SDK de los sistemas comerciales en C# .NET

contpaqi.com

Aviso de derechos del propietario

20140116

SDK de los sistemas comerciales en C# .NET

Visión general

Introducción

Este documento contiene la información necesaria para utilizar el SDK para el desarrollo de aplicaciones externas creadas en **C# .NET** que envíen información a los siguientes sistemas comerciales de **CONTPAQ i®**:

- **AdminPAQ**
- **CONTPAQ i® FACTURA ELECTRÓNICA**

Objetivo

En este documento conocerás:

- Cómo declarar funciones del SDK de los sistemas comerciales en C# .NET

En este documento

Este documento contiene los siguientes capítulos:

Introducción

Visión general

Introducción

En este capítulo conocerás los conceptos básicos del SDK de los sistemas comerciales, qué es, para qué sirve y dónde encontrar documentación para su uso.

Sistemas comerciales

El SDK de los sistemas comerciales de **CONTPAQ i®** que se explica en este documento funciona para:

- **AdminPAQ**
- **CONTPAQ i® FACTURA ELECTRÓNICA**

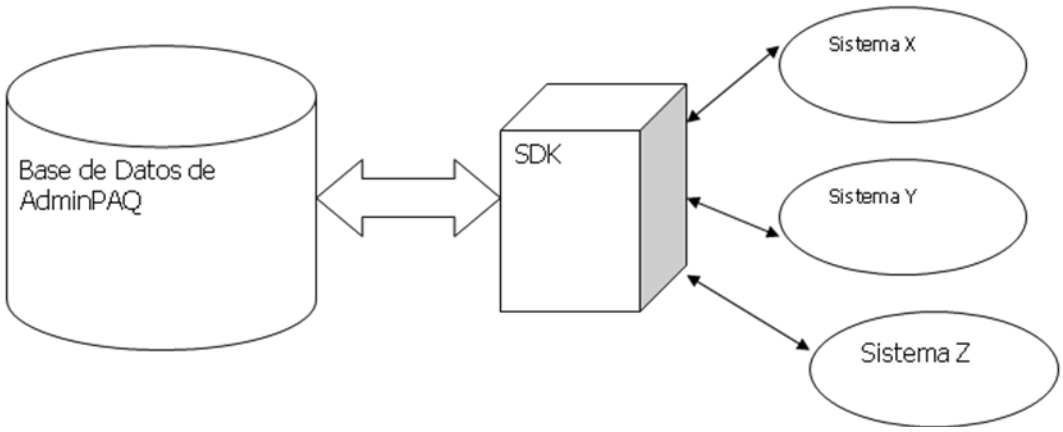
El SDK es básicamente el mismo para los 2 sistemas solo cambian algunas instrucciones que se explicarán a lo largo de este documento.

Qué es el SDK

El SDK es un grupo de funciones creadas en C++ que permiten leer y guardar información en la Base de Datos de los sistemas comerciales de **CONTPAQ i®**.

- El acceso del SDK es de lectura y escritura.
- Controla la concurrencia en un ambiente multiusuario.
- Respeta las reglas de negocio por lo que aseguran la integridad de la Base de Datos.
- No es necesario conocer la estructura de la base de datos.
- Estas funciones están almacenadas en el archivo **MGW_SDK.DLL**

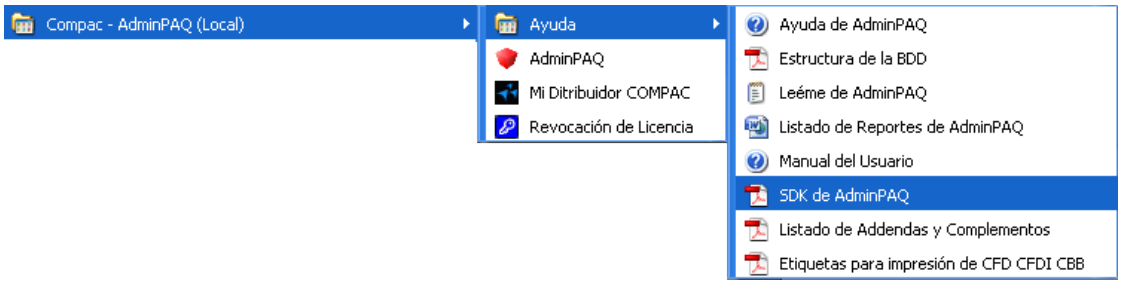
Nota: El SDK solo funciona en versiones de Office de 32 bits.



Documentación

Documentación

Al instalar **AdminPAQ** se agrega en el grupo de programas **Ayuda**, un archivo con el listado de funciones del **SDK de AdminPAQ** así como información para la utilización de esta librería. Aquí encontrarás la sintaxis de las funciones, los parámetros y los valores que retorna cada una de ellas, definición de estructuras y constantes.



MGW_SDK.h

Las funciones del SDK fueron creadas en C++.

En la ruta **C:\Archivos de programa\Compacw\AdminPAQ\SDK** se encuentra el archivo **MGW_SDK.h** que contiene la declaración de las funciones, estructuras y constantes del SDK definidas en C++.

También puedes apoyarte de este archivo para ver la declaración de las funciones y sus parámetros tal cual como las recibe C++.

Diagrama para crear un desarrollo en SDK

Pasos para el uso del SDK

Pasos generales para crear un desarrollo utilizando el SDK.

1 – Establecer directorio del SDK

Visión general

Introducción

El primer paso para poder usar el SDK de los sistemas comerciales es establecer el directorio donde está ubicado el archivo **MGW_SDK.DLL**.

Este archivo se encuentra por omisión en las siguientes rutas:

Ubicación del MGW_SDK.DLL

El archivo **MGW_SDK.DLL** se instala por omisión en las siguientes carpetas:

Sistema	Ubicación
AdminPAQ	C:\Archivos de Programa\Compacw\AdminPAQ
CONTPAQ i® FACTURA ELECTRÓNICA	C:\Archivos de Programa\Compacw\Facturacion

Registro de Windows

Si instalaste el sistema en otra ubicación podrás saber la carpeta donde se encuentra el archivo **MGW_SDK.DLL** leyendo las siguientes llaves del registro de Windows:

Sistema	Ubicación
AdminPAQ	HKEY_LOCAL_MACHINE\SOFTWARE\Computación en Acción, SA CV\AdminPAQ\DIRECTORIOBASE
CONTPAQ i® FACTURA ELECTRÓNICA	HKEY_LOCAL_MACHINE\SOFTWARE\Computación en Acción, SA CV\ CONTPAQ I Facturacion\DIRECTORIOBASE

Función de Windows

Para establecer este directorio puedes utilizar la función **SetCurrentDirectory** de la librería de Windows **Kernel32.dll**

Declaración de la función

El siguiente código muestra la declaración de la función **SetCurrentDirectory** la cual establece un directorio activo.

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

Continúa en la siguiente página

Visión general, Continuación

Llamado de la función

El siguiente código muestra el llamado a la función **SetCurrentDirectory** que establece el directorio activo almacenado en la llave del sistema **AdminPAQ**. Si deseas acceder a otro sistema asigna la llave correspondiente indicada en la tabla anterior.

```
' Se asigna a una variable la llave del registro de Windows de
AdminPAQ.
string szRegKeySistema = "";

szRegKeySistema = @"SOFTWARE\\Computación en Acción, SA
CV\\AdminPAQ";

RegistryKey keySistema =
Registry.LocalMachine.OpenSubKey(szRegKeySistema);
            object lEntrada = keySistema.GetValue("DirectorioBase");

            // SetCurrentDirectory
            long lResult;
            lResult = SDK.SetCurrentDirectory(lEntrada.ToString());
```

2 - Declaraciones del SDK

Funciones

Qué son

Las funciones del SDK permiten la lectura y escritura en la base de datos de los sistemas. De esta forma podrás enviar información a los sistemas comerciales.

Tipos

En el SDK existen 2 tipos de funciones:

- **Alto nivel.** Usan estructuras
- **Bajo nivel.** No usan estructuras

Lenguajes y tipos de funciones

Solo podrás usar funciones de **alto nivel** en tu proyecto si el lenguaje de programación que estás utilizando soporta el manejo de **estructuras de datos**.

Ejemplos de lenguajes que **sí** soportan el uso de Estructuras de Datos: **Visual Basic for Applications, Visual Basic, C#.**

Ejemplo de un lenguaje que **no** soporta el uso de Estructuras de Datos: **Visual Fox**.

Documentación

Para ver la sintaxis de las funciones y sus parámetros, auxíliate de los archivos **SDK de AdminPAQ.pdf** y **MGW_SDK.h.**

Valores de retorno

En general la mayoría de las funciones del SDK regresan un entero el cual será:

- **0 (cero).** Si la función se ejecutó con éxito.
- **Diferente de cero**. Retorna el número del error.

Parámetros de las funciones

Las funciones pueden recibir parámetros los cuales tienen un tipo y un uso:

- **Tipo**. Enteros, Largos, Cadenas, Booleanos, etcétera. En el caso de las funciones de alto nivel reciben como parámetro una estructura que previamente tuvo que haber sido declarada.
- **Uso**.
    - **Por valor.** Este tipo de declaración se utiliza para los datos que se **envían** a la función.
    - **Por referencia.** Este tipo de declaración se utiliza para **recibir** datos regresados por la función.

Continúa en la siguiente página

Funciones, Continuación

Consejos generales

A continuación se listan los siguientes consejos en la declaración de funciones:

- Siempre ten en cuenta que las funciones del SDK están en C++, el objetivo al declarar las funciones en tu lenguaje es pasar los tipos de datos que C++ pueda recibir. Busca el tipo de datos en tu lenguaje que coincida mejor con el tipo de C++.
- En C++ no existe el tipo **String** sino que todas las cadenas son apuntadores a una cadena de tipo **Char ***. Por lo que, en lenguajes que soporten datos de tipo **String**, se recomienda hacer siempre la declaración del parámetro **Por valor**.
- Si al usar una función recibes un error de tipo **Intento de escribir en memoria protegida** es porque estás utilizando un parámetro con tipo de datos incorrecto o porque no lo estás pasando de forma correcta (Por Valor o Por Referencia).

Parámetros en C# .NET

Enseguida se muestra una tabla con los tipos de datos de los parámetros de funciones para las declaraciones recomendados en **C# .NET:**

SDK de AdminPAQ.pdf		**C# .NET**
Tipo	**Cadena**	**Declaración**
Cadena	Por Valor	string parametro
***Cadena**	**Por Referencia**	**StringBuilder parametro**
Entero	Por Valor	int parametro
Entero	Por Referencia	ref int parámetro
Entero Largo	Por Valor	long parametro
Entero Largo	Por Referencia	ref long parametro
Doble	Por Valor	double parametro
Doble	Por Referencia	ref double parametro
Bool	Por Valor	Boolean parametro
Unsigned char	Por Valor	Byte parametro
Estructuras (Dato abstracto)	Por Referencia	Ref tEstructura parametro

***Importante:** En C++ las cadenas son apuntadores a datos de tipo Char, por lo que ya se reciben por referencia, debido a esto, en C# .NET **siempre** declara todas las cadenas **Por valor** (incluso si en la documentación dice que su uso es Por Referencia).

Si vas a recibir un parámetro en una cadena de texto se recomienda usar el tipo de datos StringBuilder. Los tipos de datos **StringBuilder** requieren de la declaración de la librería "`using System.Text;`".

Continúa en la siguiente página

Funciones, Continuación

Ejemplo C# .NET

A continuación se muestra un ejemplo de declaración de una función en C# .NET, observa cómo están definidas en el archivo PDF y cómo se declara en C# .NET:

- **Documentación en archivo PDF:**

fSiguienteFolio ()

Disponibilidad AdminPAQ 2002.

Sintaxis fSiguienteFolio(*aCodigoConcepto, aSerie, aFolio*)

Parámetros

Nombre	Tipo	Uso	Descripción
aCodigoConcepto	Cadena	Por valor	Código del concepto del documento.
aSerie	Cadena	Por referencia	Serie del documento
aFolio	Doble	Por referencia	Folio del documento

Retorna Valores enteros:
kSIN_ERRORES = 0 (cero) – La operación fue realizada con éxito.
!kSIN_ERRORES = Diferente de 0 (cero) – Código del error.

Declaración en C# .NET:

```
[DllImport("MGW_SDK.dll")]
        public static extern int fSiguienteFolio(string
aCodigoConcepto, string aSerie, ref double aFolio);
```

- **Observa lo siguiente:**
  - El campo **aCodigoConcepto** se define como **String** (Cadena) **Por referencia**.
  - El campo **aSerie** se declara como tipo **String**.
  - El campo **aFolio** se define como **Double** y el uso es **Por Referencia (ref)**.
  - El valor que regresa la función es un **int** (Entero).

Constantes

Constantes

Opcionalmente se sugiere declarar constantes de longitud para los datos de tipo cadena que se utilizarán en las estructuras.

Documentación

Para ver las longitudes de las constantes, auxíliate de los archivos **SDK de AdminPAQ.pdf** (sección **Tipos de dato Abstractos del SDK**) y **MGW_SDK.h.**

Caracter nulo

En la documentación se define el tamaño de las constantes de tipo cadena, es importante sumar 1 caracter al tamaño, correspondiente al caracter nulo.

Declarar constantes

Enseguida se muestra un ejemplo de declaraciones de constantes.

```
// Declaración de constantes
        {
            public const int kLongFecha = 24;
            public const int kLongSerie = 12;
            public const int kLongCodigo = 31;
            public const int kLongNombre = 61;
            public const int kLongReferencia = 21;
            public const int kLongDescripcion = 61;
            public const int kLongCuenta = 101;
            public const int kLongMensaje = 3001;
            public const int kLongNombreProducto = 256;
            public const int kLongAbreviatura = 4;
```

Estructuras

Qué son

Las estructuras definen los datos con sus tipos y longitudes pertenecientes a un objeto.

Ejemplo: La estructura **tDocumento** define los datos que incluye este objeto, como su Folio, Serie, Fecha, Total, etcétera.

Funciones de alto nivel

Las funciones de alto nivel reciben como parámetro una estructura. Para utilizarlas el lenguaje de programación que manejes deberá soportar el uso de estructuras.

Documentación

Para ver la sintaxis de las estructuras, auxíliate de los archivos **SDK de AdminPAQ.pdf** (sección **Tipos de dato Abstractos del SDK**) y **MGW_SDK.h.**

Caracter nulo

En la documentación se define el tamaño de las constantes de tipo cadena, es importante sumar un caracter al tamaño, correspondiente al caracter nulo.

Tipos de datos en C# .NET

Enseguida se muestra una tabla con los tipos de datos en **C# .NET:**

SDK de AdminPAQ.pdf	C# .NET
Tipo	**Declaración**
Cadena	string
Entero	int
Entero Largo	long
Doble	double

Ejemplo C# .NET

A continuación se muestra un ejemplo de declaración de una estructura en C# .NET, observa cómo están definidas en el archivo PDF y cómo se declara en C# .NET:

- **Documentación en archivo PDF:**

Movimientos – RegMovimiento – tMovimiento

Campo	Tipo	Longitud	Descripción
aConsecutivo	Entero	NA	Consecutivo del movimiento.
aUnidades	Doble	NA	Unidades del movimiento.
aPrecio	Doble	NA	Precio del movimiento (para doctos. de venta).
aCosto	Doble	NA	Costo del movimiento (para doctos. de compra).
aCodProdSer	Cadena	kLongCodigo + 1	Código del producto o servicio.
aCodAlmacen	Cadena	kLongCodigo + 1	Código del Almacén.
aReferencia	Cadena	kLongReferencia + 1	Referencia del movimiento.
aCodClasificacion	Cadena	kLongCodigo + 1	Código de la clasificación

Nota: kLongCodigo y kLongReferencia son constantes de longitud definidas previamente.

Continúa en la siguiente página

Estructuras, Continuación

Declarar Estructuras

El siguiente ejemplo muestra la declaración de la estructura para movimientos.

Observa que los datos de tipo **String** tienen definido un tamaño fijo en base a las constantes previamente definidas.

Las constantes de longitud previamente definidas son: kLongCodigo y kLongReferencia.

```
[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;
        }
```

Tamaños de los datos

Es importante que el tamaño de las cadenas sea exacto al que se especifica en la documentación. Para el caso de C# .NET se utiliza el parámetro **SizeConst** para definir el tamaño exacto de la cadena.

3 – Iniciar el SDK

Visión general

Inicializar SDK

Una vez que ya estableciste el directorio donde se ubica el archivo **MGW_SDK.DLL** deberás inicializar el SDK para esto:

Declara y llama la función **fSetNombrePAQ(aSistema)** dependiendo el sistema:

Sistema a usar	Sistema
AdminPAQ	fSetNombrePAQ(“AdminPAQ”)
CONTPAQ i® FACTURA ELECTRÓNICA	fSetNombrePAQ(“CONTPAQ I Facturacion”)

Nota: Para el caso exclusivo de **AdminPAQ** también se puede inicializar el SDK a través de la función **fInicializaSDK()**. Pero es importante que uses solo una de las dos funciones (fInicializaSDK o fSetNombrePAQ) para que no inicialices el SDK 2 veces.

Recomendación: Se recomienda inicializar solo una vez el SDK por proyecto.

Declarar fSetNombrePAQ

El siguiente código declara la función **fSetNombrePAQ** la cual se utiliza para inicializar el SDK.

```
// Función para inicializar el SDK
[DllImport("MGW_SDK.DLL")]
        public static extern int fSetNombrePAQ(String aNombrePAQ);
```

Inicializar SDK

El siguiente ejemplo inicializa el SDK en **CONTPAQ i® FACTURA ELECTRÓNICA.** Si deseas inicializar el SDK de **AdminPAQ**, cambia el parámetro por “AdminPAQ”.

```
// Función para inicializar el SDK en CONTPAQ i FACTURA ELECTRÓNICA
fSetNombrePAQ("CONTPAQ I Facturacion");
```

4 – Abrir la empresa

Visión general

Abrir empresa

Después de inicializar el SDK deberás abrir la empresa con la función **fAbreEmpresa**.

Declarar fAbreEmpresa

El siguiente código declara la función para abrir empresas**.**

```
// Función para Abrir la empresa
[DllImport("mgw_sdk.dll")]
        public static extern int fAbreEmpresa(string Directorio);
```

Abrir Empresa

El siguiente código abre la empresa ubicada en la carpeta “MiEmpresa”**.**

```
// Abre la empresa
fAbreEmpresa("C:\\CompacW\\Empresas\\MiEmpresa");
```

5 – Realizar las operaciones deseadas

Visión general

Introducción

Después de abrir la empresa podrás realizar las operaciones deseadas como crear clientes, productos, documentos, etcétera.

Operaciones comunes

En la siguiente tabla se muestran las funciones del SDK requeridas para realizar las operaciones más comunes:

Operación	Funciones del SDK
Crear producto	• fAltaProducto
Editar producto	• fBuscaIdProducto • fEditaProducto • fSetDatoProducto • fGuardaProducto
Crear clientes	• fAltaCteProv
Editar clientes	• fBuscaIdCteProv • fEditaCteProv • fSetDatoCteProv • fGuardaCteProv
Crear documentos	• fSiguienteFolio • fAltaDocumento • fAltaMovimiento • fAfectaDocto_Param
Crear documentos con series o lotes	• fSiguienteFolio • fAltaDocumento • fAltaMovimiento • fAltaMovimientoSeriesCapas • fCalculaMovtoSerieCapa • fAfectaDocto_Param
Timbrar y entregar documentos	• fInicializaLicenseInfo • fEmitirDocumento • fEntregEnDiscoXML
Timbrar XML	• fInicializaLicenseInfo • fTimbraXML

6 – Cerrar la empresa

Visión general

Cerrar empresa

Antes de salir de tu aplicación cierra la empresa con la función **fCierraEmpresa**.

Declarar fCierraEmpresa

El siguiente código declara la función para cerrar empresas**.**

```
// Función para cerrar la empresa
[DllImport("mgw_sdk.dll")]
        public static extern void fCierraEmpresa();
```

Cierra Empresa

El siguiente código hace el llamado a la función para cerrar empresas**.**

```
// Función para cerrar la empresa
fCierraEmpresa();
```

7 – Terminar SDK

Visión general

Terminar SDK

Al terminar de utilizar el SDK deberás usar la función **fTerminaSDK** para finalizar la conexión a este.

Declarar fTerminaSDK

El siguiente código declara la función para terminar la sesión del SDK**.**

```
// Función para terminar el SDK
[DllImport("mgw_sdk.dll")]
        public static extern void fTerminaSDK();
```

Llamado a la función fTerminaSDK

El siguiente código llama la función para terminar la sesión del SDK**.**

```
// Función para terminar el SDK
fTerminaSDK();
```