Manual de referencia

SDK Anexo 20 versión 3.3

CONTPAQ i®

Software empresarial fácil y completo

Contenido

Asignar uso de CFDI

Para asignar el uso que se le dará al CFDI por medio de las funciones del SDK de los sistemas comerciales **AdminPAQ** y **CONTPAQi® Factura Electrónica**, es necesario utilizar la función de bajo nivel **fSetDatoCteProv**.

Esta función lleva como parámetros en primer lugar el campo que se va a afectar en la base de datos, en este caso el campo es **cUsoCFDI** y como segundo parámetro se le asignara el valor a dicho campo.

A continuación, se muestra un pequeño ejemplo de cómo asignar el uso del CFDI a un cliente y proveedores.

Declaración de la función

```
[DllImport("MGW_SDK.DLL")]
public static extern Int32 fSetDatoCteProv(string aCampo, string aValor);

[DllImport("MGW_SDK.DLL")]
public static extern Int32 fEditaCteProv();

[DllImport("MGW_SDK.DLL")]
public static extern Int32 fGuardaCteProv();
```

Implementación de la función

```
SDK.fEditaCteProv();
SDK.fSetDatoCteProv("cUsoCFDI", "G03");
lError = SDK.fGuardaCteProv();

if (lError != 0)
    SDK.rError(lError);
else
    MessageBox.Show("Datos del cliente grabados correctamente");
```

Nota: Este campo será visualizado en la tabla **MGW10002 y catalogo - Clientes y Proveedores**

Asignar clave del SAT al producto

Para asignar la clave del SAT al producto por medio de funciones del SDK, es necesario utilizar la función de bajo nivel **fSetDatoProducto**. Esta función lleva como parámetros el campo que se va a afectar en la base de datos, en este caso el campo es **cClaveSAT** y como segundo parámetro se le asignara el valor a dicho campo.

A continuación, se muestra un pequeño ejemplo de cómo asignar la clave SAT al producto

Declaración de la función

```
[DllImport("MGW_SDK.DLL")]
public static extern Int32 fEditaProducto();

[DllImport("MGW_SDK.DLL")]
public static extern int fSetDatoProducto(string aCampo, string aValor);

[DllImport("MGW_SDK.DLL")]
public static extern Int32 fGuardaProducto();
```

Implementación de la función

```
SDK.fEditaProducto();
SDK.fSetDatoProducto("cClaveSAT", "11101600");
lError = SDK.fGuardaProducto();

if (lError != 0)
    SDK.rError(lError);
else
    MessageBox.Show("Datos del producto generados correctamente");
```

Asignar la clave del SAT de pedimentos

Para el caso de la **Clave SAT** en pedimentos se modificó la función de **fSetDatoMovimiento** donde se manda el campo "**cClaveSat¬Pedimento**" cuando el producto solo cuenta con pedimentos o "**cClaveSat¬Serie**" cuando el producto cuenta con series y pedimentos.

Esta función debe ser utilizada después de hacer uso de la función **fAltaMovimientoSeriesCapas**.

Para hacer uso de la función **fSetDatoMovimiento**, no es necesario hacer uso de las funciones para editar y guardar este dato.

Nota: Se debe asignar de acuerdo a la estructura del SAT:

- últimos dos dígitos del año de validación seguidos por dos espacios, 2 dígitos de la aduana de despacho seguido por dos espacios, 4 dígitos del número de la patente seguidos por dos espacios, 1 dígito que corresponde al último dígito del año en curso, salvo que se trate de un pedimento consolidado iniciando en el año inmediato anterior o del pedimento original de una rectificación, seguido de 6 dígitos de la numeración progresiva por aduana.

  Longitud 21. Patrón: [0-9]{2} [0-9]{2} [0-9]{4} [0-9]{7}

A continuación, se muestra un pequeño ejemplo de cómo asignar la clave SAT del pedimento o de la serie en un documento de entrada al almacén.

Declaración de la función

```
[DllImport("MGW_SDK.DLL")]
public static extern Int32 fSetDatoMovimiento(string aCampo, string aValor);
```

Implementación de la función

```
lError = SDK.fAltaMovimientoSeriesCapas(idMto, ref lSeriesCapas);
if (lError != 0)

    SDK.rError(SDK.lError);
else
    SDK.lError = SDK.fSetDatoMovimiento("cClaveSat¬Pedimento", "17  30  3435  7059332");
```

Funciones de timbrado

Las funciones **fEmitirDocumento** y **fTimbraXML** tuvieron cambios cuando se presente alguno de los siguientes casos:

- **Cuando el valor del campo TipoCambio se encuentre fuera de los límites establecidos.**

Si el valor está fuera del porcentaje aplicable a la moneda, tomado del catálogo **c_Moneda**, el emisor debe obtener del proveedor de certificación de CFDI que vaya a timbrar el CFDI, de manera no automática, una clave de confirmación para ratificar que el valor es correcto e integrar dicha clave en el campo Confirmación.

El límite superior se obtiene multiplicando el valor publicado del tipo de cambio por la suma de uno más el porcentaje aplicable a la moneda tomado del catálogo **c_Moneda**.

El límite inferior se obtiene multiplicando el valor publicado del tipo de cambio por la suma de uno menos el porcentaje aplicable a la moneda tomado del catálogo **c_Moneda**. Si este límite fuera negativo se toma cero.

- **Cuando el valor del campo Total se encuentre fuera de los límites establecidos.**

Se recomienda revisar el catálogo de **c_TipoDeComprobante** que se encuentra en la página del SAT, ya que el límite establecido depende de cada tipo de comprobante.

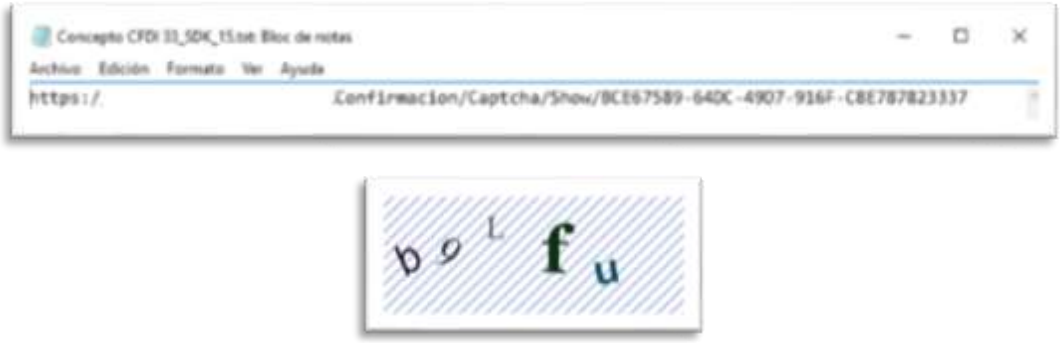
Al realizar el timbrado en alguno de los casos anteriores se mostrará uno de los siguientes mensajes dependiendo el caso:

166184 = Cuando el valor del campo Total se encuentre fuera de los límites establecidos, debe existir el campo: **confirmacion**.

166182 = Cuando el valor del campo **TipoCambio** se encuentre fuera de los límites establecidos, debe existir el campo **confirmacion**.

a) Al utilizar la función **fEmitirDocumento** se generan dos archivos en la ruta de la empresa:

El archivo con extensión **.txt** contiene la URL de la cadena de confirmación.



En el archivo con extensión ".**confirmacion**", se debe capturar y guardar el código mostrado al ingresar al URL que se encuentra en el archivo con extensión **.txt**

Posteriormente se vuelve a realizar el timbrado por medio de la función **fEmitirDocumento**

b) En el caso de la función de **fTimbraXML** solo se generará el archivo de texto con la URL de la cadena de confirmación, este archivo se genera en la ruta donde se encuentre el XML que se desea timbrar.

   Se deberá de modificar el XML para asignar el atributo de confirmación a nivel comprobante, en este atributo se debe capturar el código que aparezca al ingresar a la URL que se generó en el archivo .txt

Posteriormente se debe volver a mandar a timbrar el XML.

Timbrado de Pagos XML

Para realizar el timbrado de un XML que sean Recibos Electrónico De Pagos, es necesario utilizar las funciones **fTimbraComplementoPagoXML** y **fInicializaLicenseInfo**.

La función **fTimbraComplementoPagoXML** utiliza siete parámetros, estos parámetros son los siguientes:

Parámetro	Tipo	Descripción
aRutaXML	String	Ruta del XML que se requiere timbrar
aCodConcepto	String	Código del concepto que se utilizara para timbrar
aUUID	StringBuilder	Regresa el UUID del documento timbrado
aRutaDDA	String	Ruta del archivo dda a utilizar
aRutaResultado	String	Ruta de entrega del XML timbrado
aPass	String	Contraseña del certificado utilizado para timbrar
aRutaFormato	String	Ruta y nombre del formato amigable a utilizar

La función **fInicializaLicenseInfo** se encarga de leer la licencia del sistema, esta función utiliza un parámetro de tipo byte con el cual se indicará el sistema con el que se desea trabajar.

- 0 = AdminPAQ
- 1 = Factura electrónica

Declaración de las funciones

```
[DllImport("MGW_SDK.DLL")]
public static extern int fInicializaLicenseInfo(byte aSistema);

[DllImport("MGW_SDK.dll")]
public static extern int fTimbraComplementoPagoXML(string aRutaXML, string aCodConcepto,
    StringBuilder aUUID, string aRutaDDA, string aRutaResultado, string aPass, string aRutaFormato);
```

Implementación de la función

```
string rutaXML = @"C:\Compacw\Empresas\Admin\XML_RecepPagos\Pago.XML";
string concepto = "102017";
StringBuilder UUID = new StringBuilder(300);
string rutaDDA = "";
string rutaResultado = @"C:\Compacw\Empresas\Admin\Timbrados";
string pass = "12345678a";
string rutaFormato = @"C:\Compacw\Empresas\Reportes\Facturacion\Plantilla_Factura_cfdi_1.htm";

lError = SDK.fInicializaLicenseInfo(0);
lError = SDK.fTimbraComplementoPagoXML(rutaXML, concepto, UUID, rutaDDA,
                rutaResultado, pass, rutaFormato);
if (lError != 0)
    SDK.rError(lError);
else
    MessageBox.Show($"UUID: {UUID.ToString()}");
```

Nota:

Para hacer uso de esta función, es necesario contar con una licencia de 5 o más usuarios.

Generar documentos de pago y emitirlos

Para generar documentos de “Recibo electrónico de pago”, es necesario hacer uso de dos estructuras, tDocumento y RegLlaveDoc, la primera de ellas nos ayudara a ingresar los datos necesarios para el encabezado del documento, y la segunda estructura sera utilizada dos veces, ya que una sera para capturar los datos de la factura que se desea pagar, y la siguiente sera utilizada para capturar los datos del documento de pago.

Declaración de las estructuras

Documentos

```
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack = 4)]
public struct tDocumento
{

    public Double aFolio;
    public int aNumMoneda;
    public Double aTipoCambio;
    public Double aImporte;
    public Double aDescuentoDoc1;
    public Double aDescuentoDoc2;
    public int aSistemaOrigen;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongCodigo)]
    public String aCodConcepto;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongSerie)]
    public String aSerie;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongFecha)]
    public String aFecha;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongCodigo)]
    public String aCodigoCteProv;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongCodigo)]
    public String aCodigoAgente;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongReferencia)]
    public String aReferencia;
    public double aAfecta;
    public double aGasto1;
    public double aGasto2;
    public double aGasto3;

}
```

RegLlaveDoc

```
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack = 4)]
public struct RegLlaveDoc
{
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongCodigo)]
    public String aCodConcepto;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongSerie)]
    public String aSerie;
    public double folio;
}
```

Funciones a utilizar

Para generar documentos de “Recibo electrónico de pago” es necesario hacer uso de las siguientes funciones:

Declaración de las funciones

fAltaDocumentoCargoAbono

```
[DllImport("MGW_SDK.DLL")]
public static extern Int32 fAltaDocumentoCargoAbono(ref tDocumento atDocumento);
```

Esta función generará el encabezado del documento en base a los datos proporcionados en la estructura.

fEditarDocumento

```
[DllImport("MGW_SDK.DLL")]
public static extern Int32 fEditarDocumento();
```

Esta función nos ayudara a poner en modo edición el registro actual en la tabla MGW10008 (Documentos).

fSetDatoDocumento

```
[DllImport("MGW_SDK.DLL")]
public static extern Int32 fSetDatoDocumento(string aCampo, string aValor);
```

Esta función nos ayudará a ingresar información en algún campo indicado por el parámetro aCampo, esta información se verá reflejado en la tabla MGW10008 (Documentos).

fGuardaDocumento

```
[DllImport("MGW_SDK.DLL")]
public static extern Int32 fGuardaDocumento();
```

Esta función guarda los cambios realizados por la función fSetDatoDocumento.

fAfectaDocto_Param

```
[DllImport("MGW_SDK.DLL")]
public static extern Int32 fAfectaDocto_Param([MarshalAs(UnmanagedType.LPStr)] string aCodConcepto,
    [MarshalAs(UnmanagedType.LPStr)] string aSerie, double aFolio, bool aAfecta);
```

Esta función nos ayudará a realizar la afectación de los documentos, esta afectación se realizará para los saldos del cliente, existencias del producto, etc.

fSaldarDocumento

```
[DllImport("MGW_SDK.DLL")]
public static extern Int32 fSaldarDocumento(ref RegLlaveDoc astDocAPagar,ref  RegLlaveDoc astDocPago,
    double importe, int moneda, string aFecha);
```

Esta función nos ayudará a saldar o abonar el importe indicado al documento deseado, esta información se pasará por medio de los parámetros de la función.

fInicializaLicenseInfo

```
[DllImport("MGW_SDK.DLL")]
public static extern int fInicializaLicenseInfo(byte aSistema);
```

Esta función nos ayudará a leer la licencia del sistema.
*Esta licencia debe ser de 5 usuarios

fEmitirDocumento

```
[DllImport("MGW_SDK.DLL")]
public static extern Int32 fEmitirDocumento([MarshalAs(UnmanagedType.LPStr)] string aCodConcepto,
    [MarshalAs(UnmanagedType.LPStr)] string aSerie, double aFolio,
    [MarshalAs(UnmanagedType.LPStr)] string aPassword,
    [MarshalAs(UnmanagedType.LPStr)] string aArchivoAdicional);
```

Esta función nos ayudara a realizar el timbrado del documento de pago.

Implementación de la función

```
double folio = 0;

SDK.tDocumento DoctoCargoAbono = new SDK.tDocumento();
SDK.RegLlaveDoc saldar = new SDK.RegLlaveDoc();
SDK.RegLlaveDoc pago = new SDK.RegLlaveDoc();

var concepto = selecconcepto(cmbConceptos.SelectedItem.ToString());
lError = SDK.fSiguienteFolio(concepto, serie, ref folio);

DoctoCargoAbono.aCodConcepto = concepto;
DoctoCargoAbono.aFolio = folio;
DoctoCargoAbono.aSerie = "";
DoctoCargoAbono.aFecha = DateTime.Today.ToString("MM/dd/yyyy");
DoctoCargoAbono.aCodigoCteProv = "CL001";
DoctoCargoAbono.aTipoCambio = 1;
DoctoCargoAbono.aNumMoneda = 1;
DoctoCargoAbono.aImporte = 116;
DoctoCargoAbono.aReferencia = "DOCTO SDK";
lError= SDK.fAltaDocumentoCargoAbono(ref DoctoCargoAbono);
if (lError != 0)
{
    SDK.rError(lError);
}
else
{
    MessageBox.Show("Documento creado");
    SDK.fEditarDocumento();
    //SDK.fSetDatoDocumento("CIDCUENTA", "2");//Identificador de la cuenta bancaria de la empresa
    SDK.fSetDatoDocumento("CMETODOPAG", "01");//Forma de pago
    SDK.fSetDatoDocumento("CCANTPARCI", "1");//Cantidad de parcialidades
    SDK.fSetDatoDocumento("CNUMPARCIA", "1");//Numero de parcialidad
    //SDK.fSetDatoDocumento("CIDMONEDCA", "3");//Identificador de la cuenta bancaria del cliente
    SDK.fGuardaDocumento();

    saldar.aCodConcepto = "102017";//codigo de concepto del documento a saldar
    saldar.aSerie = "";//Serie del documento a saldar
    saldar.folio = 20;//Folio del documento a saldar

    pago.aCodConcepto = concepto;//Concepto del documento de pago
    pago.aSerie = "";//Serie del documento de pago
    pago.folio=folio;//Folio del documento de pago
```

```
    lError = SDK.fAfectaDocto_Param(concepto, "", folio, true);

    lError = SDK.fSaldarDocumento(ref saldar, ref pago, 116, 1, DateTime.Now.ToString("MM/dd/yyyy"));
    if (lError != 0)
    {
        SDK.rError(lError);
    }
    else
    {
        MessageBox.Show("Documento saldado");
        SDK.fInicializaLicenseInfo(0);

        lError = SDK.fEmitirDocumento(concepto, "", folio, "12345678a", "");
        if (lError != 0)
        {
            SDK.rError(lError);
        }
        else
        {
            MessageBox.Show("Documento timbrado");

        }
    }

}
```

Relacionar CFDI's

Recuperar relaciones

Para recuperar o identificar las relaciones que tiene un documento, se utiliza la función fRecuperarRelacionesCFDIs.

La función fRecuperarRelacionesCFDIs utiliza seis parámetros, los parámetros son los siguientes:

Parámetro	Tipo	Descripción
aCodConcepto	string	Código del concepto del documento
aSerie	string	Serie del documento
aFolio	string	Folio del documento
aTipoRelación	string	Tipo de relación del documento
aUUIDs	StringBuilder	Se almacenan los UUID's relacionados al documento separados por el carácter |
aRutaNombreArchivoInfo	string	Ruta de un archivo .txt previamente generado, en este archivo se escribirán los UUID's que tenga relacionado el documento

Declaración de la función

```
[DllImport("MGW_SDK.DLL")]
public static extern int fRecuperarRelacionesCFDIs(string aCodConcepto, string aSerie, string aFolio,
    string aTipoRelación, StringBuilder aUUIDs, string aRutaNombreArchivoInfo);
```

Agregar relación CFDI

La función fAgregarRelacionCFDI nos ayudara a relacionar otro documento existente en la base de datos.

La función fAgregarRelacionCFDI utiliza siete parámetros, los parámetros son los siguientes:

Parámetro	Tipo	Descripción
aCodConcepto	string	Código de concepto del documento
aSerie	string	Serie del documento
aFolio	string	Folio del documento
aTipoRelación	string	Tipo de relación que se realizara
aConceptoRelacionar	string	Código de concepto del documento a relacionar
aSerieRelacionar	string	Serie del documento a relacionar
aFolioRelacionar	string	Folio del documento a relacionar

Declaración de la función

```
[DllImport("MGW_SDK.DLL")]
public static extern int fAgregarRelacionCFDI(string aCodConcepto, string aSerie, string aFolio,
    string aTipoRelación, string aConceptoRelacionar, string aSerieRelacionar, string aFolioRelacionar);
```

Agregar relación CFDI 2

La función fAgregarRelacionCFDI2 nos ayudara a realizar una relación por UUID's.

Esta función requiere de cinco parámetros, los parámetros utilizados por la función son los siguientes:

Parámetro	Tipo	Descripción
aCodConcepto	string	Código de concepto del documento
aSerie	string	Serie del documento
aFolio	string	Folio del documento
aTipoRelación	string	Tipo de relación que se realizara
aUUID	string	UUID´s a relacionar

Nota:

Se podrá agregar más de un UUID, solo es necesario ingresar los UUID's separados por el carácter |

Ejemplo:

```
string uuids = "AC62B9D7-0388-4D40-8CCC-A1CAE0653559|DB54D8AE-0431-4ADA-A713-685FC8A0FCC1";
```

Declaración de la función

```
[DllImport("MGW_SDK.DLL")]
public static extern int fAgregarRelacionCFDI2(string aCodConcepto, string aSerie, string aFolio,
    string aTipoRelación, string aUUID);
```

Eliminar relaciones

Para poder eliminar las relaciones que tiene un documento se utilizara la función fEliminarRelacionesCFDIs.

Esta función requiere de tres parámetros, los parámetros requeridos por la función son los siguientes:

Parámetro	Tipo	Descripción
aCodConcepto	string	Código de concepto del documento
aSerie	string	Serie del documento
aFolio	string	Folio del documento

Declaración de la función

```
[DllImport("MGW_SDK.DLL")]
public static extern int fEliminarRelacionesCFDIs(string aCodConcepto, string aSerie, string aFolio);
```

Ejemplos

Conexión SDK y Abrir empresa

Declaración de las funciones

```
[DllImport("KERNEL32")]
public static extern int SetCurrentDirectory(string pPtrDirActual);

[DllImport("MGW_SDK.DLL")]
public static extern int fSetNombrePAQ(String aNombrePAQ);

[DllImport("MGW_SDK.DLL")]
public static extern int fAbreEmpresa(string Directorio);
```

Implementación de la función

```
int lError = 0;
string rutaEmpresa = @"C:\Compacw\Empresas\Admin";
string rutaBinarios = @"C:\Program Files (x86)\Compacw\AdminPAQ";
string sNombrePAQ = "AdminPAQ"; //Factura electrónica utiliza "CONTPAQ I Facturacion"

SDK.SetCurrentDirectory(rutaBinarios);
lError = SDK.fSetNombrePAQ(sNombrePAQ);
if (lError != 0)
{
    SDK.rError(lError);
}
else
{
    lError = SDK.fAbreEmpresa(rutaEmpresa);
    if (lError != 0)
    {
        SDK.rError(lError);
    }
    else
    {
        Console.WriteLine("Se abrio la empresa");
        Console.ReadKey(true);
    }
}
```

Nota:

Todos los ejemplos requieren de una conexión al SDK y contar con una empresa activa (Abierta).

Asignar uso de CFDI al cliente

Declaración de las funciones

```
[DllImport("MGW_SDK.DLL")]
public static extern int fBuscaCteProv(string codigo);

[DllImport("MGW_SDK.DLL")]
public static extern Int32 fEditaCteProv();

[DllImport("MGW_SDK.DLL")]
public static extern Int32 fSetDatoCteProv(string aCampo, string aValor);

[DllImport("MGW_SDK.DLL")]
public static extern Int32 fGuardaCteProv();
```

Implementación de la función

```
lError= SDK.fBuscaCteProv("CL003");
if (lError != 0)
{
    SDK.rError(lError);
}
else
{
    SDK.fEditaCteProv();
    lError = SDK.fSetDatoCteProv("cUsoCFDI", "G03");
    if (lError != 0)
    {
        SDK.rError(lError);
    }
    else
    {
        lError = SDK.fGuardaCteProv();
        if (lError != 0)
        {
            SDK.rError(lError);
        }
        else
        {
            Console.WriteLine("Cliente modificado");
        }
    }
}
```

Asignar clave SAT al producto

Declaración de las funciones

```
[DllImport("MGW_SDK.DLL")]
public static extern int fBuscaProducto(string Codigo);

[DllImport("MGW_SDK.DLL")]
public static extern Int32 fEditaProducto();

[DllImport("MGW_SDK.DLL")]
public static extern int fSetDatoProducto(string aCampo, string aValor);

[DllImport("MGW_SDK.DLL")]
public static extern Int32 fGuardaProducto();
```

Implementación de la función

```
lError = SDK.fBuscaProducto("PR001");
if (lError != 0)
{
    SDK.rError(lError);
}
else
{
    SDK.fEditaProducto();
    SDK.fSetDatoProducto("cClaveSAT", "11101600");
    lError = SDK.fGuardaProducto();
    if (lError != 0)
    {
        SDK.rError(lError);
    }
    else
    {
        Console.WriteLine("Clave SAT asignada");
    }
}
```

Asignar clave SAT de un pedimento

Declaración de las estructuras

```
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack = 4)]
public struct tDocumento
{

    public Double aFolio;
    public int aNumMoneda;
    public Double aTipoCambio;
    public Double aImporte;
    public Double aDescuentoDoc1;
    public Double aDescuentoDoc2;
    public int aSistemaOrigen;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongCodigo)]
    public String aCodConcepto;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongSerie)]
    public String aSerie;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongFecha)]
    public String aFecha;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongCodigo)]
    public String aCodigoCteProv;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongCodigo)]
    public String aCodigoAgente;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongReferencia)]
    public String aReferencia;
    public double aAfecta;
    public double aGasto1;
    public double aGasto2;
    public double aGasto3;

}
```

```
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack = 4)]
public struct tMovimiento
{

    public int aConsecutivo;
    public Double aUnidades;
    public Double aPrecio;
    public Double aCosto;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongCodigo)]
    public String aCodProdSer;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongCodigo)]
    public String aCodAlmacen;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongReferencia)]
    public String aReferencia;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongCodigo)]
    public String aCodClasificacion;
}

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi, Pack = 4)]
public struct tSeriesCapas
{
    public double aUnidades;
    public double aTipoCambio;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongCodigo)]
    public string aSeries;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongDescripcion)]
    public string aPedimento;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongDescripcion)]
    public string aAgencia;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongFecha)]
    public string aFechaPedimento;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongDescripcion)]
    public string aNumeroLote;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongFecha)]
    public string aFechaFabricacion;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = constantes.kLongFecha)]
    public string aFechaCaducidad;


}
```

Declaración de las funciones

```
[DllImport("MGW_SDK.DLL")]
public static extern Int32 fSiguienteFolio([MarshalAs(UnmanagedType.LPStr)] string aCodigoConcepto,
    [MarshalAs(UnmanagedType.LPStr)] StringBuilder aSerie, ref double aFolio);

[DllImport("MGW_SDK.DLL")]
public static extern Int32 fAltaDocumento(ref Int32 aIdDocumento, ref tDocumento atDocumento);

[DllImport("MGW_SDK.DLL")]
public static extern Int32 fAltaMovimiento( Int32 aIdDocumento, ref Int32 aIdMovimiento,
    ref tMovimiento astMovimiento);

[DllImport("MGW_SDK.DLL")]
public static extern Int32 fAfectaDocto_Param([MarshalAs(UnmanagedType.LPStr)] string aCodConcepto,
    [MarshalAs(UnmanagedType.LPStr)] string aSerie, double aFolio, bool aAfecta);

[DllImport("MGW_SDK.DLL")]
public static extern Int32 fAltaMovimientoSeriesCapas(int aIdMovimiento, ref tSeriesCapas lSeries);

[DllImport("MGW_SDK.DLL")]
public static extern Int32 fSetDatoMovimiento(string aCampo, string aValor);

[DllImport("MGW_SDK.DLL")]
public static extern Int32 fCalculaMovtoSerieCapa(int lIdMovimiento);
```

Implementación de la función

```
lDocto = new SDK.tDocumento();
lMovto = new SDK.tMovimiento();
lSeriesCapas = new SDK.tSeriesCapas();
serie = new StringBuilder();

double folio = 0;

var concepto = selecconcepto(cmbConceptos.SelectedItem.ToString());
SDK.lError = SDK.fSiguienteFolio(concepto, serie, ref folio);

lDocto.aCodConcepto = concepto;
lDocto.aFolio = folio;
lDocto.aFecha = DateTime.Today.ToString("MM/dd/yyyy");
lDocto.aTipoCambio = 1;
lDocto.aNumMoneda = 1;
lDocto.aSerie = "";
lDocto.aImporte = 0;
lDocto.aAfecta = 0;
lDocto.aDescuentoDoc1 = 0;
lDocto.aDescuentoDoc2 = 0;
```

```
lError = SDK.fAltaDocumento(ref idDocto, ref lDocto);
if (lError != 0)
{
    SDK.rError(lError);

}
else
{
    MessageBox.Show("Documeto Creado");
    lMovto.aCodAlmacen = "1";
    lMovto.aCodProdSer = "PEDI001";
    lMovto.aCosto = 50;
    lMovto.aPrecio = 100;
}

lError = SDK.fAltaMovimiento(idDocto, ref idMto, ref lMovto);
if (lError != 0)
{
    SDK.rError(lError);

}
else
{
    SDK.fAfectaDocto_Param(concepto, "", folio,true);

}

//SE LLENAN LAS CAPAS POR MEDIO DE LA ESTRUCTURA

lSeriesCapas.aUnidades = 1;
lSeriesCapas.aPedimento = "172433061478965";
lSeriesCapas.aAgencia = "30|CIUDAD REYNOSA, CIUDAD REYNOSA, TAMAULIPAS";
lSeriesCapas.aTipoCambio = 1;
lSeriesCapas.aFechaPedimento = DateTime.Today.ToString("MM/dd/yyyy");

//SE DAN DE ALTA LOS MOVIMIENTOS

lError = SDK.fAltaMovimientoSeriesCapas(idMto, ref lSeriesCapas);
if (lError != 0)
{
    SDK.rError(lError);

}
else
{
```

```
    lError = SDK.fSetDatoMovimiento("cClaveSat¬Pedimento", "17  30  3435  7059332");
    if (lError != 0)
    {
        SDK.rError(lError);
    }
    else
    {

        lError = SDK.fCalculaMovtoSerieCapa(idMto);
        if (lError != 0)
        {
            SDK.rError(lError);

        }
        else
        {
            MessageBox.Show("CALCULA SERIE CAPA");

        }
    }
}
```

Timbrado de pagos XML

Declaración de las funciones

```
[DllImport("MGW_SDK.DLL")]
public static extern int fInicializaLicenseInfo(byte aSistema);

[DllImport("MGW_SDK.dll")]
public static extern int fTimbraComplementoPagoXML(string aRutaXML, string aCodConcepto,
    StringBuilder aUUID, string aRutaDDA, string aRutaResultado, string aPass, string aRutaFormato);
```

Implementación de la función

```
string rutaXML = @"C:\Compacw\Empresas\Admin\XML_RecepPagos\Pago.XML";
string concepto = "102017";
StringBuilder UUID = new StringBuilder(300);
string rutaDDA = "";
string rutaResultado = @"C:\Compacw\Empresas\Admin\Timbrados";
string pass = "12345678a";
string rutaFormato = @"C:\Compacw\Empresas\Reportes\Facturacion\Plantilla_Factura_cfdi_1.htm";

lError = SDK.fInicializaLicenseInfo(0);
lError = SDK.fTimbraComplementoPagoXML(rutaXML, concepto, UUID, rutaDDA,
                    rutaResultado, pass, rutaFormato);
if (lError != 0)
    SDK.rError(lError);
else
    MessageBox.Show($"UUID: {UUID.ToString()}");
```

Recuperar relaciones CFDI

Declaración de las funciones

```
[DllImport("MGW_SDK.DLL")]
public static extern int fRecuperarRelacionesCFDIs(string aCodConcepto, string aSerie, string aFolio,
    string aTipoRelación, StringBuilder aUUIDs, string aRutaNombreArchivoInfo);
```

Implementación de la función

```
string concepto = "42017";
string serie = "";
string folio = "90";
string tipoRelacion = "07";
StringBuilder uuids = new StringBuilder(256);
string rutaArchivo = @"C:\Compac\Empresas\adComercial\info.txt";

lError = SDK.fRecuperarRelacionesCFDIs(concepto, serie, folio, tipoRelacion, uuids, rutaArchivo);
if (lError != 0)
    SDK.rError(lError);
else
    MessageBox.Show(aValor.ToString());
```

Agregar relación CFDI

Declaración de las funciones

```
[DllImport("MGW_SDK.DLL")]
public static extern int fAgregarRelacionCFDI(string aCodConcepto, string aSerie, string aFolio,
    string aTipoRelación, string aConceptoRelacionar, string aSerieRelacionar, string aFolioRelacionar);
```

Implementación de la función

```
string concepto = "42017";
string serie = "";
string folio = "90";
string tipoRelacion = "07";
string conceptoRelacionar = "42017";
string serieRelacionar = "";
string folioRelacionar = "222";

lError = SDK.fAgregarRelacionCFDI(concepto, serie, folio, tipoRelacion, conceptoRelacionar, serieRelacionar, folioRelacionar);
if (lError != 0)
    SDK.rError(lError);
else
    MessageBox.Show("Documento relacionado");
```

Agregar relación CFDI 2

Declaración de las funciones

```
[DllImport("MGW_SDK.DLL")]
public static extern int fAgregarRelacionCFDI2(string aCodConcepto, string aSerie, string aFolio,
    string aTipoRelación, string aUUID);
```

Implementación de la función

```
string concepto = "42017";
string serie = "";
string folio = "224";
string tipoRelacion = "07";
string uuids = "AC62B9D7-0388-4D40-8CCC-A1CAE0653559";

lError = SDK.fAgregarRelacionCFDI2(concepto, serie, folio, tipoRelacion, uuids);
if (lError != 0)
    SDK.rError(lError);
else
    MessageBox.Show("Documento Relacionado");
```

Eliminar relaciones CFDI

Declaración de las funciones

```
[DllImport("MGW_SDK.DLL")]
public static extern int fEliminarRelacionesCFDIs(string aCodConcepto, string aSerie, string aFolio);
```

Implementación de la función

```
string concepto = "42017";
string serie = "";
string folio = "224";

lError = SDK.fEliminarRelacionesCFDIs(concepto, serie, folio);
if (lError != 0)
    SDK.rError(lError);
else
    MessageBox.Show("Relaciones eliminadas");
```