

SDKinApp 2.3.0

Manual de integración Android

Versión: 2.3.0
Fecha: 7/11/23

USO RESTRINGIDO



Redsys, Servicios de Procesamiento, S.L. – c/ Francisco Sancha, 12 – 28034 Madrid (España)

www.redsys.es

Autorizaciones y control de versión

AUTOR: Redsys	VALIDADO POR: Redsys	APROBADO POR:
Empresa: Redsys	Empresa: Redsys	Empresa: Redsys
Firma:	Firma:	Firma:
Fecha: 7/11/23	Fecha: 7/11/23	Fecha: 7/11/23

Comentarios: La gestión de la documentación impresa es responsabilidad de la persona que la imprime.

Las versiones impresas de las normas de seguridad no garantizan ser la última versión aprobada. Para consultar la última versión acceder a la base de datos de Alejandría.

Para el tratamiento de la información contenida en este documento se deberá seguir las pautas establecidas en la Normativa Redsys, y en particular en la norma RS.RI.SEG.NOR.0003 NORMA DE CLASIFICACIÓN Y TRATAMIENTO DE LA INFORMACIÓN

Versión	Fecha	Afecta	Breve descripción del cambio
1.0	15/06/2018	Todo	Versión inicial
1.0.1	18/06/2018	Todo	Ejemplos de integración añadidos
1.0.2	21/06/2018	Todo	Corrección de erratas
1.0.3	27/06/2018	Todo	Añadidos nuevos códigos de error
1.0.4	04/10/2018	Todo	Notas versión 2.0.1: <ul style="list-style-type: none"> Alerta configurable al finalizar el Pago con WebView y redirección a URL's OK y KO. Nuevas validaciones de tarjetas para el Pago directo. Color del ProgressBar Configurable
1.0.5	30/10/2018	Todo	Versión 2.0.2
1.0.6	31/10/2018	Todo	Versión 2.0.3: Modificación del nombre de algunos parámetros de la clase TPVVConfiguration y corrección de erratas en la documentación
1.0.7	21/12/2018	Todo	Versión 2.0.4: Se ha subido la API mínima a la 21. Ajuste en operaciones con importe cero.

1.0.8	04/02/2018	Todo	Actualización de los ejemplos de las llamadas de pago directo y pago a través de WebView
1.0.9	26/09/2019	Cambio internos librería	Versión 2.0.5 Se ha actualizado el sistema de redirecciones en los WebView. No necesita cambios en la implementación.
1.0.10	14/01/2020	Añadido tipo de operación	Versión 2.0.6 Se ha añadido un tipo de operación, autenticación. Afecta: <ul style="list-style-type: none"> 3.1.1 Configuración del pago directo 3.2 Pago Webview
1.0.11	22/09/2020	Implementación de parámetros extra en las peticiones	Versión 2.2.0 Se ha añadido la posibilidad de introducir parámetros extra en la petición y respuesta de los pagos. Añadido JSONParser como utilidad para ayuda a la gestión de las peticiones y respuestas en String con formato JSON y HashMap.
1.0.12	22/01/2021	Añadido soporte para un nuevo tipo de tarjetas. Añadido soporte a AndroidX	Versión 2.2.1 <ul style="list-style-type: none"> Para soportar un nuevo tipo de tarjetas se ha añadido la lógica necesaria. No es necesario aplicar cambios en la integración. 1.2 AndroidX
1.1	07/06/2021	Control de errores y correcciones menores. Cambio en diseño y clarificación de ejemplos de código.	Versión 2.2.2 Se controlan los errores generados por pérdida de instancia del callback a causa del ciclo de vida de Android.
1.1.1	08/08/2023	Afecta a las llamadas con certificados	Versión 2.2.3 Mejora en la validación con los certificados en la comunicación hacia Servidor
2.3.0	07/11/2023	Métodos de pago en formato Strings	Versión 2.3.0 Uso de Strings para asignar método de pago

ÍNDICE

1. INTRODUCCIÓN	6
1.1 INTRODUCCIÓN	6
1.2 ANDROIDX	6
1.3 INTEGRACIÓN DE LA LIBRERÍA	6
1.3.1 IMPORTAR MÓDULO EN ANDROID STUDIO	6
1.3.2 PERMISOS EN EL MANIFEST	7
2. CONFIGURACIÓN	8
2.1 PARÁMETROS OBLIGATORIOS	8
2.1.1 LICENCIA DE LA APLICACIÓN	8
2.1.2 ENTORNO	8
2.1.3 FUC DEL COMERCIO	8
2.1.4 TERMINAL	8
2.1.5 CÓDIGO DE LA MONEDA UTILIZADA	8
2.2 PARÁMETROS OPCIONALES	9
3. OPERACIONES DISPONIBLES	10
3.1 PAGO DIRECTO	11
3.1.1 CONFIGURACIÓN DEL PAGO DIRECTO	11
3.1.2 PERSONALIZACIÓN DE LA PANTALLA DE PAGO DIRECTO (OPCIONAL)	11
3.2 PAGO WEBVIEW	12
3.2.1 CONFIGURACIÓN DEL IDIOMA	12
3.2.2 CONFIGURAR REDIRECCIÓN A URLS DE RESULTADO	12
3.2.3 CONFIGURAR ALERTA RESULTADO	12
4. FORMA DE LA RESPUESTA Y CÓDIGOS DE ERROR	13
4.1 CÓDIGOS DE ERROR	13
4.2 RESULTRESPONSE	13
4.3 ERRORRESPONSE	14
5. ANEXO:	15
5.1 LISTA DE IDIOMAS	15
5.2 EJEMPLOS DE IMPLEMENTACIÓN	16
5.2.1 EJEMPLO PAGO DIRECTO NORMAL	16
5.2.2 EJEMPLO PAGO DIRECTO CON SOLICITUD DE REFERENCIA	16
5.2.3 EJEMPLO PAGO DIRECTO CON REFERENCIA	16
5.2.4 EJEMPLO CONFIGURACIÓN PANTALLA PAGO DIRECTO	17
5.2.5 EJEMPLO PAGO WEBVIEW NORMAL	18
5.2.6 EJEMPLO PAGO WEBVIEW CON SOLICITUD DE REFERENCIA	18
5.2.7 EJEMPLO PAGO WEBVIEW CON REFERENCIA	18
5.2.8 PAGOS UTILIZANDO EXTRAParams	19

5.2.9	CONFIGURACIÓN DEL COLOR DEL PROGRESSBAR	19
5.2.10	USO DE JSONPARSER	19

1. Introducción

1.1 Introducción

En el presente documento se define la integración y funcionalidades disponibles del framework TPVV para Android, que permite realizar operativas de comercio electrónico. La librería está disponible para los entornos de integración, producción y test.

Hay disponibles dos tipos de operativa: pago directo y pago a través de WebView.

En el pago directo se muestra una pantalla propia del Framework personalizable que recoge los datos de la tarjeta para realizar el pago. (Este método no tiene autenticación 3DSecure).

Por otro lado, en el pago WebView, la operativa se realiza a través de la web y, además, utiliza autenticación 3DSecure.

1.2 AndroidX

La librería, a partir de la versión 2.2.1, se migra a AndroidX.

AndroidX es el nuevo estándar de Google. La librería android.support ya no recibe soporte por parte de Google en favor de AndroidX, por lo que se ha migrado a AndroidX para tener un mayor soporte y compatibilidad.

<https://developer.android.com/jetpack/androidx>

En caso de que debido a la ofuscación no encuentre algún recurso de la librería, se debe añadir en el fichero gradle.properties del proyecto la siguiente línea.

```
android.enableR8=false
```

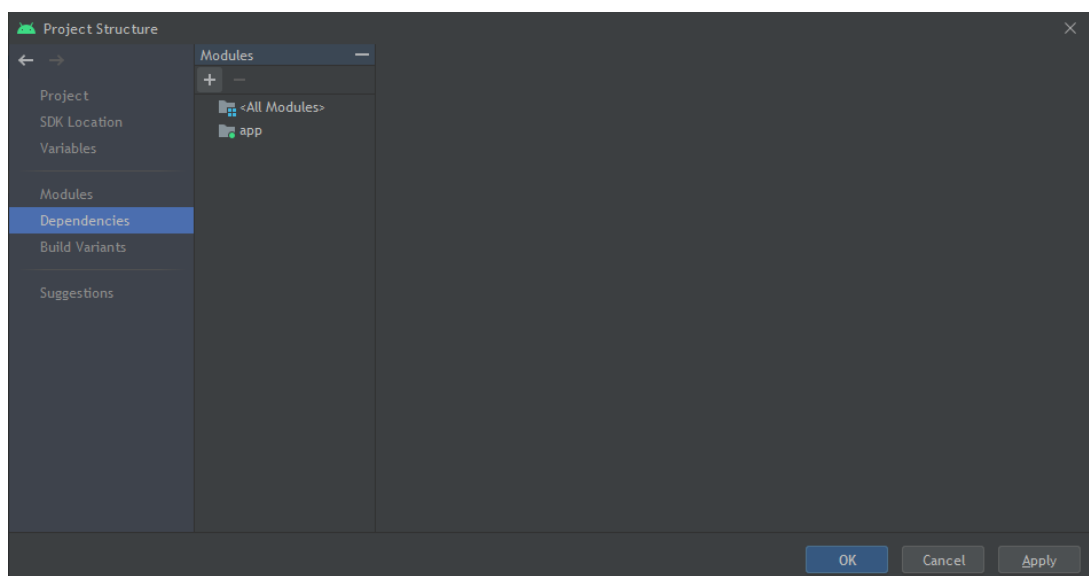
Esto es debido a que por defecto Android Studio usa R8 en vez de Proguard.

1.3 Integración de la librería

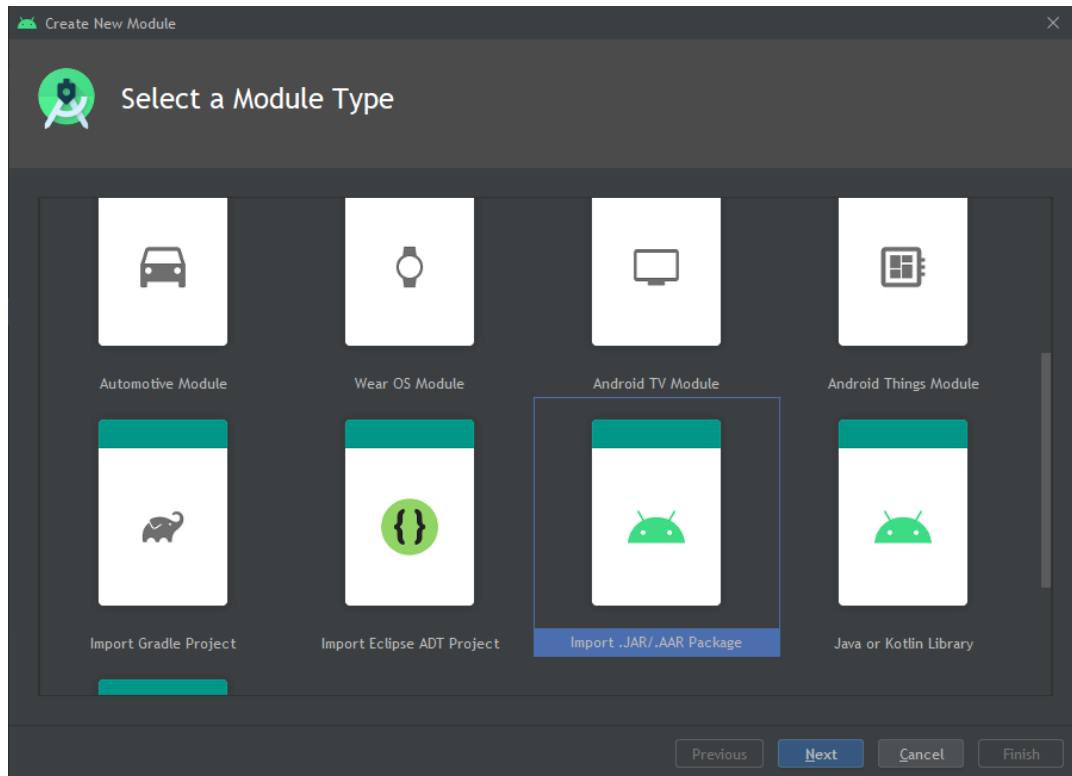
1.3.1 Importar módulo en Android Studio

Para la integración de la librería en Android Studio hay que realizar los siguientes pasos:

- 1) File -> Project Structure. Aquí vamos a Dependencies y hacemos click en el '+' para agregar una nueva



- 2) Seleccionamos Import .JAR/.AAR Package e introducimos la ruta donde se encuentra el SDK almacenado



- 3) Introducimos en el gradle de la aplicación (o del módulo que vaya a implementar el SDK) la siguiente línea:

```
implementation project(path: ':sdkinapp-library')
```

- 4) Además, es necesario introducir estas dependencias a nivel de aplicación:

```
implementation 'com.android.volley:volley:1.2.0'
implementation 'com.google.code.gson:gson:2.8.6'
implementation 'androidx.appcompat:appcompat:1.3.0'
implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
implementation 'androidx.legacy:legacy-support-v4:1.0.0'
implementation 'com.google.android.material:material:1.3.0'
implementation 'androidx.cardview:cardview:1.0.0'
implementation 'androidx.core:core:1.5.0'
```

1.3.2 Permisos en el Manifest

Para el correcto funcionamiento de la librería es necesario que dentro del fichero "AndroidManifest.xml" de la aplicación que hace uso de la misma, se declaren al menos, los siguientes permisos:

```
<uses-permission android:name="android.permission.INTERNET" />
```

2. Configuración

2.1 Parámetros obligatorios

Para utilizar la librería es necesario configurar una serie de parámetros **obligatorios** a través del objeto `TPVVConfiguration`.

2.1.1 Licencia de la aplicación

Es un código alfanumérico proporcionado por Redsys para validar las aplicaciones que hacen uso de la librería.

```
TPVVConfiguration.setLicense("XXXXXXXXXX");
```

2.1.2 Entorno

El entorno se selecciona utilizando las constantes definidas en el objeto de configuración `TPVVConstants`.

- Integración:

```
TPVVConfiguration.setEnvironment(TPVVConstants.ENVIRONMENT_INTEGRATION);
```

- Producción:

```
TPVVConfiguration.setEnvironment(TPVVConstants.ENVIRONMENT_REAL);
```

- Test:

```
TPVVConfiguration.setEnvironment(TPVVConstants.ENVIRONMENT_TEST);
```

2.1.3 Fuc del comercio

Código de identificación del comercio.

```
TPVVConfiguration.setFuc("XXXXXXXXXX");
```

2.1.4 Terminal

El terminal registrado para el comercio en cuestión.

```
TPVVConfiguration.setTerminal("XXX");
```

2.1.5 Código de la moneda utilizada

Este código sigue el estándar ISO 4217. Por defecto es "978" (euros).

```
TPVVConfiguration.setCurrency("978");
```


2.2 Parámetros opcionales

Son parámetros configurables por el comercio, pero que no son obligatorios para el funcionamiento básico de la librería.

- Nombre del titular: Nombre del titular del pago

```
TPVVConfiguration.setTitular("XXX");
```
- Métodos de Pago: (Aplica para el pago a través de WebView)
 - Forma de pago con tarjeta:

```
TPVVConfiguration.setPaymentMethods("TPVVConstants.PAYMENT_METHOD_T");
```
 - Forma de pago por transferencia:

```
TPVVConfiguration.setPaymentMethods("TPVVConstants.PAYMENT_METHOD_R");
```
 - Forma de pago por domiciliación:

```
TPVVConfiguration.setPaymentMethods("TPVVConstants.PAYMENT_METHOD_D");
```
 - Forma de pago con PayPal:

```
TPVVConfiguration.setPaymentMethods("TPVVConstants.PAYMENT_METHOD_P");
```
 - Forma de pago con PPII:

```
TPVVConfiguration.setPaymentMethods("TPVVConstants.PAYMENT_METHOD_Z");
```
 - A especificar por el comercio: String, diferentes formas de pago disponibles se encuentran en <https://pagosonline.redsys.es/parametros-entrada-salida.html>

```
TPVVConfiguration.setPaymentMethods("XXX");
```
- Url del comercio:

```
TPVVConfiguration.setMerchantUrl("XXX");
```
- Nombre del comercio:

```
TPVVConfiguration.setMerchantName("XXX");
```
- Datos adicionales del comercio:

```
TPVVConfiguration.setMerchantData("XXX");
```
- Descripción del comercio:

```
TPVVConfiguration.setMerchantDescriptor("XXX");
```
- FUC del grupo de comercio: Es necesario configurarlo para poder utilizar referencias entre comercios pertenecientes al mismo grupo de comercio.

```
TPVVConfiguration.setGroup("XXX");
```

3. Operaciones disponibles

El Framework implementa dos métodos que permiten realizar las operativas de **Pago Directo** y **Pago WebView**. Estas operativas tienen tres modos de uso:

- **Pago normal:** se introducen los datos de la tarjeta (número de tarjeta, caducidad y código de seguridad) para realizar la operación.
- **Pago con solicitud de referencia:** se introducen los datos de la tarjeta y en el resultado de la operación el servidor devuelve una referencia que se puede utilizar en futuros pagos.
- **Pago con referencia:** no es necesario introducir los datos de la tarjeta, se utiliza la referencia obtenida anteriormente en el pago con solicitud de referencia.

Para realizar un pago con cualquiera de estas operativas es **necesario** haber configurado al menos los parámetros obligatorios en el objeto `TPVVConfiguration`. Todos los parámetros opcionales pueden tener valor null.

En la clase `TPVV` se encuentran los dos métodos para realizar las diferentes operativas. Ambos métodos requieren de los mismos parámetros que se describen a continuación.

Variable	Descripción	Tipo
Context context	Contexto de la actividad que está llamando a la función.	Obligatorio
String orderCode	Código alfanumérico identificativo de la operación. Debe ser único por cada operación.	Obligatorio
Double amount	Importe de la operación	Obligatorio
String operationType	Se utilizan las constantes definidas en <code>TPVVConstants</code> <ul style="list-style-type: none"> ▪ <code>TPVVConstants.PAYMENT_TYPE_NORMAL</code> ▪ <code>TPVVConstants.PAYMENT_TYPE_PREAUTHORIZATION</code> ▪ <code>TPVVConstants.PAYMENT_TYPE_TRADITIONAL</code> ▪ <code>TPVVConstants.PAYMENT_TYPE_AUTHENTICATION</code> 	Obligatorio
String merchantIdentifier	En caso de tener un valor nulo realiza un pago normal, en vez de introducir una referencia valida se realiza un pago con referencia y, por último, para realizar un pago con solicitud de referencia hay que utilizar el valor de la constante <code>TPVVConstants.PAYMENT_TYPE_AUTHENTICATION</code>	Opcional
String productDescription	Descripción del producto o pago	Opcional
HashMap<String, String> extraParams	Tabla hash con clave-valor para introducir variables adicionales que no están contempladas en el SDK. NO INTRODUCIR parámetros que se encuentren en <code>TPVVConfiguration</code> .	Opcional
IPaymentResult callback	Objeto que implementa la interfaz <code>IPaymentResult</code> . Se llamará a las funciones definidas y se recibirá un objeto <code>ResultResponse</code> o <code>ErrorResponse</code> en función del resultado de la operación. Ver punto 4.	Obligatorio

3.1 Pago directo

3.1.1 Configuración del Pago Directo

El pago directo se efectúa y se gestiona la respuesta desde el mismo dispositivo. El método con el que se ejecuta es el siguiente:

```
public static void doDirectPayment(Context context, String orderCode,
    Double amount, String operationType, String merchantIdentifier,
    String productDescription, HashMap<String, String> extraParams,
    final IPaymentResult callback);
```

Se lanzará una nueva Activity en la que se introducirán los datos de la tarjeta para el pago y finalizar la transacción.

3.1.2 Personalización de la pantalla de Pago Directo (Opcional)

Es posible personalizar algunos aspectos de la pantalla del pago directo tales como el logo, colores y literales.

Para llevar a cabo la configuración de la pantalla del pago directo se hace uso del objeto del tipo `UIDirectPaymentConfig`. Si se quiere cambiar el idioma de esta actividad hay que hacerlo personalizando los textos con las setters proporcionados (se recomienda el uso de strings y la localización de Android Studio).

Son personalizables los siguientes elementos de la pantalla:

Elementos	Descripción
logo	Logo que aparece en la parte superior de la pantalla.
progressBarColor	Color de la barra de progreso.
topBarColor	Color de la barra superior de la actividad.
backgroundColor	Color del fondo de la actividad.
cardHeadText, cardHeadTextColor, cardHeadBackgroundTextColor	Texto, color del texto y color de fondo de la cabecera de la tarjeta.
cardText, cardTextColor	Texto de informativo y de error para el campo del número de tarjeta.
expiryText, expiriTextError	Texto de informativo y de error para el campo de la caducidad.
cvvText, cvvTextError	Texto de informativo y de error para el campo del CVV.
btnText, btnTextColor, btnColor	Texto, color y color del texto del botón para continuar.
infoText, infoTextColor	Texto y color de fondo del texto descriptivo.

3.2 Pago WebView

El Pago WebView se efectúa en un WebView que carga la URL del comercio proporcionado para efectuar el pago, dependerá del tipo de pago que se haya configurado en [TPVVConfiguration](#). La respuesta se gestiona en el dispositivo. El método con el que se ejecuta es el siguiente:

```
public static void doWebViewPayment(Context context, String orderCode,
    Double amount, String operationType, String merchantIdentifier,
    String productDescription, HashMap<String, String> extraParams,
    final IPaymentResult callback);
```

Se lanzará una nueva Activity con un WebView en el que se cargará la URL que corresponda. En caso de no configurar URLs de redirección, una vez terminada la operación se vuelve al flujo principal de la aplicación.

Para esta operativa, la función `onBackPressed()` está controlada para que no se cancele la operación usando el botón de "atrás". En caso de querer cancelar la operación, sería necesario pulsar sobre el botón "Cancelar" dentro de la web para así cerrar el WebView y devolver el control a la aplicación.

3.2.1 Configuración del idioma

En el caso del pago a través de WebView es posible configurar el idioma utilizando [códigos de idioma válidos](#). En caso de no indicar ninguno se configura por defecto en español.

```
TPVVConfiguration.setLanguage("XXX");
```

3.2.2 Configurar redirección a URLs de Resultado

Existe la posibilidad de redireccionar a una URL definida por el desarrollador tras realizar la operación. Una para el caso en que la operación se realice correctamente y otra en caso de error.

En el caso de redireccionar a una URL resultado configurada por el comercio es posible volver al flujo principal de la aplicación pulsando el botón back del sistema.

```
TPVVConfiguration.setEnableRedirection(true)
TPVVConfiguration.setUrlOK("urlOK")
TPVVConfiguration.setUrlKO("urlKO")
```

3.2.3 Configurar Alerta Resultado

Se ha añadido la posibilidad de mostrar una alerta indicando el resultado de la operación una vez se ha realizado la redirección a la web configurada por el desarrollador. Esta alerta es configurable por el desarrollador.

```
TPVVConfiguration.setEnableResultAlert(true);
TPVVConfiguration.setResultAlertTextButtonOk("Continuar");
TPVVConfiguration.setResultAlertTextButtonKo("Continuar");
TPVVConfiguration.setResultAlertTextOk("Operación realizada
correctamente.");
TPVVConfiguration.setResultAlertTextKo("Se ha producido un error al
intentar realizar la operación.");
```

4. Forma de la respuesta y códigos de error

4.1 Códigos de error

Los códigos de error propios de la librería móvil son los siguientes:

Código	Descripción
11	La firma del mensaje no es correcta
29	Fallo en la criptografía del servicio
31	La aplicación es incorrecta
60	Formato de JSON incorrecto
61	Error al obtener la clave de firma del comercio
62	Tipo de firma del terminal no soportado
78	Error propio del TPV virtual de Redsys (SIS)*
5548	Error genérico (Detalles en la descripción)
5550	Error de conexión (Detalles en la descripción)
5551	Sin parámetros en la respuesta

Los errores con código 78 contienen en su descripción el error correspondiente del TPV virtual de Redsys. Para más información consultar el documento: "TPV-Virtual Manual Integración – Redirección"

4.2 ResultResponse

Objeto con los datos de respuesta de una operación con éxito. Contiene los siguientes atributos:

Variable	Descripción
String amount	Cantidad total de la operación.
String currency	Moneda usada en la operación.
String order	Número de pedido.
String merchantCode	FUC del comercio.
String terminal	Número del terminal.
String responseCode	Código de la respuesta.
String authorisationCode	Código de autorización.
String transactionType	Tipo de transacción.
String securePayment	Pago seguro.
String language	Idioma de la operación.
String cardNumber	Número de la tarjeta utilizada.
String cardType	Tipo de tarjeta utilizada.
String merchantData	Datos del comercio.
String cardCountry	País de origen de la tarjeta utilizada.
String date	Fecha de la operación. (Solo aplica a pago a través de WebView)
String hour	Hora de la operación. (Solo aplica a pago a través de WebView)
String identifier	Referencia para el pago con referencia.
String signature	Firma.
String cardBrand	Marca de la tarjeta.
String expiryDate	Fecha de caducidad de la tarjeta.
String extraParams	Parámetros extra que se han recibido en la respuesta en formato JSON.

4.3 ErrorResponse

Objeto con los datos de respuesta de una operación fallida. Contiene los siguientes atributos:

Variable	Descripción
String code	Código de error.
String desc	Descripción del error que se ha producido.

5. ANEXO:

5.1 Lista de idiomas

Código	Idioma
0	Por defecto
1	Español
2	English - Ingles
3	Català
4	Français - Frances
5	Deutsch - Aleman
6	Nederlands - Holandes
7	Italiano
8	Svenska - Sueco
9	Português
10	Valencià
11	Polski - Polaco
12	Galego
13	Euskara
100	български език - Bulgaro (Bulgaro)
156	Chino
191	Hrvatski - Croata (Croatian)
203	Čeština - Checo (čeština Czech)
208	Dansk - Danes
233	Eesti keel - Estonio (Estonian)
246	Suomi - Finlandes (Finnish)
300	ελληνικά - Griego (Greek)
348	Magyar - Hungaro (Hungarian)
392	Japonés
428	Latviešu valoda - Leton (Latvian Latviešu valoda)
440	Lietuvių kalba - Lituano (Lithuanian u)
470	Malti - Maltes (Maltese)
642	Română - Rumano (Romanian româna)
643	русский язык - Ruso (Russkiy)
703	Slovenský jazyk - Eslovaco (Slovak slovenský jazyk)
705	Slovenski jezik - Esloveno (Slovenian)
792	Türkçe - Turco

5.2 Ejemplos de Implementación

5.2.1 Ejemplo Pago directo Normal

```
TPVV.doDirectPayment(getApplicationContext(), "orderCode01",
Double.valueOf(3050), TPVVConstants.PAYMENT_TYPE_NORMAL, null, "Descripción",
null, new IPaymentResult() {
    @Override
    public void paymentResultOK(ResultResponse response) {
        //Implementar comportamiento para operación correcta
    }

    @Override
    public void paymentResultKO(ErrorResponse error) {
        //Implementar comportamiento para operación fallida
    }
});
```

5.2.2 Ejemplo Pago directo con solicitud de referencia

```
TPVV.doDirectPayment(getApplicationContext(), "orderCode02",
Double.valueOf(3050), TPVVConstants.PAYMENT_TYPE_NORMAL,
TPVVConstants.REQUEST_REFERENCE, "Descripción", null, new IPaymentResult() {
    @Override
    public void paymentResultOK(ResultResponse response) {
        //Implementar comportamiento para operación correcta
    }

    @Override
    public void paymentResultKO(ErrorResponse error) {
        //Implementar comportamiento para operación fallida
    }
});
```

5.2.3 Ejemplo Pago directo con referencia

```
TPVV.doDirectPayment(getApplicationContext(), "orderCode02",
Double.valueOf(3050), TPVVConstants.PAYMENT_TYPE_NORMAL, reference,
"Descripción", null, new IPaymentResult() {
    @Override
    public void paymentResultOK(ResultResponse response) {
        //Implementar comportamiento para operación correcta
    }

    @Override
    public void paymentResultKO(ErrorResponse error) {
        //Implementar comportamiento para operación fallida
    }
});
```

Donde <reference> es el valor de la referencia obtenido previamente en un pago con solicitud, se puede recoger en el parámetro `ResultResponse.getIdentifier()`.

5.2.4 Ejemplo Configuración Pantalla Pago directo

```

Bitmap bm = BitmapFactory.decodeResource(getResources(),
R.drawable.android_logo);

uiDirectPaymentConfig.setLogo(bm);

uiDirectPaymentConfig.setProgressBarColor("#FFFF0000");
uiDirectPaymentConfig.setTopBarColor("#FFFF0000");

uiDirectPaymentConfig.setBackgroundColor("#7792E484");

uiDirectPaymentConfig.setCardHeadText("INFORMACION TARJETA");
uiDirectPaymentConfig.setCardHeadTextColor("#FFFFFF");
uiDirectPaymentConfig.setCardHeadTextBackgroundColor("#FFFF0000");

uiDirectPaymentConfig.setCardText("TARJETA");
uiDirectPaymentConfig.setCardTextError("ERROR TARJETA");

uiDirectPaymentConfig.setExpiryText("CADUCIDAD");
uiDirectPaymentConfig.setExpiryTextError("CADUCIDAD ERROR");

uiDirectPaymentConfig.setCvvText("CVV CODE");
uiDirectPaymentConfig.setCvvTextError("CVV ERROR");

uiDirectPaymentConfig.setBtnText("BOTON");
uiDirectPaymentConfig.setBtnTextColor("#FFFFFF");
uiDirectPaymentConfig.setBtnBackgroundColor("#FFFF0000");

uiDirectPaymentConfig.setInfoText("TEXTO DESCRIPCION");
uiDirectPaymentConfig.setInfoTextColor("#FFFF0000");

TPVVConfiguration.setUiDirectPaymentConfig(uiDirectPaymentConfig);

```



5.2.5 Ejemplo Pago WebView normal

```
TPVV.doWebViewPayment(getApplicationContext(), "orderCode11",
Double.valueOf(3050), TPVVConstants.PAYMENT_TYPE_NORMAL, null, "Descripción",
null, new IPaymentResult() {
    @Override
    public void paymentResultOK(ResultResponse response) {
        //Implementar comportamiento para operación correcta
    }

    @Override
    public void paymentResultKO(ErrorResponse error) {
        //Implementar comportamiento para operación fallida
    }
});
```

5.2.6 Ejemplo Pago WebView con solicitud de referencia

```
TPVV.doWebViewPayment(getApplicationContext(), "orderCode12",
Double.valueOf(3050), TPVVConstants.PAYMENT_TYPE_NORMAL,
TPVVConstants.REQUEST_REFERENCE, "Descripción", null, new IPaymentResult() {
    @Override
    public void paymentResultOK(ResultResponse response) {
        //Implementar comportamiento para operación correcta
    }

    @Override
    public void paymentResultKO(ErrorResponse error) {
        //Implementar comportamiento para operación fallida
    }
});
```

5.2.7 Ejemplo pago WebView con referencia

```
TPVV.doWebViewPayment(getApplicationContext(), "orderCode12",
Double.valueOf(3050), TPVVConstants.PAYMENT_TYPE_NORMAL, reference,
"Descripción", null, new IPaymentResult() {
    @Override
    public void paymentResultOK(ResultResponse response) {
        //Implementar comportamiento para operación correcta
    }

    @Override
    public void paymentResultKO(ErrorResponse error) {
        //Implementar comportamiento para operación fallida
    }
});
```

Donde <reference> es el valor de la referencia obtenido previamente en un pago con solicitud, se puede recoger en el parámetro `ResultResponse.getIdentifier()`.

5.2.8 Pagos utilizando extraParams

```
TPVV.doWebViewPayment(getApplicationContext(), "orderCode11",
Double.valueOf(3050), TPVVConstants.PAYMENT_TYPE_NORMAL, , "Descripción", null,
new IPaymentResult() {
    @Override
    public void paymentResultOK(ResultResponse response) {
        //Implementar comportamiento para operación correcta
    }

    @Override
    public void paymentResultKO(ErrorResponse error) {
        //Implementar comportamiento para operación fallida
    }
});
```

5.2.9 Configuración del color del ProgressBar

El color del progressBar es configurable tanto para el pago directo como para el pago a través de WebView.

```
TPVVConfiguration.setProgressBarColor("#FE9A2E");
```

5.2.10 Uso de JSONParser

Esta clase contiene herramientas para la transformación de Strings en formato JSON en tablas hash con clave-valor (String-String) y viceversa.

```
String json = "{\"variable1\":\"valor1\", \"variable2\":\"valor2\", \"variable3\":\n{\"var1\":\"vall\"}}\"";

HashMap<String, String> variables = JSONParser.JSONtoHashMap(json);

variables.getKey("variable1"); // devuelve un String con valor: "valor1"
variables.getKey("variable2"); // devuelve un String con valor: "valor2"
variables.getKey("variable3"); // devuelve un String con valor: "{\"var1\":\"vall\"}"

HashMap<String, String> variables = new HashMap<String, String>();

variables.put("variable1", "valor1");
variables.put("variable2", "{\"var1\":\"vall\"}");
variables.put("variable3", "valor3");

String json = JSONParser.HashMaptoJSON(variables);

Log.d(json); //Valor en Logcat: "{\"variable1\":\"valor1\", \"variable2\":\n{\"var1\":\"vall\"}, \"variable3\": \"valor3\"}"
```