Pay Club

“BOTON DE PAGOS: PLUG-IN CLIENT”

INDICE

OBJETIVO

En este documento se establecerá como objetivos principales describir:

1. La funcionalidad del **Plug-in Client** y su integración con el **VPOS** (registra datos sensibles del tarjetahabiente y solicita autorización)
2. Descripción detallada de tramas de envío al **VPOS** y recepción desde el **VPOS**.
3. Clases y Algoritmos de encriptación.
4. Administración de llaves, firmas y certificados digitales.
5. Funcionalidades generales del **VPOS.**

ALCANCE

Los componentes que contempla este documento se han orientado al desarrollo del **Plug-in Client** en lenguaje PHP y su integración al **VPOS**.

Los comercios para realizar sus transacciones necesitan integrarse al **VPOS**.

El **VPOS** contempla como eje principal los procesos de solicitud de autenticación y solicitud de autorización, permitiendo al comercio integrarse fácilmente al esquema de pago mediante el uso del **Plug-in Client** descrito detalladamente en este documento.

DEFINICIONES

Antes de continuar debemos definir las entidades participantes en todo el proceso.

DOMINIO	ENTIDAD	DESCRIPCION
Emisor	**TH: Tarjeta habiente**	El **TH** realiza compras en línea, proporcionando el nombre del titular de la cuenta, el número de tarjeta, y la fecha de vencimiento, directamente. En respuesta a la página de la autentificación de la compra, **TH** proporciona la información necesitada para la autentificación, tal como una contraseña
	TH browser	Actúa como conducto para transportar mensajes entre los dominios
	Emisor	Institución financiera que: • Entra en una relación contractual con el TH para emisión de una o más tarjetas. • Determina la elegibilidad del TH para participar en el Programa **PayClub** • Define los rangos de números de tarjeta elegibles participar en el programa **PayClub** • Proporciona datos acerca de esos rangos de números de tarjeta al DS. • Realiza la inscripción del TH por cada cuenta de la tarjeta.
Adquirente	**Comercio**	Software que maneja la experiencia de compra. Y envía datos de la compra al software **VPOS** utilizando el **Plug-in Client.**
	Plug-in Client	Software de seguridad que permite comunicarse entre el Software del Comercio y el Software **VPOS**. Encripta los datos del pago y los envía al **VPOS**. Recibe las respuestas del **VPOS** y las envía al software del **Comercio.**
	V-POS	Software que su función principal es concentrar las transacciones de compra por comercio electrónico para validar datos generales de la compra, así como centralizar la obtención y seguridad de los datos sensibles del TH (numero de tarjeta, fecha de expiración y CVV2/CVC2), iniciar el proceso de pago – Proceso de Autenticación y Proceso de Autorización - para que finalmente se le entregue la respuesta de la transacción al **Comercio** que lo solicitó vía **Plug-in Client**
	Adquirente	Es una institución financiera que: • Entra en una relación contractual con un Comercio con el propósito de aceptar tarjetas. • Determina la elegibilidad del comercio para participar en el programa **PayClub** • Seguido a la autenticación del pago, el adquiriente realiza estos roles: • Recibe petición de autorización desde el **VPOS**. • Lo reenvía al sistema de Autorización (CAO) • Proporciona respuestas de autorización al **VPOS**. • Somete la transacción terminada al sistema del establecimiento.

FLUJO DE COMPRA GENERAL

En esta parte se detalla brevemente el ciclo de compra completa y los procesos que intervienen donde interactúan todos los componentes o partes de ellos dependiendo de la parametrización que cada establecimiento haya definido. Más adelante se detalla cada proceso.

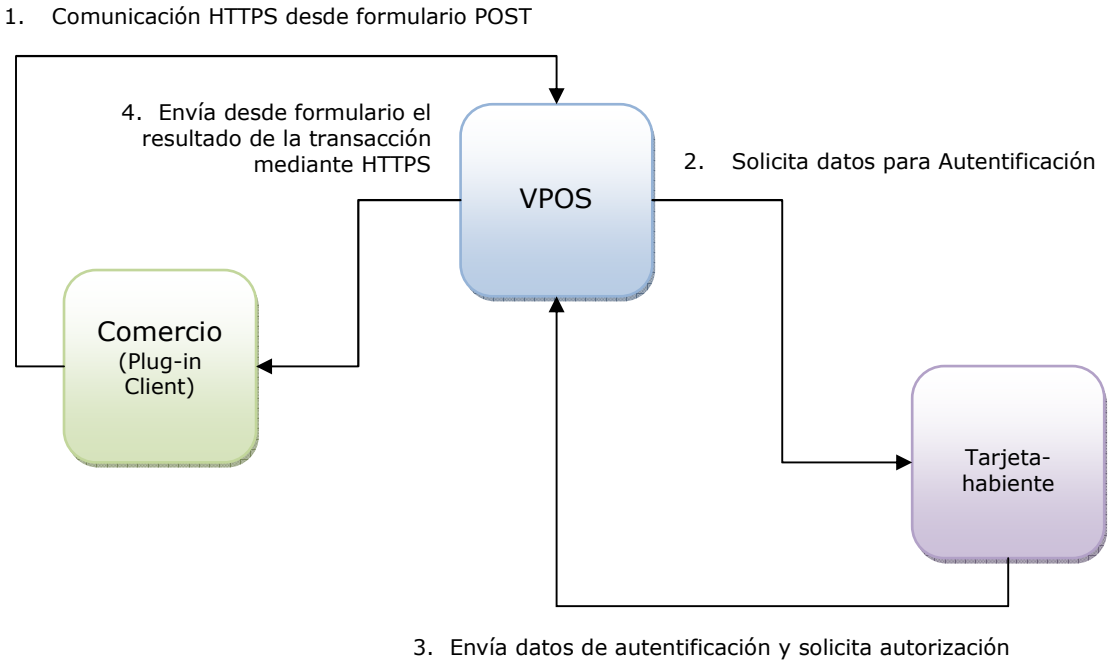
N	Nombre	Descripción
1.	**TH Realiza la compra**	El TH compra en un sitio Web de un comercio y realiza la transacción compra. El Comercio desde un formulario envía al **VPOS** los datos de compras, haciendo uso del **Plug-in Client** proporcionado, exceptuando el número de Tarjeta de Crédito, la fecha de Vencimiento y el CVV2/CVC2. El detalle de los datos se encuentra en la sección FORMATO DE TRAMA PARA SOLICITUD DE PAGO. El **VPOS** muestra en el navegador del TH una página donde se ingresa el número de tarjeta de crédito, la fecha de vencimiento, y el valor CVV2/CVC2.
2.	**VPOS recibe los datos.**	El **VPOS** recibe los datos, verifica los datos de seguridad,
3.	**VPOS – Ejecuta Pre- Proceso**	En caso de que el comercio tenga parametrizado un PROCESO PRELIMINAR, solicita los datos, interactúa con el comercio, para verificación de estos datos.
4.	**VPOS – CNA (Canales Alternativos) verifica enrolamiento.**	Verifica la participación del tarjetahabiente en el servicio. Si esta enrollado, continua con el paso 6. Si no esta enrolado y se permite Activación durante la compra continua al paso 5.
5.	**VPOS – CNA crea la clave segura**	Se muestra una ventana de Creación de Clave segura, para las futuras compras, continuar al paso 8.
6.	**VPOS – CNA solicita autenticación.**	Se Muestra una ventana de autenticación de compra. La ventana muestra información acerca de esa compra en particular. El TH ingresa información de identidad que se verifica en las DB de enrolamiento.
7.	**VPOS – CNA obtiene respuesta de Autenticación.**	Si la autenticación ha fallado o el TH ha cancelado se continua con el paso 11 Si la autenticación es satisfactoria se continua con el paso 8
8.	**VPOS – ejecuta Pre-Proceso**	En caso de tener habilitado un pre-proceso se ejecuta antes de solicitar autorización.
9.	**VPOS – ISO Autorización del Pago**	El **VPOS** procede a solicitar la autorización al componente ISO que formatea la trama y la envía al CAO.
10.	**VPOS ejecuta Pos-Proceso**	En caso de tener habilitado un pos-proceso se ejecuta después de haber solicitado la autorización. (Registro del pago de la transacción por parte del Establecimiento)
11.	**VPOS- devuelve el resultado al Comercio**	**El VPOS** devuelve el resultado de la compra al comercio. El comercio haciendo uso del **Plug-in Client** obtiene los valores de respuesta; quien finalmente le mostrará al TH el resultado de su compra mediante una página dinámica.

PLUG – IN CLIENT.

Software de seguridad que tiene como función principal:

- Comunicarse entre el Software del Comercio y el Software **VPOS** mediante un canal HTTPS seguro.
- Encripta los datos del pago y los envía al **VPOS**, el VPOS realiza el proceso de autentificación y autorización de la compra
- Recibe las respuestas del **VPOS** –y las envía al software del **Comercio** quien muestra la información recibida haciendo uso del **Plug-in Client**.

ARQUITECTURA



El **Plug-in Client** del comercio se comunicará vía Internet, a través de un canal seguro (HTTPS) con el **VPOS** para enviarle los datos de la compra desde un formulario POST; luego éste le presentará al tarjetahabiente una página de ingreso de datos sensibles para luego iniciar los procesos de pago: Autenticación y Autorización.

Finalmente el **VPOS** le enviará desde un formulario, el resultado de la transacción al comercio vía Internet, a través de un canal seguro (HTTPS).

LENGUAJES:

En este documento se contempla el desarrollo del **Plug-in Client** en Lenguaje PHP.

INTEGRACIÓN AL VPOS

El comercio deberá ser registrado en los sistemas de administración de autenticación y autorización por el Adquirente (INTERDIN) antes de iniciar un proceso de compra. Estas parametrizaciones se realizarán por separado según los ambientes, ya sean desarrollo o producción.

Al comercio se le entregará el software **Plug-in Client** de seguridad, el cual será usado durante el proceso de compra, este software es distribuido según el lenguaje de programación del comercio.

Con el fin de proteger y asegurar la información transmitida al realizar una transacción por internet, se debe generar e intercambiar dos pares de llaves asimétricas, para lo cual:

El comercio deberá generar e intercambiar llaves criptográficas asimétricas para la protección e integridad de datos, es decir mediante este par de llaves el plug-in realizará los procesos de cifrado de los datos.

El comercio deberá generar e intercambiar llaves criptográficas asimétricas para la generación y validación de firma, mediante este par de llaves el comercio asegurará su identidad al VPOS.

Cada comercio deberá enviar los ***dos pares de llaves criptográficas asimétricas en formato PEM mediante correo seguro***.

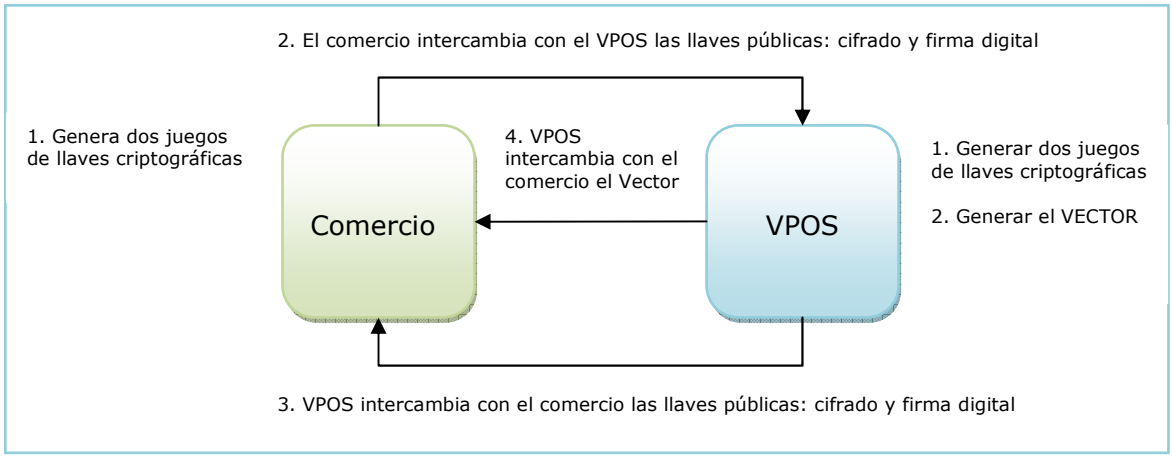
El comercio creará una página dinámica con ***formulario POST*** desde donde enviará la solicitud de pago. Una solicitud de pago es el proceso que inicia el envío de datos de la compra hacia el VPOS.

Integrar a su sitio Web usando el **Plug-in Client** proporcionado por INTERDIN

El comercio creará una página dinámica que recibirá la respuesta a la solicitud de pago, estos parámetros se los detalla en la sección FORMATO DE TRAMA RESPUESTA DE PAGO DEL VPOS. Además el comercio proporcionará al Adquirente la dirección URL de esta página, a la que el **VPOS** devolverá el control.

LLAVES CRIPTOGRÁFICAS PARA INTERCAMBIO DE MENSAJES.

Diagrama de generación e intercambio de llaves



Para que la transmisión de información entre el comercio usando el **Plug-in Client** y el **VPOS** se realice de manera segura el comercio deberá generar:

- ***Par de llaves (llave pública y llave privada) RSA de 1024 bits*** para el cifrado de la información.

- Par de llaves (llave pública y llave privada) RSA de 1024 bits para la generación de la firma digital de la solicitud de pago que enviará el comercio al **VPOS**.

Por otra parte, el **VPOS** generará para cada comercio:

- Par de llaves RSA de 1024 bits para el cifrado y un
- Par de llaves RSA de 1024 bits para la generación de la firma digital de la respuesta de pago que enviará el **VPOS** hacia el comercio.
- Un vector para encriptación 3Des, este vector es de 8 bits codificado en BASE64.

Cabe indicar que por seguridad, estos juegos de llaves, tanto del comercio como del VPOS, deben ser ***independientes al ambiente en el que se apliquen***, es decir las llaves en producción deben ser distintas a las de desarrollo. Así como también los juegos de llaves generados y entregadas por el VPOS serán distintos por cada comercio.

Una vez que se han generado las llaves, tanto el comercio como el **VPOS** deberán intercambiar sus respectivas llaves públicas mutualmente, para desencriptar el mensaje y validación de la firma digital. Este intercambio se lo puede realizar mediante el uso de correo seguro, y el formato de intercambio de las llaves que se deberá utilizar es PEM.

El Vector generado por el **VPOS** debe ser entregado al comercio.

FORMATO DE TRAMA PARA SOLICITUD DE PAGO.

La solicitud de pago se define como el conjunto de datos que son necesarios para que se pueda realizar la solicitud de autenticación y la solicitud de autorización que son generadas por el **VPOS**.

Para la preparación de la información que conforma la solicitud de pago se utilizará el **Plug-in Client** entregado al comercio, de manera que el comercio solo deberá preparar una petición mediante un formulario **POST HTTPS** para comunicarse y enviar la solicitud de pago al **VPOS**. Los campos que formarán parte de la solicitud de pago son los siguientes:

Campo	Descripción
XMLREQUEST	Es el mensaje de solicitud de pago que contiene los datos de la compra. Este mensaje es generado por el **Plug-in Client**, quien se encarga de encriptarlo para su envío al **VPOS**.
XMLDIGITALSIGN	Es la firma digital del mensaje XMLREQUEST, y asegura tanto la autenticidad del emisor y receptor como la integridad de la información. Esta firma digital es generada por el **Plug-in Client**. Codificada en BASE64
XMLACQUIRERID	Es el identificador del adquirente que permite al **VPOS** reconocer a la entidad adquirente del comercio. Este valor es generado por INTERDIN.
XMLMERCHANTID	Es el identificador del comercio o tienda virtual que permite al **VPOS** reconocer al comercio que esta enviando la solicitud de pago.
XMLGENERATEKEY	Es la llave de sesión con la cual se encriptará el mensaje XMLREQUEST. Esta llave es generada por el **Plug-in Client**.

NOTA:

Cada campo son string de caracteres en caso de tener más de un campo vendrá separado por comas.

Cada campo esta codificados en BASE64 y URL Encode.

El campo XMLREQUEST contiene la información del pago. Inicialice los datos como se muestra en cada lenguaje para que el plug-in Client genere la cadena, la encripte y la codifique para que sea enviada en un formulario POS, tal y como se muestra en cada lenguaje.

Tomar en cuentan para una invocación exitosa, la invocación al botón debe ser desde la url técnica registrada en los ambientes de INTERDIN. La URL tiene que ser idéntica, ya que es susceptible a la diferencia de minúsculas y mayúsculas.

Se debe utilizar un único valor de número de orden (referencia) por cada pago exitoso. Si se vuelve a utilizar un número utilizado anteriormente en un pago exitoso, le presentará pantalla de error.

La dirección de invocación del botón de pagos son las siguientes:

URL DE PRUEBAS: https://www3.interdinmpi.com.ec/webmpi/qvpos?
URL DE PRODUCCION: https://www.interdinmpi.com.ec/webmpi/qvpos?

Ejemplo de campo con la información del pago en claro.

1. **XMLREQUEST**=”001;0000000000000001;840;1000;200;000;150;http://mxj702090g.optarecuador.local:8010/BontonPagos/respuesta.aspx;;;Referencia de prueba;;”

   XMLREQUEST=”UzHqxc5j4MPnsDNk7TxSNIqBxl3fctFKdJ%2fWjWCFxUGnmm95b%2bE2pzCcv5ZwWdbmcQ1tIDr0P9mK1nmy9SeWwfFdl5jG46LcsRv6M3ipc9hDwfd1RzJwVc0KQA7gGk%2f9UYU0OWG31DIVyBH%2bgnlDCPPE3B72O%2bD0QeDdvKfth1IylQmBgjwgrosM2ZTFC7cY”

2. **XMLREQUEST**=”002;0000000000000002;840;1000;200;000;150;http://mxj702090g.optarecuador.local:8010/BontonPagos/respuesta.aspx;;;Referencia de prueba;;”
   XMLREQUEST=”lisInZbOWkoWYeJJFX%2b2OFDp5QwZZH4yLs4isCnhTgwDs60gLWE8vyry3c3%2bl6EZDy17%2b9QIqemFxiPzu50twaUYmFqyi9tEgOTvkb7zcxY%2fiAAV2Xqr%2fJQRSuRnN7asnyAgbDwMc5f%2fd1tFkGMJfTYYT%2bb%2bkLSITF0N5FudKWgBx%2b8RM%2f0CjhXp93l2AgdL”

Para el envío de la solicitud de pago se deberá generar una solicitud POST HTTPS, usando un formulario HTML, implementado en el lenguaje de programación del comercio. Este formulario deberá tener la forma siguiente:

```
<form name="frmSolicitudPago" method="post" action="http://as400web:10100/webmpi/VPOS">
        <input type="hidden" name="XMLREQUEST"
        value="AsTOz8S5Lu9j8Q1nnzsXQu1o7POErpOri9FqNWmf08pOXjlmC9SNJKZqfExBWTMQLYaYsRXy3Rzc%0D%0A%2B1Zpk%2BV%2FEl40wkB%2BgRVsPf5Lcgt1sFhH6G5aKRwTlaiX47rz5Rx%2BP4CqcpmMzWnZFYXWP2ug9laa%0D%0AgEQJpgJ72QeSyBQf45rk3Y%2FM%2FUyxvN2e8SvrnAqu3ftlpgnGJK8%3D"
        >
        <input type="hidden" name="XMLDIGITALSIGN" value="
        gU%2Bh5pZT%2BVrl9i7vFkrjf8wjDh1YaFZ0JlVLCe8F2fZfkl0dLn5FKAYvEulAzYTF0v6yrRmFFGDl%0D%0
```

```
AnIWilxB0tdFkvpDMzbFi65WMoQMdL9f22jzN41TXBpWs%2BvsOoKYB7YenonsG%2Fxo1M8xoP%2FkTQ
Nzg%0D%0AA38UQkfCi%2Fk9h1ep8hE%3D ">
<input type="hidden" name="XMLACQUIRERID" value="000000001">
<input type="hidden" name="XMLMERCHANTID" value="0990149054001">
<input type="hidden" name="XMLGENERATEKEY" value="
YX%2BEBqKuDLksGmS%2BPIbaJV1cB3mjWKek3JIJFpt6pbF1XZOnIAlrA7GmxucXlaa10TWfy3zB1Z3x%0D%0AO
V6SkNcc2g%2Fo%2B9QCXoiYd1r8b8a5xEJwywg%2Fc%2BYotup2QXUNYFZ6qybOtTijiC0dVegcvPvGYuXl%0D%0
A2KGvvKqXi8z7DlnRS3A%3D">
</form>
```

ESTRUCTURA DE XMLREQUEST

Campo	Oblg.	Campo Plug-in	Ejemplo	Observaciones
Código de adquirente	X	AdquirerID		Asignado por INTERDIN(No usado)
Código de Local	X	LocalID	001	Asignado por INTERDIN
Código único de comercio		MerchantID		Asignado por INTERDIN(No usado)
Id Transacción	X	TransacctionID	0000000000000001	Identificador único por cada transacción, dado por el comercio. Valor alfa numérico no mayor a 40 caracteres, y excluyendo caracteres especiales(solo letras)
Código de Moneda	X	CurrencyID	840	Según Estándar ISO, por ejemplo en Ecuador el código ISO es 840 que corresponde a Dólares de los Estados Unidos de Norte América.
Valor transacción	X	TransacctionValue	1000	Valor total de la compra, dado por el comercio. el monto debe ir sin decimales, se le agregan tantos ceros a la derecha como numero tenga el exponente, (Si el monto es 100 dólares entonces la cantidad a enviar es 10000). El Valor total de la transacción es el valor neto excluyendo los valores de iva, ice y propina.
Impuesto Principal		TaxValue1	200	Valor numérico que corresponde al impuesto que genera una compra, por ejemplo IVA
Impuesto Secundario		TaxValue2	000	Valor numérico que corresponde al impuesto que puede generar una compra, por ejemplo ICE
Propina		TipValue		Valor numérico que corresponde a la propina efectuada en una compra
Origen	X	SourceDescription	http://mxj702090g.otar	Fuente de donde proviene la transacción (url comercio), su valor debe ser exactamente igual al el mismo que se registró en el sistema de administración del **VPOS** del adquirente. No se aceptan parámetros variables en la URL.
Referencia 1		Reference1		Campos usado para envío de datos adicionales. **30 caracteres.**
Referencia 2		Reference2		Campos usado para envío de datos adicionales. **30 caracteres.**
Referencia 3		Reference3		Campos usado para envío de datos adicionales. **30 caracteres.**
Referencia 4		Reference4		Campos usado para envío de datos adicionales. **30 caracteres.**
Referencia 5		Reference5		Campos usado para envío de datos adicionales. **30 caracteres.**

El valor de TransaccionID, puede ser utilizado en otra invocación, siempre que no exista un pago anterior exitoso con el valor de TransaccionID, pero se recomienda utilizar un número de transacciónID diferente por cada invocación al botón de pagos.

El establecimiento puede hacer uso de las variables de referencia (Opcionales), para enviar valores adicionales que necesite en la página de retorno.
Además de los valores o campos mostrados, se deberá inicializar en el plug-in los valores o campos necesarios para el cifrado del mensaje XMLREQUEST así como también para la generación de la firma digital del mismo. Estos valores se describen a continuación:

Campo	Campo Plug-in	Observaciones
Vector de Inicialización	IV	Valor necesario para el cifrado del mensaje XMLREQUEST. Este valor es generado al momento del registro del comercio al **VPOS**. Este vector esta codificado en Base64
Llave privada Firma Establecimiento	SignPrivateKey	Es la llave privada del comercio con la que se generará la firma digital del mensaje XMLREQUEST. Esta llave privada es generada y almacenada por el comercio.
Llave pública cifrado Interdin	CipherPublicKey	Es la llave pública con la que se cifrará el mensaje XMLREQUEST. Esta llave pública es generada por **INTERDIN** y debe ser almacenada por el comercio.

A continuación se mostrarán ejemplos de uso del plug-in

Lenguaje PHP version 4.4 o superior

```
<?php
include_once("PlugInClient/PlugInClientSend.php");
include_once("PlugInClient/RSAEncryption.php");
$plugin = new PlugInClientSend();
$vector = "Rmu9nHz24/Y=";
$AdquirerID = "PI";
$MerchantID = "0990149054001";
$LocalID = "001";
        $e = $plugin->setLocalID($LocalID);
        if($e!= "")
                echo "Error: $e";

        $e = $plugin->setTransacctionID($_POST["txtTransaccion"]);
        if($e!= "")
                echo "Error: $e";

        $e = $plugin->setTransacctionValue($_POST["txtTotal"] * 100);
        if($e!= "")
                echo "Error: $e";

        $e = $plugin->setTaxValue1($_POST["txtTax1"] * 100);
        if($e!= "")
                echo "Error: $e";

        $e = $plugin->setTaxValue2($_POST["txtTax2"] * 100);
        if($e!= "")
                echo "Error: $e";

        $e = $plugin->setTipValue($_POST["txtPropina"] * 100);
        if($e!= "")
                echo "Error: $e";

        $e = $plugin->setCurrencyID($_POST["txtMoneda"]);
        if($e!= "")
                echo "Error: $e";

        $e = $plugin->setReferencia1($_POST["txtReferencia1"]);
        if($e!= "")
                echo "Error: $e";

        $e = $plugin->setReferencia2($_POST["txtReferencia2"]);
    if($e != "")
                echo "Error: $e";
```

```
        $e = $plugin->setReferencia3($_POST["txtReferencia3"]);
        if($e!= "")
                echo "Error: $e";
        $e = $plugin->setReferencia4($_POST["txtReferencia4"]);
        if($e!= "")
                echo "Error: $e";

        $e = $plugin->setReferencia5($_POST["txtReferencia5"]);
        if($e!= "")
                echo "Error: $e";

        $e = $plugin->setIV($vector);
        if($e!= "")
                echo "Error: $e";

        $plugin->setSignPrivateKey("file://" . realpath("LLAVES/OPTARPriPHPFirma.key"));
        $plugin->setCipherPublicKey("file://" . realpath("LLAVES/OPTARPubJava.key"));
        $xmlGenerateKeyI = $plugin->CreateXMLGENERATEKEY();
        $plugin->XMLProcess("http://10.100.68.108:81/interactive/Comercio/respuesta.php");
        $xmlRequest = $plugin->getXMLREQUEST();
        $xmlFirma = $plugin->getXMLDIGITALSIGN();

?>
```

Lenguaje PHP version 5 o superior

```
<?php
include_once("PlugInClient/PlugInClientSend.php");
include_once("PlugInClient/RSAEncryption.php");
$plugin = new PlugInClientSend();
/*Datos Establecimiento*/
$vector = "mV6VoYVJ54A=";
$AdquirerID = "1711248995001";
$MerchantID = "1711248995001";
$LocalID = "GN01";
$moneda = "840";
$URL_Tecnico = "http://10.100.68.55/Tienda/Invoca.php;";
$ambiente = $_POST["ambiente"];


        $e = $plugin->setLocalID($LocalID);
        if($e!= "")
                echo "Error: $e";
        $e = $plugin->setTransacctionID($_POST["txtTransaccion"]);
        if($e!= "")
                echo "Error: $e";
        $e = $plugin->setTransacctionValue($_POST["Subtotal"] * 100);
        if($e!= "")
                echo "Error: $e";
        $e = $plugin->setTaxValue1($_POST["impuesto1"] * 100);
        if($e!= "")
                echo "Error: $e";
        $e = $plugin->setTaxValue2($_POST["impuesto2"] * 100);
```

```
if($e!= "")
        echo "Error: $e";
$e = $plugin->setTipValue($_POST["propina"] * 100);
if($e!= "")
        echo "Error: $e";
$e = $plugin->setCurrencyID($moneda);
if($e!= "")
        echo "Error: $e";
$e = $plugin->setReferencia1($_POST["txtReferencia1"]);
if($e!= "")
        echo "Error: $e";
$e = $plugin->setReferencia2($_POST["txtReferencia2"]);
 if($e != "")
echo "Error: $e";
$e = $plugin->setReferencia3($_POST["txtReferencia3"]);
if($e!= "")
        echo "Error: $e";
$e = $plugin->setReferencia4($_POST["txtReferencia4"]);
if($e!= "")
        echo "Error: $e";
$e = $plugin->setReferencia5($_POST["txtReferencia5"]);
if($e!= "")
        echo "Error: $e";
$e = $plugin->setIV($vector);
if($e!= "")
                echo "Error: $e";
try
{
        $plugin->setSignPrivateKey("file://" . realpath("llaves/PRIVADA_FIRMA_ESTABLECIMIENTO.pem"));
        $plugin->setCipherPublicKey("file://" . realpath("llaves/PUBLICA_CIFRADO_INTERDIN.pem"));
        $xmlGenerateKeyI = $plugin->CreateXMLGENERATEKEY();
        $plugin->XMLProcess($URL_Tecnico);
        $xmlRequest = $plugin->getXMLREQUEST();
}
catch (Exception $e)
  {
        echo "Error: $e";
  }

?>
```

INTEGRACIÓN DEL PLUG-IN AL COMERCIO.

Al comercio se le entregará un jar, dll o código, dependiendo del lenguaje del comercio, que deberá ser acoplada a su solución Web.

El Comercio debe referenciar al **Plug-in Client** en la carpeta Referencias de la aplicación Web, o ruta de clases

FORMATO DE TRAMA RESPUESTA DE PAGO DEL VPOS.

La respuesta de pago que enviará el **VPOS** al software del comercio estará compuesta por los siguientes campos:

Campo	Descripción
XMLRESPONSE	Es el mensaje de respuesta de pago que contiene los datos del resultado de la autorización.
XMLDIGITALSIGN	Es la firma digital del mensaje XMLRESPONSE, y asegura tanto la autenticidad del emisor y receptor como la integridad de la información. Esta firma digital es generada por el **VPOS**.
XMLACQUIRERID	Identificador del adquirente.
XMLMERCHANTID	Identificador del comercio
XMLGENERATEKEY	Es la llave de sesión con la cual se descifrará el mensaje XMLRESPONSE.

Para recibir la respuesta de pago del **VPOS** es necesario que el comercio implemente una página dinámica en donde se extraigan los valores o campos mostrados en la tabla anterior.

Así mismo el XMLRESPONSE está conformado por 4 campos principales, que son los siguientes:

Campo	Descripción
AUTHORIZATIONSTATE	Este campo contiene el resultado de la autorización. Tiene dos posibles valores: Y = transacción autorizada. N = transacción negada
AUTHORIZATIONCODE	En caso que la transacción haya sido autorizada, este campo contendrá el código de autorización de la transacción.
ERRORCODE	En caso que la transacción haya sido denegada este campo contendrá el código de error respectivo que indicará el motivo del rechazo.
ERRODETAILS	Este campo contendrá la descripción del código de error en caso de producirse un rechazo.

Además de estos campos, la respuesta de pago o XMLRESPONSE, **contendrá todos los campos enviados al VPOS** podrán ser recuperados usando el **Plug-in Client**. Ejemplo. (Campos de Referencia del 1 al 5)
Para descifrar el mensaje XMLRESPONSE, se deberá inicializar en el plug-in los valores o campos criptográficos necesarios, así como también para la verificación de la firma digital del mismo. Estos valores se describen a continuación:

Campo	Campo Plug-in	Observaciones
Vector de Inicialización	IV	Valor necesario para el descifrado del mensaje XMLRESPONSE. Este valor es generado al momento del registro del comercio al V-POS.
Llave pública Firma Interdin	SignPublicKey	Es la llave pública del **VPOS** con la que se verificará la validez de la firma digital del mensaje XMLRESPONSE. Esta llave es generada por el **VPOS** y almacenada por el comercio.
Llave privada cifrado Establecimiento	CipherPrivateKey	Es la llave privada con la que se descifrará el mensaje XMLRESPONSE. Esta llave es generada y almacenada por el comercio.

A continuación se muestra un ejemplo de código para el descifrado de los datos y la verificación de la firma digital:

Lenguaje PHP versión 4.4 o superior

```
<?
        include_once("PlugInClient/PlugInClientRecive.php");
        include_once("PlugInClient/RSAEncryption.php");
        // desencriptado
        $vector = "Rmu9nHz24/Y=";
        $xmlGenerateKey = $_POST["XMLGENERATEKEY"];
```

```
$pluginr = new PlugInClientRecive();
$pluginr->setIV($vector);

$pluginr->setSignPublicKey("file://" . realpath("LLAVES/OPTARPubPHPFirma.key"));
$pluginr->setCipherPrivateKey("file://" . realpath("LLAVES/COMERPriJava.key"));
$error = $pluginr->setXMLGENERATEKEY($xmlGenerateKey);
if($error != "")
{
	echo "Error: $error";
	return;
}

$cadeEnc =  $_POST["XMLRESPONSE"];
$firmaCorrecta = $pluginr->XMLProcess($cadeEnc, $_POST["XMLDIGITALSIGN"]);

if($firmaCorrecta == 0)
{
	echo "<b>Los datos han sido alterados.</b><br>";
	return;
}
else
{
	if($firmaCorrecta != 1)
	{
		echo "<br><br>Error: $firmaCorrecta";
		return;
	}
	else
		echo "<b>Los datos no han sido alterados.</b><br>";
}

?>
```

Lenguaje PHP versión 5 o superior

```
<?
	include_once("PlugInClient/PlugInClientRecive.php");
	include_once("PlugInClient/RSAEncryption.php");
	// desencriptado
	$vector = "mV6VoYVJ54A=";
	$xmlGenerateKey = $_POST["XMLGENERATEKEY"];

	$pluginr = new PlugInClientRecive();
	$pluginr->setIV($vector);

	$pluginr->setSignPublicKey("file://" . realpath("LLAVES/PUBLICA_FIRMA_INTERDIN.pem"));
	$pluginr->setCipherPrivateKey("file://" . realpath("LLAVES/PRIVADA_CIFRADO_ESTABLECIMIENTO.pem"));
	$error = $pluginr->setXMLGENERATEKEY($xmlGenerateKey);
	$msg = "";


	if($error != "")
	{
		$msg = "Error:" . $error;
		return;
	}

	$cadeEnc =  $_POST["XMLRESPONSE"];
	$firmaCorrecta = $pluginr->XMLProcess($cadeEnc, $_POST["XMLDIGITALSIGN"]);
	if($firmaCorrecta == 0)
	{
		//echo "<b>Los datos han sido alterados.</b><br>";
		return;
	}
	else
	{
		if($firmaCorrecta != 1)
		{
			$msg = "<br><br>Error: $firmaCorrecta";
			return;
		}
		else
```

```
                $msg = "<b>Los datos no han sido alterados.</b><br>";
        }
        ?>

<TABLE id="Table16" style="WIDTH: 456px; HEIGHT: 249px" height="249" cellSpacing="1" cellPadding="1"
        width="456" border="0">
        <TR>
        <TD style="WIDTH: 128px" width="128"></TD>
        <TD>Transaccion ID Session Portal <? ?></TD>
        <TD><? print $pluginr->getTransacctionID();?></TD>
        </TR>
        <TR>
        <TD style="WIDTH: 128px" width="128"></TD>
        <TD>Impuesto 1</TD>
        <TD><? print $pluginr->getTaxValue1(); ?></TD>
        </TR>
        <TR>
        <TD style="WIDTH: 128px" width="128"></TD>
        <TD>Impuesto 2</TD>
        <TD><? print $pluginr->getTaxValue2(); ?></TD>
        </TR>
        <TR>
        <TD style="WIDTH: 128px" width="128"></TD>
        <TD>Propina</TD>
        <TD><? print $pluginr->getTipValue(); ?></TD>
        </TR>
        <TR>
        <TD style="WIDTH: 128px" width="128"></TD>
        <TD>Valor</TD>
        <TD><? print $pluginr->getTransacctionValue();?></TD>
        </TR>
        <TR>
        <TD style="WIDTH: 128px" width="128"></TD>
        <TD>ESTADO</TD>
        <TD><?  print $pluginr->getAuthorizationState(); ?></TD>
        </TR>
        <TR>
        <TD style="WIDTH: 128px" width="128"></TD>
        <TD>NUMERO AUTORIZACION</TD>
```

```
<TD><? print $pluginr->getAuthorizationCode(); ?></TD>
</TR>
<TR>
<TD style="WIDTH: 128px" width="128"></TD>
<TD>Referencia 1</TD>
<TD><? print $pluginr->getReferencia1(); ?></TD>
</TR>
<TR>
<TD style="WIDTH: 128px" width="128"></TD>
<TD>Referencia 2</TD>
<TD><? print $pluginr->getReferencia2(); ?></TD>
</TR>
<TR>
<TD style="WIDTH: 128px" width="128"></TD>
<TD>Referencia 3</TD>
<TD><? print $pluginr->getReferencia3(); ?></TD>
</TR>
<TR>
<TD style="WIDTH: 128px" width="128"></TD>
<TD>Referencia 4</TD>
<TD><? print $pluginr->getReferencia4(); ?></TD>
</TR>
<TR>
<TD style="WIDTH: 128px" width="128"></TD>
<TD>Referencia 5</TD>
<TD><? print $pluginr->getReferencia5(); ?></TD>
</TR>
</TABLE>
```

Consideraciones especiales

Para el normal funcionamiento del plug-in se requiere tomar en encuenta las siguientes consideraciones.

Adicional a la librería del plug-in se requiere las siguientes librerías externas:

Lenguaje PHP

Se requiere que se active los siguientes módulos:

1. openssl
2. mcrypt

Pre y Pos Proceso

Mediante estos procesos, el VPOS podrá enviar un conjunto de datos hacia el comercio, donde:

Pre Proceso

Permitirá al VPOS realizar consultas al sitio del establecimiento, justo antes de iniciar el proceso de autorización de la transacción de compra, dependiendo de la respuesta del comercio (ej. ESTADO=OK) se continúa o no con el proceso.

Esta URL la proporciona el establecimiento y es parametrizada en el conjunto de URLS del mismo en el sistema de administración del Botón de Pagos con el tipo de URL “Preproceso”

Post Proceso

Mediante este proceso el VPOS podrá notificar al comercio si el pago se ha realizado exitosamente, y esperar una notificación desde el establecimiento, si la respuesta esperada es exitosa (ej. ESTADO=OK) la transacción termina satisfactoriamente, de lo contrario, es decir si la respuesta no es la esperada o el tiempo de espera a la respuesta del comercio termina, se procede a cancelar la transacción. Este proceso únicamente se inicia si la transacción ha finalizado con éxito, es decir después de la llamada al proceso de autorización y antes de entregar el control del sitio al comercio, quien es el encargado de informar al cliente el estado de la transacción.
En caso contrario si la transacción no es autorizada por la entidad Interdin, esta invocará directamente a la página de retorno con el estado de la autorización de pago en N.

Esta URL es proporcionada por el establecimiento y es parametrizada en el conjunto de URLS del mismo en el sistema de administración del Botón de Pagos con el tipo de URL “Pos proceso”

Esta página no tiene interfaz y tampoco interactúa con el cliente final, es invocada directamente por el botón de pagos, es en éste evento en el cual el establecimiento deberá asentar la transacción en curso del Cliente Final.

Los datos de la trama genérica de Pos Proceso son:

Campo	Descripción	Ejemplo de valor posible
datos	Referencia 2 (dato enviado en el formulario de invocación)	Producto 1
aut	Número de autorización del pago(Generado por el CAO)	558110
Cre	Tipo de Crédito (Configurado por cada establecimiento)	00
mes	Número de meses	1
ttar	Tipo de Tarjeta	DN
sub *	Subtotal	1000
Iva *	Iva	130
Ice *	Ice	84
Int *	Intereses	0
Tot *	Total (incluyendo intereses)	1214
tNo	Referencia del pago (Transaccion ID)	42455
cDt	Referencia 1 (dato enviado en el formulario de invocación)	Juan Perez
Tipo	Parámetro que indica si la invocación es para Pago P, o para reverso R	P o R

*** Los valores numéricos no deberán contener decimales, para ello se le agregan 2 ceros a la derecha, por ejemplo si el valor es 100 dólares entonces la cantidad a enviar es 10000, si el valor es 1.2, entonces se enviará 120.**

Los valores de Referencia 3, Referencia 4, Referencia 5, serán devueltos únicamente en la página de retorno, en la página de post proceso solo se devolverá únicamente el valor de las Referencia1 y Referencia2

Las tramas podrán ser cifradas o no, según la parametrización, en el caso de serlo, se debe determinar las llaves de cifrado. Para este ejemplo se ha generado una llave para el comercio de tipo ***TRIPLEDES "Simetric Key"*** en el uso, 01 lo cual significa que usará también el vector de este mismo uso, tanto la llave como el vector deben ser compartidos al comercio.

Objetivo: Es indispensable, que el establecimiento, en la invocación de la pagina de post proceso, se implemente toda la lógica que necesite realizar el establecimiento, para el asentamiento de la transacción en curso, y si el proceso paso todas las validaciones y funcionamiento correcto, este debería imprimir ESTADO=OK, o el establecimiento verifique el estado del pago por cualquier inconveniente que se presente durante la transacción de compra, está en la facultad de anular la transacción en curso imprimiendo ESTADO=KO.

Ejemplo.
Si el establecimiento es una Aerolínea, y esta al verificar la disponibilidad de un ticket de avión, en la página de post proceso, y este ya no estuviera disponible, estaría en la facultad de anular el pago imprimiendo en la página de post proceso ESTADO=KO;

Reverso en Linea: El botón de pagos, realiza la invocación de la página registrada en la plantilla de post proceso, con el parámetro **tipo=P**, en primera instancia, y si la pagina no responde en el tiempo de espera, o si esta devuelve una cadena ***diferente a ESTADO=OK***, esta es invocada por segunda ocasión cambiando el parámetro **tipo=R**, para indicar que el pago se reverso.

Para descifrar las tramas se tienen las siguientes instrucciones:

```
<?php
include("PlugInClient/TripleDESEncryption.php");
$lsdata = $_POST['xmlReq'];
$lsdata = urldecode($lsdata);
$d3 = new TripleDESEncryption();
$llave = "Z1uSbjhFtboxrmdoha4NRoyJbUq1MgiX";
$iv = "o272gY99fIE=";
$lsdata = $d3->decrypt($lsdata, $llave, $iv);

list($datos, $aut, $Cre,$mes,$ttar,$sub,$Iva,$Ice,$Int,$Tot,$tNo,$cD,$tipo) = split('[&]', $lsdata);
list($p0, $DATOS) = split('[=]', $datos);
list($p1, $AUT) = split('[=]', $aut);
list($p3, $CRE) = split('[=]', $Cre);
list($p4, $MES) = split('[=]', $mes);
list($p5, $TTAR) = split('[=]', $ttar);
list($p6, $SUB) = split('[=]', $sub);
list($p7, $IVA) = split('[=]', $Iva);
list($p8, $ICE) = split('[=]', $Ice);
list($p9, $INT) = split('[=]', $Int);
list($p10, $TOT) = split('[=]', $Tot);
list($p11, $TNO) = split('[=]', $tNo);
list($p12, $CD) = split('[=]', $cD);
list($p13, $TIPO) = split('[=]', $tipo);
?>

<?php if ($TIPO == 'P')
{
        echo 'ESTADO=OK';
}
else
{
        echo 'ESTADO=KO';
```

```
}
?>
```

Ejemplo

URL de invocación a la página de post proceso:

http://www.miempresa.com/proceso1.aspx?
xmlReq=W%2b3srFkyO6wCilCY0Jvf36thQtoACydfvT%2f0C0%2bnLLgBOEm2FZCYpfrhJXXt%2
bxcsAzSwgJA0zD06%0aFOLs6w%2f18r1ptO2ZX3avAEEfrl0%2fn5KhvQ%2b4piyvLtUGU0vXKr
Yz

Trama luego de desencriptar:

datos=ref2&aut=131790&Cre=40&mes=1&ttar=DN&sub=1000&Iva=130&Ice=84&Int=0&Tot=121
4&tNo=5785&cDt= ref1 &tipo=P

Consultas de Estado del Pago.

Objetivo: Permitir al establecimiento verifique el estado del pago por cualquier inconveniente que se presente durante la transacción de compra. Con esto el sitio del establecimiento tendrá la capacidad de mostrar al cliente el estado final de sus pagos (No realizado, Exitoso, Anulado) y se mantenga la consistencia entre pagos exitosos del lado del establecimiento y del lado de PAYCLUB.

Servicio	URL	Usado por
VPOS	https://www3.optar.ec/webmpi/vpos	Establecimiento
Conciliación	https://www3.optar.ec/webmpi/nvpos	Establecimiento
Consulta xml	https://www3.optar.ec/webmpi/qvpos	Establecimiento
Consulta de socios y establecimientos	www.interdin.info.ec	Web Interdin

PAYCLUB 1.0 proporciona una página que permite al sitio web del establecimiento consultar el estado de un pago en particular.

La url de pruebas habilitada para esta consulta es:

URL DE PRUEBAS: https://www3.optar.ec/webmpi/qvpos
URL DE PRODUCCION: https://www.optar.com.ec/webmpi/qvpos

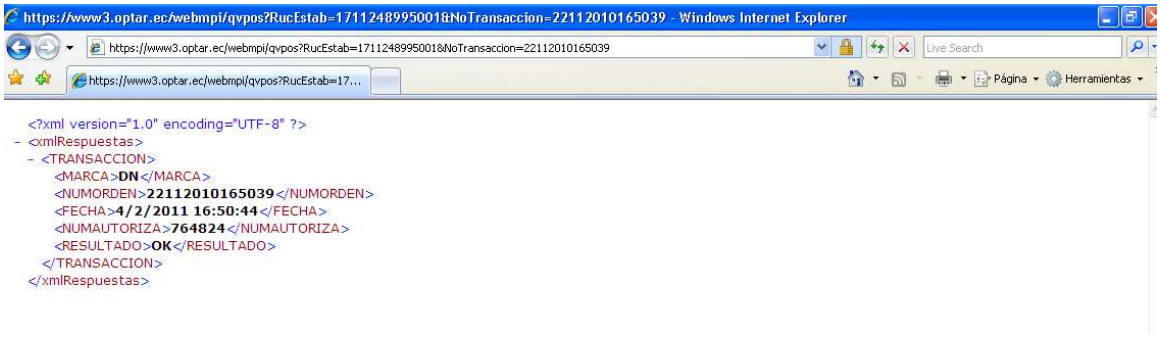
Los parametros que se debe enviar en la URL son:

No.	Descripcion	Nombre	Ejemplos:
1	RUC	RucEstab	1792093872001
2	Numero de Transacción	NoTransaccion(Transaccion ID)	24082009165756

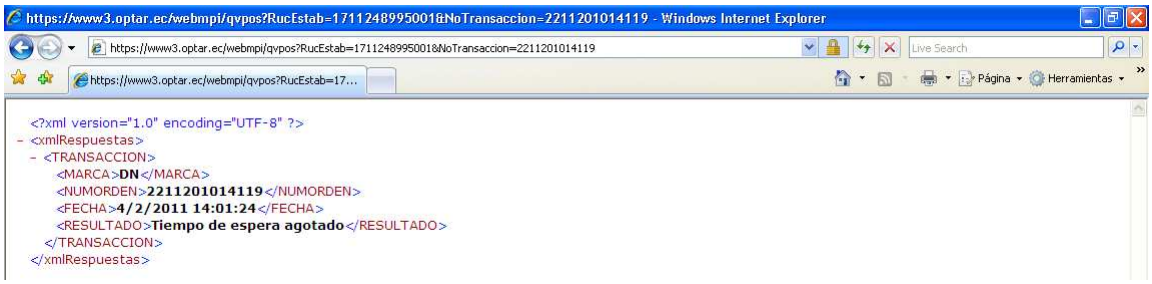
Ejemplos:

https://www3.optar.ec/webmpi/qvpos?RucEstab=1711248995001&NoTransaccion=2
2112010165039

1. Consuta y formato de XML para pago exitoso.



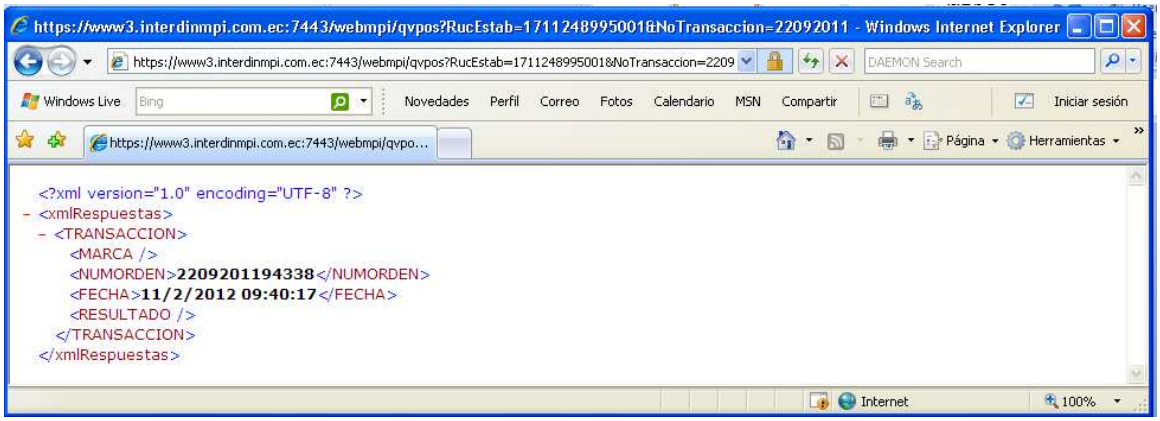
2. Consulta y formato de XML para pago negado.



<NUMORDEN></NUMORDEN> campo que respresenta el TransaccionID, enviado en el formulario que invoca al botón de pagos.

3. Consulta y formato de XML cuando el pago no existe.

4. Consulta y Formato de XML Pago Cancelado.



<RESULTADO/> Indica que Cliente cerro la página y no terminó el pago, o pulsó en botón cancelar la transacción.

Modalidades:

La implementación de la consulta debe realizarse en dos modalidades:

Bajo demanda. La consulta por demanda, debe ser implementada en el historial de pagos realizados por el cliente final. Esta se invocará únicamente en el suseso que el cliente (Tarjeta Habiente) no haya recibido confirmación del estado de su transacción en la página de retorno.

Automática. Esta consulta se implementa con el objetivo de solventar en el caso que el cliente final al no recibir confirmación en la página de retorno y este proceda a terminar y cerrar su ventana de navegación, el establecimiento implementará la consulta automática para que cada determinado tiempo (Ej. 5 minutos) todos los que no hayan sido confirmados en la página de retorno, o en el historial de pagos. Esta consulta deberá a tender a tener a cero pagos no confirmados.

LLAVES

Ejemplo de llave pública en formato PEM

-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCUHmMcFg6+qqoPnePY6t9ym2qZ
kFsGZmE7eKkvw2TTAdDCoqxf1Nfl2gQ30IKYW/tM1BTDYETOkwF8DOsCptc9vxoM
UMZmbDmRXdp01giTks7wzvCqvcIKdIfi7u3PNQamDeB/h4R/6hb8xN4FcyBssQKE
Q2AEe8tO8Gw3qQBYrwIDAQAB
-----END PUBLIC KEY-----

Ejemplo de llave privada en formato PEM

-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQCUHmMcFg6+qqoPnePY6t9ym2qZkFsGZmE7eKkvw2TTAdDCoqxf
1Nfl2gQ30IKYW/tM1BTDYETOkwF8DOsCptc9vxoMUMZmbDmRXdp01giTks7wzvCq
vcIKdIfi7u3PNQamDeB/h4R/6hb8xN4FcyBssQKEQ2AEe8tO8Gw3qQBYrwIDAQAB
AoGAcLnTqetrm1ZdiPccEdlm5106utZvQCaot080S8KEOEa4b3jIgUKGWzj5XtUU
nwnZ7nW1u5/HngOpbJSUQvdg9VJRxBEkpROlpdcyjaKurf2oB6YYurX85arlMpC/
jOA5251ezjkcqvijf6pcXaySK97NfogRBkFZO0e1ipyXT5ECQQDsGl9FHyMxZw4B
4kxak0sWUtR2pn+LMlkp6E54k+fBD7sZjc5Ax1thBgIVf0/ukBrwfpAebj7TVOEf

UOUeuu+5AkEAoJnc+sYNV0DbYSQawvQ5laPDih7VwEEs5s9Tji5uxUwn+EnTTQmB
aKF3/a/wFNe44HrFC2GPHepGNWogiuMvpwJAc2FT63iS/0KIct0/SQgwKqGh2LX3
lHQjZLp1FrHZENz6JzvIbpm+C0Ui7JhB3KipsZt4HJbttuME/QPQZFsgaQJAHMEv
tZ68SMEOBmiGeh1sKgS2QAahUtjYh0LbvPtHTgACsAvXz1VCXE98wmk73R96tIaG
vTDshPirNWkZBoUCxQJBANmYSC/Q1Ty0RR2LpbtRzl5zVziOh9p9yEWRXMiyd3jH
f2rJAXv3dQUjnT9FP/Y9hA+6ZampAhSx0/6cEdjVriM=
-----END RSA PRIVATE KEY-----

Ejemplo de IV Vector codificado en Base64

Rmu9nHz24/Y=

Generación de llaves

Lenguaje PHP version 4.4 o superior

Para generar llaves para usar en versiones de PHP 4.4 o superior, se debe usar el OpenSSL mediante línea de comandos

Ahora se indicarán los pasos para generar cada llave:

1. Primero se genera la llave Privada mendiante el comando:

   openssl genrsa –out e:\\llaves\\PrivadaCifrado.pem 1024

   Esto genera una llave privada de 1024 bits en el archivo **e:\\llaves\\PrivadaCifrado.pem**

2. Ahora de la llave privada hay que extraer la llave pública

   openssl rsa –in e:\\llaves\\PrivadaCifrado.pem –out e:\\llaves\\PublicaCifrado.pem –pubout

   Esto coloca la llave pública en el archivo **e:\\llaves\\PublicaPrivado.pem**

3. Repetir el mismo proceso para el otro par de llaves (Firma)

Lenguaje version PHP 5 >= 5.2.0

```
include_once("RSAEncryption.php");
$filePubK = "e:\\llaves\\PUBLICACIFRADO.pem";
$filePriK = "e:\\llaves\\PRIVADACIFRADO.pem";
$rsaEnc = new RSAEncryption();
$rsaEnc->generateKey($filePubK, $filePriK);


$filePubK = "e:\\llaves\\PUBLICAFIRMA.pem";
$filePriK = "e:\\llaves\\PRIVADAFIRMA.pem";
$rsaEnc = new RSAEncryption();
$rsaEnc->generateKey($filePubK, $filePriK);
```