datosPago = evt.datosPago;
      // Se crea una nueva peticion de pago en caso de que esta pago haya sufrido alguna modificacion.
      PeticionPago peticionPagoModificada = new PeticionPago();
      // introducir los cambios.....

      InitializeCSBPaymentResult initializeCSBPaymentResult = new InitializeCSBPaymentResult();
      initializeCSBPaymentResult.resultadoOK = true;
      initializeCSBPaymentResult.modificado = true;
      initializeCSBPaymentResult.peticionPago = peticionPagoModificada;
      // Se pueden incluir mensajes que serviran para informar de la modificacion a
      // quien esta realizando el pago.
      Mensaje mensaje = new Mensaje();
      mensaje.id = "msg1";
      // Mensaje en castellano
      mensaje.texto.put("es", "El importe ha sido modificado.");
      // Mensaje en euskera
      mensaje.texto.put("eu", "El importe ha sido modificado._eu");
      initializeCSBPaymentResult.mensajes.put(mensaje.id, mensaje);

      return initializeCSBPaymentResult;
 }

 /**
  * Recepción de un pago en el Gestor de Pagos desde una Aplicación Departamental
  * @param evt datos del evento
  */
 public void onInitializePayment(GatewayEvent evt){
      System.out.println("La peticion de pago " + evt.id + " ha sido registrada en la Pasarela de
Pagos...");
 }

 /**
  * Un ciudadano accede al interfaz de usuario de la Pasarela de Pagos
  * @param evt datos del evento
  */
 public void onBeginPayment(GatewayEvent evt){
      System.out.println("Se ha solicitado la tramitacion del pago " + evt.id + " a través de la Pasarela
de Pagos...");
 }

 /**
  * Se cancela un pago que se había registrado en la pasarela y que aún no se
  * había comenzado a tramitar on-line / off-line
  * @param evt datos del evento
  */
 public void onCancelPayment(GatewayEvent evt){
      System.out.println("Se ha cancelado el pago con identificador: " + evt.id);
 }

 /**
  * El ciudadano se imprime la liquidación de un pago para poder pagarlo off-line
  * en una ventanilla / cajero de una Entidad Financiera
  * @param evt datos del evento
  */
 public void onPayOFFLine(GatewayEvent evt){
      System.out.println("El pago con identificador: " + evt.id + " se va a pagos off-line...");
 }

 /**
  * El ciudadano decide paga on-line en la banca electrónica de una entidad financiera
  * @param paymentOid Identificador de pago que se paga on-line
  */
 public void onPayONLine(GatewayEvent evt){
      System.out.println("El pago con identificador: " + evt.id + " se va a pagos on-line...");
```

```
 }

 /**
  * El ciudadano entra en la banca electrónica de una entidad financiera y esta envía
  * el pago a la Pasarela de Pagos para su validación resultando esta correcta.
  * @param evt datos del evento
  */
 public void onValidatePaymentOK(GatewayEvent evt){
       System.out.println("El pago con identificador: " + evt.id + " se ha validado de forma
correcta...");
       System.out.println("Peticion de validacion recibida desde la entidad: " + evt.estado.entidad);
 }

 /**
  * El ciudadano entra en la banca electrónica de una entidad financiera y esta envía
  * el pago a la Pasarela de Pagos para su validación resultando esta incorrecta.
  * NOTA: No indica que se haya pagado todavía
  * @param evt datos del evento
  */
 public void onValidatePaymentNOK(GatewayEvent evt){
       System.out.println("El pago con identificador: " + evt.id + " se ha validado de forma
incorrecta...");
       System.out.println("Peticion de validacion recibida desde la entidad: " + evt.estado.entidad);
 }

 /**
  * Un pago se ha realizado en la banca electrónica de una entidad financiera de forma correcta
  * @param paymentStateData los datos de estado del pago
  */
 public void onPayONLineOK(GatewayEvent evt){
       System.out.println("El pago con identificador: " + evt.id + " se ha pagado de forma correcta...");
       System.out.println("Operacion realizada en la entidad: " + evt.estado.entidad);
 }

 /**
  * Un pago se ha realizado en la banca electrónica de una entidad financiera de forma incorrecta
  * @param evt datos del evento
  */
 public void onPayONLineNOK(GatewayEvent evt){
       System.out.println("El pago con identificador: " + evt.id + " NO se ha pagado de forma
correcta...");
       System.out.println("Operacion realizada en la entidad: " + evt.estado.entidad);
 }

}
```

4.4.3 Implementación en base a un Servlet RPC

Un servlet RPC es una implementación de Servlet que "abstrae" al programador de la complejidad de la programación de Servlets ya que simplemente ha de codificar una clase java plana (módulo) con métodos "normales".

En este caso, la clase *RPCServletGatewayEventListenerImpl* envía los eventos que se producen en la pasarela a un servlet RPC de la Aplicación Departamental responsable del tipo de pago correspondiente.

Para implementar un Servlet RPC que escuche por los eventos del Gestor de Pagos únicamente hay que seguir los siguientes pasos:

PASO 1
Registrar el Servlet RPC en el fichero web.xml del WAR
(suponer que el WAR se llama aaaWar)

```
<servlet>
        <servlet-name>RPCServlet</servlet-name>
        <description>Event Listener Gestor Pagos</description>
        <servlet-class>
                com.ejie.r01f.rpcdispatcher.RPCDispatcherServlet
        </servlet-class>
        <init-param>
                <param-name>appCode</param-name>
                <param-value>aaa</param-value>
                <description>Codigo de Aplicacion</description>
        </init-param>
</servlet>

<servlet-mapping>
        <servlet-name>RPCServlet</servlet-name>
        <url-pattern>/aaaGatewayEventListenerServlet</url-pattern>
</servlet-mapping>
```

Con esta configuración se registra un servlet en el war aaaWar accesible mediante la URL:

http://sitio/aaaWar/aaaGatewayEventListenerServlet

PASO 2
Mapear los módulos RPC

El servlet RPC utiliza el fichero de propiedades **aaa.properties.xml** situado en **/config/aaa** (o en cualquier sitio accesible por el classPath) para mapear los módulos que contienen el código.

El fichero aaa.properties.xml debe contener al menos:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<properties>
 <!-- RPC Dispatcher -->
 <rpcDispatcherConfig>
  <usesToken>false</usesToken>
  <reloadPassword>almeja</reloadPassword>
  <modules>
   <module>
    <name>eventListener</name>
    <executor>[paquete].GatewayEventListenerRPCModule</executor>
   </module>
  </modules>
 </rpcDispatcherConfig>
<properties>
```

PASO 3
Implementar la clase con el módulo que contiene los manejadores de eventos
GatewayEventListenerRPCModule

```
package p12d.test.eventlistener;

import java.sql.Date;
import java.text.SimpleDateFormat;

import p12d.exe.pasarelapagos.eventlistener.RPCGatewayEventListener;
import p12f.exe.pasarelapagos.exceptions.AdminException;
import p12f.exe.pasarelapagos.objects.DatosPago;
import p12f.exe.pasarelapagos.objects.GatewayEvent;
import p12f.exe.pasarelapagos.objects.Mensaje;
import p12f.exe.pasarelapagos.objects.PeriodoPago;
import p12f.exe.pasarelapagos.paymentrequest.InitializeCSBPaymentResult;

import com.ejie.r01f.log.R01FLog;
import com.ejie.r01f.rpcdispatcher.RPCContext;
import com.ejie.r01f.rpcdispatcher.RPCException;
import com.ejie.r01f.rpcdispatcher.RPCExec;
import com.ejie.r01f.xml.marshalling.XOMarshallerException;

public class RPCServletGatewayEventListenerImpl extends RPCExec implements
RPCGatewayEventListener{

    public RPCServletGatewayEventListenerImpl() {
        super();
    }

    public RPCServletGatewayEventListenerImpl(RPCContext otherContext) {
        super(otherContext);
    }

    public String service() throws RPCException {
```

```
            return null;
      }

      public void initialize(RPCContext newContext) throws RPCException {}

      /**
       * Método que maneja el evento en que la pasarela actualiza el CSB de
una peticion de pago.
       * @param event Objeto que encapsula la información sobre el evento
que se ha producido.
       */
      public String onInitializeCSBPayment(String evtXML) throws
AdminException{
            try {
                  GatewayEvent evt = GatewayEvent.getObject(evtXML);
                  Date fechaEvento = new Date(evt.timeStamp);
                  SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yy
H:mm:ss");

      R01FLog.to("p12d.eventListener").info("\n\n\n**********************
*********************************");

      R01FLog.to("p12d.eventListener").info("onInitializeCSBPayment.... " +
formatter.format(fechaEvento));

      R01FLog.to("p12d.eventListener").info("**************************
*************************\n\n\n");

                  // Se recuperan los datos del pago que ha producido el
evento
                  DatosPago datosPago = evt.datosPago;
                  // Se comprueban los datos del pago y en caso de que esta
pago haya sufrido
                  // alguna modificacion se introducen los cambios.....
                  PeriodoPago periodo =
(PeriodoPago)evt.datosPago.periodosPago.get(PeriodoPago.PERIODO_NORMAL);
                  periodo.importe = 222;
                  evt.datosPago.periodosPago.put(PeriodoPago.PERIODO_NORMAL,
periodo);

                  InitializeCSBPaymentResult initializeCSBPaymentResult = new
InitializeCSBPaymentResult();
                  initializeCSBPaymentResult.resultadoOK = true;
                  initializeCSBPaymentResult.modificado = true;
                  initializeCSBPaymentResult.datosPago = datosPago;
                  // Se pueden incluir mensajes que serviran para informar de
la modificacion a
                  // quien esta realizando el pago.
                  Mensaje mensaje = new Mensaje();
                  mensaje.id = "msg1";
                  // Mensaje en castellano
                  mensaje.texto.put("es", "El importe ha sido modificado.");
                  // Mensaje en euskera
                  mensaje.texto.put("eu", "El importe ha sido
modificado._eu");
                  initializeCSBPaymentResult.mensajes.put(mensaje.id,
mensaje);

                  return initializeCSBPaymentResult.toXML();
            } catch (XOMarshallerException e) {
                  e.printStackTrace();
                  throw new AdminException(e);
            }
      }

      /**
       * Recepción de un pago en el Gestor de Pagos desde una Aplicación
Departamental
       * @param evt datos del evento
       */
      public void onInitializePayment(String evtXML) throws AdminException{
            try {
                  GatewayEvent evt = GatewayEvent.getObject(evtXML);
                  Date fechaEvento = new Date(evt.timeStamp);
                  SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yy
H:mm:ss");

      R01FLog.to("p12d.eventListener").info("\n\n\n**********************
*********************************");

      R01FLog.to("p12d.eventListener").info("onInitializePayment.... " +
formatter.format(fechaEvento));

      R01FLog.to("p12d.eventListener").info("**************************
*************************\n\n\n");

            } catch (XOMarshallerException e) {
```

```
                e.printStackTrace();
                throw new AdminException(e);
            }
        }

        /**
         * Un ciudadano accede al interfaz de usuario de la Pasarela de Pagos
         * @param evt datos del evento
         */
        public void onBeginPayment(String evtXML) throws AdminException{
            try {
                GatewayEvent evt = GatewayEvent.getObject(evtXML);
                Date fechaEvento = new Date(evt.timeStamp);
                SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yy
H:mm:ss");

        R01FLog.to("p12d.eventListener").info("\n\n\n**********************
**********************************");
                R01FLog.to("p12d.eventListener").info("onBeginPayment.... "
+ formatter.format(fechaEvento));

        R01FLog.to("p12d.eventListener").info("****************************
*************************\n\n\n");

            } catch (XOMarshallerException e) {
                e.printStackTrace();
                throw new AdminException(e);
            }
        }

        /**
         * Se cancela un pago que se había registrado en la pasarela y que
aún no se
         * había comenzado a tramitar on-line / off-line
         * @param evt datos del evento
         */
        public void onCancelPayment(String evtXML) throws AdminException{
            try {
                GatewayEvent evt = GatewayEvent.getObject(evtXML);
                Date fechaEvento = new Date(evt.timeStamp);
                SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yy
H:mm:ss");

        R01FLog.to("p12d.eventListener").info("\n\n\n**********************
**********************************");
                R01FLog.to("p12d.eventListener").info("onCancelPayment....
" + formatter.format(fechaEvento));

        R01FLog.to("p12d.eventListener").info("****************************
*************************\n\n\n");

            } catch (XOMarshallerException e) {
                e.printStackTrace();
                throw new AdminException(e);
            }
        }

        /**
         * El ciudadano se imprime la liquidación de un pago para poder
pagarlo off-line
         * en una ventanilla / cajero de una Entidad Financiera
         * @param evt datos del evento
         */
        public void onPayOFFLine(String evtXML) throws AdminException{
            try {
                GatewayEvent evt = GatewayEvent.getObject(evtXML);
                Date fechaEvento = new Date(evt.timeStamp);
                SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yy
H:mm:ss");

        R01FLog.to("p12d.eventListener").info("\n\n\n**********************
**********************************");
                R01FLog.to("p12d.eventListener").info("onPayOFFLine.... " +
formatter.format(fechaEvento));

        R01FLog.to("p12d.eventListener").info("****************************
*************************\n\n\n");

            } catch (XOMarshallerException e) {
                e.printStackTrace();
                throw new AdminException(e);
            }
        }

        /**
         * El ciudadano decide paga on-line en la banca electrónica de una
entidad financiera
```

```
     * @param paymentOid Identificador de pago que se paga on-line
     */
    public void onPayONLine(String evtXML) throws AdminException{
        try {
            GatewayEvent evt = GatewayEvent.getObject(evtXML);
            Date fechaEvento = new Date(evt.timeStamp);
            SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yy
H:mm:ss");

    R01FLog.to("p12d.eventListener").info("\n\n\n**********************
**********************************");
            R01FLog.to("p12d.eventListener").info("onPayONLine.... " +
formatter.format(fechaEvento));

    R01FLog.to("p12d.eventListener").info("****************************
*************************\n\n\n");

        } catch (XOMarshallerException e) {
            e.printStackTrace();
            throw new AdminException(e);
        }
    }

    /**
     * El ciudadano entra en la banca electrónica de una entidad
financiera y esta envía
     * el pago a la Pasarela de Pagos para su validación resultando esta
correcta.
     * @param evt datos del evento
     */
    public void onValidatePaymentOK(String evtXML) throws AdminException{
        try {
            GatewayEvent evt = GatewayEvent.getObject(evtXML);
            Date fechaEvento = new Date(evt.timeStamp);
            SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yy
H:mm:ss");

    R01FLog.to("p12d.eventListener").info("\n\n\n**********************
**********************************");

    R01FLog.to("p12d.eventListener").info("onValidatePaymentOK.... " +
formatter.format(fechaEvento));

    R01FLog.to("p12d.eventListener").info("****************************
*************************\n\n\n");

        } catch (XOMarshallerException e) {
            e.printStackTrace();
            throw new AdminException(e);
        }
    }

    /**
     * El ciudadano entra en la banca electrónica de una entidad
financiera y esta envía
     * el pago a la Pasarela de Pagos para su validación resultando esta
incorrecta.
     * NOTA: No indica que se haya pagado todavía
     * @param evt datos del evento
     */
    public void onValidatePaymentNOK(String evtXML) throws
AdminException{
        try {
            GatewayEvent evt = GatewayEvent.getObject(evtXML);
            Date fechaEvento = new Date(evt.timeStamp);
            SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yy
H:mm:ss");

    R01FLog.to("p12d.eventListener").info("\n\n\n**********************
**********************************");

    R01FLog.to("p12d.eventListener").info("onValidatePaymentNOK.... " +
formatter.format(fechaEvento));

    R01FLog.to("p12d.eventListener").info("****************************
*************************\n\n\n");

        } catch (XOMarshallerException e) {
            e.printStackTrace();
            throw new AdminException(e);
        }
    }

    /**
     * Un pago se ha realizado en la banca electrónica de una entidad
financiera de forma correcta
     * @param paymentStateData los datos de estado del pago
```

```
     */
    public void onPayONLineOK(String evtXML) throws AdminException{
         try {
              GatewayEvent evt = GatewayEvent.getObject(evtXML);
              Date fechaEvento = new Date(evt.timeStamp);
              SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yy
H:mm:ss");

    R01FLog.to("p12d.eventListener").info("\n\n\n**********************
*********************************");
              R01FLog.to("p12d.eventListener").info("onPayONLineOK.... "
+ formatter.format(fechaEvento));

    R01FLog.to("p12d.eventListener").info("****************************
**************************\n\n\n");

         } catch (XOMarshallerException e) {
              e.printStackTrace();
              throw new AdminException(e);
         }
    }

    /**
     * Un pago se ha realizado en la banca electrónica de una entidad
financiera de forma incorrecta
     * @param evt datos del evento
     */
    public void onPayONLineNOK(String evtXML) throws AdminException{
         try {
              GatewayEvent evt = GatewayEvent.getObject(evtXML);
              Date fechaEvento = new Date(evt.timeStamp);
              SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yy
H:mm:ss");

    R01FLog.to("p12d.eventListener").info("\n\n\n**********************
*********************************");
              R01FLog.to("p12d.eventListener").info("onPayONLineNOK.... "
+ formatter.format(fechaEvento));

    R01FLog.to("p12d.eventListener").info("****************************
**************************\n\n\n");

         } catch (XOMarshallerException e) {
              e.printStackTrace();
              throw new AdminException(e);
         }
    }

}
```

5 Estructuras XML

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

Las estructuras de datos en formato XML son las siguientes:

paymentRequestData *XML de Petición de Pago*	XML enviado por la Aplicación Departamental y que contiene los datos de una o varias peticiones de pago dirigidas a la Pasarela de Pagos de la Administración
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 Pasarela de Pagos 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.
initializeCSBPaymentResult *XML de Resultado de la preinicialización de un pago*	XML de resultado de la preinicialización de un pago, operación que permite la actualización de los datos de un pago introducido por un usuario.

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

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 *toXML()***) y obtener el XML que representa la información de uno de estos objetos (método ***getObject(String XML)***).

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

5.1 PaymentRequestData.

paymentRequestData	Objeto ***PaymentRequestData***. Esta estructura encapsula una o varias Peticiones de Pago para la Pasarela de Pagos.
peticionesPago	Mapa que contiene una o varias Peticiones de Pago.
peticionPago	Objeto ***PeticionPago***. Esta estructura contiene todos los datos relacionados con una Petición de Pago.
Id	Identificador del Pago (***String***).
datosPago	Objeto ***DatosPago***. Esta estructura contiene los datos del Pago.
formato	Formato del Pago (***String***).
validar	Indica el tipo de validación que hay que aplicar a los digitos de control(***int***): ▪ **NO_VALIDAR**(0): Los digitos de control no se validan. ▪ **VALIDACION_NORMAL**(1): Se apica la validación normal a los dígitos de control. ▪ **VALIDACION_TRAFICO**(2): Se aplica una validación especial a los dígitos de control para las validaciones de tráfico.
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***).
Descripción	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***).
backendDataMap	Mapa con los atributos para el sistema de back-end.
backendData	Objeto ***BackendData.*** Esta estructura contiene un dato del sistema de back-end.
id	Identificador del dato.
value	Valor del dato.
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***).
backend	Objeto ***Backend.*** Esta estructura contiene los datos del sistema de back-end.
systemID	Identificador del sistema de back-end.
enabled	Indica si el sistema de back-end está activo o no.
backendDataMap	Mapa con los atributos para el sistema de back-end.
backendData	Objeto ***BackendData.*** Esta estructura contiene un dato del sistema de back-end.
id	Identificador del dato.
value	Valor del dato.

5.2 PresentationRequestData.

presentationRequestData	Objeto ***PresentationRequestData***. Esta estructura contiene los datos de presentación para la Pasarela de Pagos.
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
liquidaciones	Mapa con las liquidaciones de los pagos.
liquidacion	Objeto ***Liquidacion.*** Esta estructura contiene la Liquidación de un Pago.
id	Identificador de la Liquidación.
plantillasLiquidacion	Mapa con las plantillas para obtener los PDFs de las liquidaciones.
plantilla	Objeto ***Plantilla.*** Esta estructura contiene la Plantilla de una Liquidación de Pago.
oid	Identificador de la Plantilla.
urls	Mapa con las URLs de la plantilla.
url	Objeto ***Url.*** Esta estructura contiene la URL de una Plantilla de una Liquidación de Pago.
id	Identificador de la URL.
url	URL.
imagenes	Mapa que contiene una o varias Imágenes.
imagen	Objeto ***Imagen.*** Esta estructura contiene los datos de una Imagen.
id	Identificador de la imagen.
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.
bin	Imagen en formato binario codificada en base 64.
idCartaPago	Identificador de la Carta de Pago.
textoPie	Texto del pie de la Liquidación 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
lugarFirma	Texto del lugar de firma de la Liquidación 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
numeroLiquidacion	Número de la Liquidación.

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 el tipo de validación que hay que aplicar a los digitos de control(***int***): ▪ **NO_VALIDAR**(0): Los digitos de control no se validan. ▪ **VALIDACION_NORMAL**(1): Se apica la validación normal a los dígitos de control. ▪ **VALIDACION_TRAFICO**(2): Se aplica una validación especial a los dígitos de control para las validaciones de tráfico.
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 *InitializeCSBPaymentResult.*

initializeCSBPaymentResult	Objeto ***InitializeCSBPaymentResult.*** Estructura que contiene el resultado de la preinicialización de un pago, operación que permite la actualización de los datos de un pago introducido por un usuario.
resultadoOK	Indica si la operación se ha realizado con éxito o ha habido algún error durante la preinicialización (***boolean***).
modificado	Indica si la en la preinicialización se han modificado los datos del pago (***boolean***).
datosPago	Objeto ***DatosPago***. Esta estructura contiene los datos del Pago.
formato	Formato del Pago (***String***).
validar	Indica el tipo de validación que hay que aplicar a los digitos de control(***int***): ▪ **NO_VALIDAR**(0): Los digitos de control no se validan. ▪ **VALIDACION_NORMAL**(1): Se apica la validación normal a los dígitos de control. ▪ **VALIDACION_TRAFICO**(2): Se aplica una validación especial a los dígitos de control para las validaciones de tráfico.
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***).
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 PaymentRequestData: Datos de Petición de Pago

```
<paymentRequestData>
    <peticionesPago>
        <peticionPago id='90507048330015031840175196771123457012345678 20'>
            <aplicacion codigo='z99'/>
            <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><![CDATA[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>1840175196771</referencia>
                <formato>507</formato>
                <emisor>04833001-503</emisor>
                <periodosPago>
                    <periodoPago id='Periodo de Pago Actual'>
                        <fechaFin>010506</fechaFin>
                        <identificacion>123457</identificacion>
                        <importe>123456782</importe>
                        <fechaInicio>010605</fechaInicio>
                        <activo>true</activo>
                    </periodoPago>
                </periodosPago>
                <validar>1</validar>
                <tipo>503</tipo>
                <codigo>90507048330015031840175196771123457012345678 20</codigo>
            </datosPago>
            <descripcion>
                <es>Aplicacion de Prueba  CSB 507 _es</es>
                <eu>Aplicacion de Prueba  CSB 507 _eu</eu>
            </descripcion>
            <emisor>
                <cif>04833001</cif>
                <nombre>
                    <es>Administración 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>
            <conceptos>
```

```
                <conceptoPeticion>
                    <numeroLinea>1</numeroLinea>
                    <descripcion>
                        <es>Matriculas de seminarios, congresos, jornadas y
cursos</es>
                        <eu>Matriculas de seminarios, congresos, jornadas y
cursos_eu</eu>
                    </descripcion>
                    <importe>145</importe>
                </conceptoPeticion>
            </conceptos>
        </peticionPago>
    </peticionesPago>
</paymentRequestData>
```

6.2 PresentationRequestData: Datos de Presentacion

```
<presentationRequestData>
    <idioma>es</idioma>
</presentationRequestData>
```

6.3 ProtocolData: Datos de Protocolo

```
<protocolData>
    <token>h45456kj34456g4ywfs879fsfg</token>
    <responseURL></responseURL>
    <sourceSessionId> /-1062722781/6/7022/7022/7023/7023/7022/-
1|1115358971045</sourceSessionId>
    <destinationSessionId>QGFfh43nsf67df456Gc/-4345343434/4/</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=048330015071001120051186160505 1822
        </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='90507048330015031840175196771123457012345678 20'>
            <datosPago>
                <cpr>9050794</cpr>
                <referencia>1840175196771</referencia>
                <formato>507</formato>
                <emisor>04833001-503</emisor>
                <periodosPago>
                    <periodoPago id='Periodo de Pago Actual'>
                        <fechaFin>010506</fechaFin>
                        <identificacion>123457</identificacion>
                        <importe>123456782</importe>
                        <fechaInicio>010605</fechaInicio>
                        <activo>true</activo>
                    </periodoPago>
                </periodosPago>
                <validar>1</validar>
                <tipo>503</tipo>
                <codigo>90507048330015031840175196771123457012345678 20</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>
            <paymentStateData id='90507048330015031840175196771123457012345678 20'>
                <datosPago>
                    <cpr>9050794</cpr>
                    <referencia>1840175196771</referencia>
                    <formato>507</formato>
                    <emisor>04833001-503</emisor>
                    <periodosPago>
                        <periodoPago id='Periodo de Pago Actual'>
                            <fechaFin>010506</fechaFin>
                            <identificacion>123457</identificacion>
                            <importe>123456782</importe>
                            <fechaInicio>010605</fechaInicio>
                            <activo>true</activo>
                        </periodoPago>
                    </periodosPago>
                    <validar>1</validar>
                    <tipo>503</tipo>
                    <codigo>90507048330015031840175196771123457012345678 20</codigo>
                </datosPago>
                <estado>
                    <horaPago>100718</horaPago>
                    <fechaPago>051005</fechaPago>
                    <codigo>04</codigo>
                    <numeroOperacion>3475694534657</numeroOperacion>
                    <nrc>111111111111165618815</nrc>
```

```
                </estado>
            </paymentStateData>
        </returnCode>
    </resultado>
    <operationData>
        <function>getPaymentStateData</function>
        <parameters>
            <parameter>
                <name>paymentId</name>
                <value>90507048330015031840175196771123457012345678 20</value>
            </parameter>
    </operationData>
</operationResult>
```

7 Comunicación con SIPCA a través de la Pasarela de Pagos.

7.1 Información de SIPCA recogida por la Pasarela de Pagos.

Las aplicaciones del Gobierno Vasco pueden utilizar la pasarela como intermediario para la emisión de recibos de SIPCA de las órdenes de pago. Para que la Pasarela genere los recibos de SIPCA correctamente, es necesario que las aplicaciones incluyan en las peticiones de pago los siguientes datos:

- A nivel de ***PeticionPago***:
  - ***infomarSIPCA*** (OBLIGATORIO - true/false): Informa si la Pasarela de Pagos debe enviar a SIPCA información referente a este pago.
  - ***exentoPago*** (OBLIGATORIO - true/false): Informa si el tercero esta exento de efectuar el pago.
  - ***ejercicioContable*** (OBLIGATORIO - valor entero, p.ej.: "2007"): Informa del ejercicio contable al que pertenece el pago.
  - ***fechaOperacion*** (OPCIONAL – fecha en formato "ddMMyyyy", p.ej.: "10032007"): Indica la fecha de operación del pago. Si no se indica este valor, se asignaría la fecha actual.
- A nivel de ***Concepto***:
  - ***codigoIngreso*** (OBLIGATORIO - valor entero, p.ej: "007002"): Valor asignado por la OCE (Oficina de Control Económico).
  - ***aplicPresupuestaria*** (OBLIGATORIO - valor entero, p.ej: "20060248010331332159311300000G"): Valor asignado por la OCE (Oficina de Control Económico).
  - ***numInmovilizado*** (OPCIONAL - valor entero, p.ej:): Valor asignado por la OCE (Oficina de Control Económico).
  - ***baseImponible*** (OPCIONAL - valor decimal, p.ej: "40.35"): En caso de que se trate de un concepto con IVA indica el importe al que se debe aplicar.
  - ***tipoIVA*** (OPCIONAL - valor decimal, p.ej: "16.0"): En caso de que se trate de un concepto con IVA indica el tipo que se debe aplicar.
  - ***ivaRepercutido*** (OPCIONAL - true/false): Indica si el IVA del concepto es repercutido o no.

- **importeIVA** (OPCIONAL - valor decimal, p.ej: "6.46"): En caso de que se trate de un concepto con IVA indica el importe del IVA resultante.

- **conceptoIVA** (OPCIONAL - valor entero, p.ej: "002005"): En caso de que se trate de un concepto que refleja el IVA de otro, se refiere a ese otro concepto.

- **territorioIVA** (OPCIONAL - valor entero, p.ej: "1"): En caso de que se trate de un concepto con IVA indica el territorio que aplica el IVA.

7.2 Ejemplos.

A continuación se muestran 2 ejemplos de los datos de SIPCA que debería enviar una aplicación para los casos de pagos sin IVA y pagos con IVA repercutido.

En todos los casos se separa por una parte la información de SIPCA a nivel de **PeticionPago** y la información a nivel de **Concepto**.

7.2.1 Pago sin IVA.

Los pagos sin IVA están formados por un único concepto.

```
<peticionPago id='90507048330015130206001088659231206000000135000'>
    …
    <backend>
        <backendDataMap>
            <BackendData id='exentoPago'>
                <value>false</value>
            </BackendData>
            <BackendData id='ejercicioContable'>
                <value>2006</value>
            </BackendData>
            <BackendData id='informarSIPCA'>
                <value>true</value>
            </BackendData>
            <BackendData id='fechaOperacion'>
                <value>12062007</value>
            </BackendData>
        </backendDataMap>
        <enabled>true</enabled>
        <systemID>SIPCA</systemID>
    </backend>

    <conceptos>

        <conceptoPeticion>
            <numeroLinea>1</numeroLinea>
            <backendDataMap>
                <BackendData id='aplicPresupuestaria'>
                    <value>2006021013010043215145120000A</value>
                </BackendData>
                <BackendData id='codigoIngreso'>
                    <value>002004</value>
                </BackendData>
            </backendDataMap>
            <descripcion>
                <es>Pago sin IVA</es>
            </descripcion>
```

```
            <importe>13500</importe>
        </conceptoPeticion>

    </conceptos>
    …
</peticionPago>
```

7.2.2 Pago con IVA repercutido.

Las peticiones de pagos con IVA repercutido tienen 2 conceptos, el del propio pago y el del IVA repercutido asociado

```
<peticionPago id='90507048330019000241206199654241206000000046810'>
    …
    <backend>
        <backendDataMap>
            <BackendData id='exentoPago'>
                <value>false</value>
            </BackendData>
            <BackendData id='ejercicioContable'>
                <value>2006</value>
            </BackendData>
            <BackendData id='informarSIPCA'>
                <value>true</value>
            </BackendData>
            <BackendData id='fechaOperacion'>
                <value>12062007</value>
            </BackendData>
        </backendDataMap>
        <enabled>true</enabled>
        <systemID>SIPCA</systemID>
    </backend>

    <conceptos>

        <conceptoPeticion>
            <numeroLinea>1</numeroLinea>
            <baseImponible>0</baseImponible>
            <backendDataMap>
                <BackendData id='baseImponible'>
                    <value>0.0</value>
                </BackendData>
                <BackendData id='aplicPresupuestaria'>
                    <value>20060248010331332159311300000G</value>
                </BackendData>
                <BackendData id='codigoIngreso'>
                    <value>002005</value>
                </BackendData>
                <BackendData id='tipoIVA'>
                    <value>0.0</value>
                </BackendData>
                <BackendData id='ivaRepercutido'>
                    <value>false</value>
                </BackendData>
                <BackendData id='importeIVA'>
                    <value>0.0</value>
                </BackendData>
            </backendDataMap>
            <descripcion>
                <es>Concepto con IVA repercutido </es>
            </descripcion>
            <unidades>0</unidades>
            <tieneIVARepercutido>true</tieneIVARepercutido>
            <IVARepercutido>false</IVARepercutido>
            <tipoIVA>0</tipoIVA>
            <importe>4035</importe>
            <importeIVA>0</importeIVA>
        </conceptoPeticion>
```

```
        <conceptoPeticion>
            <numeroLinea>2</numeroLinea>
            <baseImponible>4035</baseImponible>
            <backendDataMap>
                <BackendData id='baseImponible'>
                    <value>40.35</value>
                </BackendData>
                <BackendData id='conceptoIVA'>
                    <value>002005</value>
                </BackendData>
                <BackendData id='territorioIVA'>
                    <value>1</value>
                </BackendData>
                <BackendData id='codigoIngreso'>
                    <value>007002</value>
                </BackendData>
                <BackendData id='tipoIVA'>
                    <value>16.0</value>
                </BackendData>
                <BackendData id='ivaRepercutido'>
                    <value>true</value>
                </BackendData>
                <BackendData id='importeIVA'>
                    <value>6.46</value>
                </BackendData>
            </backendDataMap>
            <descripcion>
                <es>Concepto de IVA repercutido</es>
            </descripcion>
            <unidades>0</unidades>
            <tieneIVARepercutido>false</tieneIVARepercutido>
            <IVARepercutido>true</IVARepercutido>
            <tipoIVA>1600</tipoIVA>
            <importe>646</importe>
            <importeIVA>646</importeIVA>
        </conceptoPeticion>

    </conceptos>
    …
</peticionPago>
```

8 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@ej-gv.es Teléfono: 945018964
DOMA (Dirección de la Oficina para la Modernización de la Administración) Dirección y Coordinación del Proyecto	Juan Luis Ronco Rodrigo e-mail: ronco@ej-gv.es Telefono: 945018567
EJIE (Eusko Jaularitzaren Informatika Elkartea) Dirección Técnica del Proyecto	Alex Lara Garachana e-mail: a-lara@ejie.es Teléfono: 945017473 Móvil: 615771967
Correo de consulta	mipago@ej-gv.es