Todo Pago – JS Hybrid Form API v0.1

Introducción

La API de Todo Pago permite gestionar el pago de diferentes productos utilizando los servicios provistos por la librería Java Script **TPHybridForm.js** de Todo Pago.

Para poder utilizar esta API es necesario incluir en las páginas de pago el script provisto por Todo Pago; además deberá cumplir con ciertos requisitos en los formularios donde se detallen los datos correspondientes al pago.

Una vez incluido el script de la API, se podrá invocar a los métodos necesarios para comunicar su página con Todo Pago.

Detalles

Uso

1) Incluir el script **TPHybridForm.js** provisto por Todo Pago en su página de pago, es requerido incluir el script desde la URL provista por Todo Pago. Ejemplo:

```
<html>
	<head>
		<script src="https://todopago.com.ar/TPHybridForm-v0.1.js"></script>
	</head>
	...
</html>
```

2) Generar el formulario con los campos necesarios para manejar el pago del producto; además se deberá incluir un botón para gestionar el pago con Billetera.
En el apartado de Generación del formulario se detallan los campos requeridos, así como otros detalles importantes en relación al formulario.

3) Inicializar el script de Todo Pago para asociar los campos de su formulario con los servicios de Todo Pago. En el proceso de inicialización deberá especificar todos los parámetros necesarios para poder determinar qué campos del formulario se van a utilizar para gestionar el pago; también podrá especificar ciertas variables para modificar el diseño visual. Ejemplo:

```
window.TPFORMAPI.hybridForm.initForm({
	// En el apartado “Parámetros de inicialización” se detallan los posibles parámetros.
});
```

4) Setear el id público del producto que se está ofreciendo, este id es provisto por Todo Pago. Ejemplo:

```
window.TPFORMAPI.hybridForm.setItem({
	publicKey: '1bad0461f492de8543a1c3.....'
});
```

Generación del formulario

El formulario implementado por el comercio deberá contar, al menos, con los siguientes componentes:

Tipo de elemento	Objetivo	Id por defecto	Campo poblado por
<option>	Mostrar los medios de pago disponibles para el producto elegido.	formaDePagoCbx	API Todo Pago
<option>	Mostrar los bancos disponibles para el medio de pago elegido.	bancoCbx	API Todo Pago
<option>	Mostrar las diferentes promociones para el medio de pago y banco elegidos.	promosCbx	API Todo Pago
<input>	Especificar el número de tarjeta que se desee utilizar.	numeroTarjetaTxt	-
<input>	Mes de vencimiento de la tarjeta.	mesTxt	-
<input>	Año de vencimiento de la tarjeta.	anioTxt	-
<input>	Código de seguridad de la tarjeta.	codigoSeguridadTxt	-
<label>	Mensaje completado por la API en relación al código de seguridad.	labelCodSegTextId	API Todo Pago
<label>	Mensaje completado por la API en relación a la promoción seleccionada	promotionDescTextId	API Todo Pago
<input>	Apellido y nombre del titular de la tarjeta.	apynTxt	-
<option>	Tipo de documento del titular de la tarjeta.	tipoDocCbx	API Todo Pago
<input>	Número de documento del titular de la tarjeta.	nroDocTxt	-
<input>	Email del titular de la tarjeta.	emailTxt	-
<button>	Botón para realizar el pago con los datos del formulario.	btnConfirmarPago	-
<button>	Botón para realizar el pago con Billetera.	btnInicarPagoBilletera	-

Restricciones sobre el formulario:

- Ninguno de los campos del formulario podrá contar con el atributo name.
- Se deberá proveer de manera obligatoria un botón para gestionar el pago con Billetera Todo Pago.
- Todos los elementos de tipo <option> son completados por la API de Todo Pago.
- Los campos tienen un id por defecto. Si se prefiere utilizar otros ids se deberán especificar los mismos cuando se inicialice el script de Todo Pago. En el apartado de “Parámetros de inicialización” se especifican los detalles.

Libertades sobre el formulario:

- Pueden aplicarse todos los detalles visuales que se crean necesarios, la API de Todo Pago no altera los atributos class y style.
- Puede utilizarse la API para setear los atributos placeholder del formulario, para ello deberá especificar dichos placeholders en la inicialización del formulario. En caso de que no se especifiquen los placeholders se usarán los valores por defecto de la API.
  En el apartado de “Parámetros de inicialización” se especifican los detalles.

Parámetros de inicialización de la API

Para inicializar la API se debe instanciar el método **initForm** con los parámetros necesarios para tal fin.

```
window.TPFORMAPI.hybridForm.initForm({ ….. })
```

El objeto utilizado para inicializar la API podrá contener varios parámetros, algunos de ellos son obligatorios; a continuación se detalla la lista de posibles parámetros.

Nombre	Descripción	Requerido	Valor por defecto
optionMedioPagoId	Id del <option> donde se almacenan los medios de pago.	No	formaDePagoCbx
optionBancoPagoId	Id del <option> donde se almacenan los bancos pertenecientes al medio de pago seleccionado.	No	bancoCbx
optionCuotasPagoId	Id del <option> donde se almacenan las promociones pertenecientes al medio de pago y banco seleccionados.	No	promosCbx
inputNumeroTarjetaId	Id del <input> donde el usuario ingresará su número de tarjeta de crédito.	No	numeroTarjetaTxt
inputMesId	Id del <input> donde el usuario ingresará el mes de vencimiento de su tarjeta de crédito.	No	mesTxt
inputAnoId	Id del <input> donde el usuario ingresará el año de vencimiento de su tarjeta de crédito.	No	anioTxt
inputCodSegId	Id del <input> donde el usuario ingresará el código de seguridad de su tarjeta de crédito.	No	codigoSeguridadTxt
labelCodSegTextId	Id del <label> donde se mostrará el mensaje de ayuda para encontrar el código de seguridad de la tarjeta.	No	labelCodSegTextId
labelPromotionTextId	Id del <label> donde se mostrará el mensaje descriptivo de la promoción seleccionada (en caso de haberlo)	No	labelPromotionTextId
inputNombreApellidoId	Id del <input> donde el usuario ingresará su nombre y apellido.	No	apynTxt
optionTipoDocId	Id del <option> donde se almacenarán los tipos de documento posibles.	No	tipoDocCbx
inputNumeroDocId	Id del <input> donde el usuario ingresará su número de documento.	No	nroDocTxt
inputMailId	Id del <input> donde el usuario ingresará su dirección de correo electrónico.	No	emailTxt
botonPagarId	Id del <button> donde el usuario hará click para procesar el pago a través del formulario.	No	btnConfirmarPago
botonPagarConBilleteraId	Id del <button> donde el usuario hará click para abrir la ventana de pago a través de la opción Billetera.	No	btnConfirmarPagoBilletera
placeholderNumeroTarjeta	Texto utilizado como placeholder en el <input> de número de tarjeta.	No	Número de tarjeta
placeholderCodSeg	Texto utilizado como placeholder en el <input> de código de seguridad.	No	Cód. de seguridad
placeholderMes	Texto utilizado como placeholder en el	No	MM

	<input> de mes de vencimiento de la tarjeta.		
placeholderAnio	Texto utilizado como placeholder en el <input> de año de vencimiento de la tarjeta.	No	AA
placeholderNombreApellido	Texto utilizado como placeholder en el <input> de nombre y apellido del usuario.	No	Nombre y Apellido
placeholderNumeroDocumento	Texto utilizado como placeholder en el <input> de número de documento del usuario.	No	Número
placeholderEMail	Texto utilizado como placeholder en el <input> de dirección de correo electrónico del usuario.	No	E-mail
originalPaymentButtonText	Texto del <button> que procesa el pago online a través del formulario.	No	Pagar
originalPaymentCuponButtonText	Texto del <button> que procesa el pago a offline a través del formulario.	No	Generar cupón
modalCssClass	Clase que se aplica al modal en el cual se mostrará el pago con Billetera.	No	
modalContentCssClass	Clase que se aplica al contenido del modal del pago con Billetera.	No	
beforeRequest	Nombre de la función implementada por el comercio que se ejecutará antes de cada request que se haga (útil para mostrar mensajes de “cargando...”)	No	
afterRequest	Nombre de la función implementada por el comercio que se ejecutará después de cada request que se haga.	No	
callbackValidationErrorFunction	Nombre de la función implementada por el comercio que se invocará si se encuentra un error en la validación de los datos ingresados por el usuario.	Sí	
callbackCustomErrorFunction	Nombre de la función implementada por el comercio que se invocará si se produce un error al procesar el pago a través del formulario.	No	
callbackCustomSuccessFunction	Nombre de la función implementada por el comercio que se invocará si se tiene éxito al procesar el pago a través del formulario.	No	
callbackBilleteraFunction	Nombre de la función implementada por el comercio que se invocará luego de que se procese el pago a través de Billetera.	No	

Parámetros de inicialización del Item de pago

Para inicializar un ítem de pago debe invocarse el método **setItem** con los parámetros necesarios para tal fin

```
window.TPFORMAPI.hybridForm.setItem({ ….. })
```

El objeto utilizado para inicializar un ítem de pago la API podrá contener varios parámetros, algunos de ellos son obligatorios; a continuación se detalla la lista de posibles parámetros.

Nombre	Descripción	Requerido	Valor por defecto
publicKey	Id público del producto que se está ofreciendo provisto por Todo Pago.	Sí	-
defaultNombreApellido	String opcional con el nombre y apellido del cliente que hará uso del formulario de pago.	No	-
defaultNumeroDoc	String opcional con el número de documento del cliente que hará uso del formulario de pago.	No	-
defaultMail	String opcional con la dirección de correo electrónico del cliente que hará uso del formulario de pago.	No	-
defaultTipoDoc	String opcional con el tipo de documento del cliente que hará uso del formulario de pago. (DNI, CI, DU, LC, LE, PAS)	No	-

Callbacks

Definición de las funciones callbacks

- **beforeRequest:**
  La función no deberá tener parámetros.

- **afterRequest:**
  La función no deberá tener parámetros.

- **callbackValidationErrorFunction:**
  La función recibe como parámetro un objeto con dos atributos: *field* utilizado para almacenar el id del campo que haya fallado la validación y *error* utilizado para almacenar el mensaje de error. Por ejemplo:

```
{
    field = 'emailTxt';
    error = 'Email inválido';
}
```

- **callbackCustomErrorFunction:**
  En caso de existir un error en el procesamiento del pago a través del formulario implementado en el comercio, la API advertirá el error correspondiente invocando esta función que recibe como parámetro un objeto con dos atributos: *ResultCode* con el código de error devuelto por el servicio de pago y *ResultMessage* con el mensaje correspondiente a dicho código. Por ejemplo:

```
{
    ResultCode = 'XXXX';
    ResultMessage = 'Mensaje de error';
}
```

- **callbackCustomSuccessFunction:**
  En caso de procesarse correctamente el pago a través del formulario implementado en el comercio, la API advertirá el resultado invocando esta función que recibe como parámetro un objeto con dos atributos: *ResultCode* con el código de error devuelto por el servicio de pago y *ResultMessage* con el mensaje correspondiente a dicho código. Por ejemplo:

```
{
    ResultCode = '-1';
    ResultMessage = 'Mensaje de éxito';
}
```

- **callbackBilleteraFunction:**
  Una vez procesado el pago a través de Billetera, la API advertirá el resultado invocando esta función implementada en el comercio, que recibe como parámetro un objeto con dos atributos: *ResultCode* con el código de error devuelto por el servicio de pago y *ResultMessage* con el mensaje correspondiente a dicho código.

```
{
    ResultCode = 'XXXX';
    ResultMessage = 'Mensaje';
}
```

Ejemplo de funciones callbacks

```
function validationCollector(parametros) {
    console.log("My validation collector callback");
    console.log(parametros.field + " -> " + parametros.error);
}

function billeteraPaymentResponse(response) {
    console.log("My billetera payment callback");
    console.log(response.ResultCode + " -> " + response.ResultMessage);
}

function customPaymentSuccessResponse(response) {
    console.log("My custom payment success callback");
    console.log(response.ResultCode + " -> " + response.ResultMessage);
}

function customPaymentErrorResponse(response) {
    console.log("My custom payment error callback");
    console.log(response.ResultCode + " -> " + response.ResultMessage);
}

function initLoading() {
    console.log('Loading...');
}

function stopLoading() {
    console.log('Stop loading...');
}

window.TPFORMAPI.hybridForm.initForm({
        callbackValidationErrorFunction: 'validationCollector',
        callbackBilleteraFunction: 'billeteraPaymentResponse',
        callbackCustomSuccessFunction: 'customPaymentSuccessResponse',
        callbackCustomErrorFunction: 'customPaymentErrorResponse',
        beforeRequest: 'initLoading',
        afterRequest: 'stopLoading'
});
```

Ejemplo de implementación

```
<html>
	<head>
		<title>Sitio de pruebas</title>
		<meta charset="UTF-8">
		<script src="https://todopago.com.ar/TPHybridForm-v0.1.js"></script>
		<style>
		   input, select {
		      display: block;
		   }

		   .modal-class {
		   }

		   .modal-content {
		   }
		</style>
	</head>
	<body>
		<select id="formaDePagoCbx"></select>
		<select id="bancoCbx"></select>
		<select id="promosCbx"></select>
		<label id="labelPromotionTextId"></label>
		<input id="numeroTarjetaTxt"/>
		<input id="mesTxt"/>
		<input id="anioTxt"/>
		<input id="codigoSeguridadTxt"/>
		<label id="labelCodSegTextId"></label>
		<input id="apynTxt"/>
		<select id="tipoDocCbx"></select>
		<input id="nroDocTxt"/>
		<input id="emailTxt"/><br/>
		<button id="MY_btnConfirmarPago"/>
		<button id="MY_btnPagarConBilletera"/>
	</body>
   <script>

	/************* CONFIGURACION DEL API ************************/
	window.TPFORMAPI.hybridForm.initForm({
	   callbackValidationErrorFunction: 'validationCollector',
	   callbackBilleteraFunction: 'billeteraPaymentResponse',
	   callbackCustomSuccessFunction: 'customPaymentSuccessResponse',
	   callbackCustomErrorFunction: 'customPaymentErrorResponse',
	   botonPagarId: 'MY_btnConfirmarPago',
	   botonPagarConBilleteraId: 'MY_btnPagarConBilletera',
	   modalCssClass: 'modal-class',
	   modalContentCssClass: 'modal-content',
	   beforeRequest: 'initLoading',
	   afterRequest: 'stopLoading'
	});

	/************* SETEO UN ITEM PARA COMPRAR ************************/
	window.TPFORMAPI.hybridForm.setItem({
	   publicKey: '1bad0461f492de8543a1.....',
	   defaultNombreApellido: 'Juan Pérez',
	   defaultMail: 'prueba@tpago.com'
	});
```

```
        function validationCollector(parametros) {
            console.log("My validation collector callback");
            console.log(parametros.field + " -> " + parametros.error);
        }

        function billeteraPaymentResponse(response) {
            console.log("My billetera payment callback");
            console.log(response.ResultCode + " -> " + response.ResultMessage);
        }

        function customPaymentSuccessResponse(response) {
            console.log("My custom payment success callback");
            console.log(response.ResultCode + " -> " + response.ResultMessage);
        }

        function customPaymentErrorResponse(response) {
            console.log("My custom payment error callback");
            console.log(response.ResultCode + " -> " + response.ResultMessage);
        }

        function initLoading() {
            console.log('Loading...');
        }

        function stopLoading() {
            console.log('Stop loading...');
        }
    </script>
</html>
```

Usando el formulario default

La API permite construir un formulario default invocando el método *constructDefaultForm* antes de inicializar los parámetros de la API.
Este método recibe como parámetro el Id del objeto HTML que contendrá dicho formulario. Por ejemplo:

```
    <body>
        ....
        <div id="formContainer"/>
        ....
    </body>

    <script>
        window.TPFORMAPI.hybridForm.constructDefaultForm("formContainer");

        window.TPFORMAPI.hybridForm.initForm({
            callbackValidationErrorFunction: 'validationCollector',
            callbackBilleteraFunction: 'billeteraPaymentResponse',
            callbackCustomSuccessFunction: 'customPaymentSuccessResponse',
            callbackCustomErrorFunction: 'customPaymentErrorResponse'
        });

        window.TPFORMAPI.hybridForm.setItem({
            publicKey: '1bad0461f492de8543a1.....',
        });
        ....
```

Requerimientos e implementación

Requerimientos del cliente

Esta API es soportado por los navegadores web:

- CHROME
- FIREFOX
- SAFARI
- IE8+

Requerimientos del comercio

Ninguno.

Implementación en Todo Pago

Para servir correctamente la API desde el host de Todo Pago deben modificarse adecuadamente los siguientes parámetros del archivo **TPHybridForm.js.** Por ejemplo:

```
// API URL
cfg.API_URL = 'http://23.23.144.247/t/1.1/api/';

// FORM EMBEDED URL
cfg.FORM_EMBEDED_URL = 'https://23.23.144.247/formulario2/commands';

// URL FORM CUSTOM PAYMENT
cfg.FORM_PAYMENT_ENDPOINT = 'https://23.23.144.247/t/1.1/FormPaymentRest';
```