El programa de facturación

Según se indica en la RG AFIP Nº 259/98, el software de gestión para puntos de venta deberá:

- No permitir la impresión de documentos similares a los Documentos Fiscales (*DF*). Sólo posibilitará la emisión de los Documentos No Fiscales (*DNF*) cuyos usos se indican en la resolución general mencionada más arriba, y de los Documentos No Fiscales Homologados (*DNFH*) disponibles en la impresora fiscal homologada.

- No emitir *DF* mediante otras impresoras del tipo no homologadas.

- Asegurar que, en forma concomitante con la captura de la información referente a cada ítem vendido o servicio prestado, se impriman los correspondientes datos en el comprobante a emitir, excepto cuando por la modalidad operativa de la actividad desarrollada por el usuario de la impresora fiscal, se requiera la facturación diferida con relación a la mencionada captura (gastronomía, hotelería, etc.).

- Asegurar la comunicación a través del protocolo de comunicaciones en forma bidireccional.

- Controlar **todos los errores** que reporta el controlador fiscal.

- Informar al usuario del software de gestión para puntos de venta acerca de los errores que impidan continuar con la facturación.

En la misma resolución general se aclara:

"*Los programas que se utilicen en las computadoras personales u otros equipamientos, para la emisión de comprobantes, deberán estar adaptados completamente a las especificaciones de funcionamiento de las impresoras fiscales*".

A las exigencias anteriores, *Compañía HASAR SAIC* agrega:

"*Es altamente recomendable que el software de gestión para puntos de venta habilite la creación y registración, en un archivo de texto, del diálogo mantenido entre dicho software y la impresora fiscal HASAR*".

El archivo indicado anteriormente, generalmente denominado "LOG del sistema", debería contener como mínimo los strings intercambiados (comandos y respuestas). Además de facilitar las tareas de depuración durante la etapa de desarrollo permite, al entrar el software en producción, determinar rápidamente si el impedimento de continuar con la emisión de comprobantes es un problema de diálogo, de la impresora fiscal HASAR, o del propio software de gestión para puntos de venta.

Las herramientas para desarrollo provistas por *Compañía HASAR SAIC*, que se incluyen en el ZIP de drivers fiscales y que se pueden descargar –gratuitamente- del sitio web oficial de *Grupo HASAR*, permiten la creación y registración indicada en los párrafos anteriores.

El desarrollador del software de gestión para puntos de venta deberá consultar la documentación correspondiente a la herramienta que utilice en su producto para interiorizarse acerca del mecanismo de habilitación del "LOG" sugerido.

El control de la respuesta a un comando enviado

Cada vez que el software de gestión para puntos de venta envía un string de comando a la impresora fiscal HASAR, ésta responde con un string cuyo formato es similar al siguiente:

…	*Estado Impresora*	FS	*Estado Fiscal*	… Opcionalmente más información …

Estado Impresora	Primer campo de información. Existe siempre; incluso si el comando fue rechazado. Contiene cuatro caracteres ASCII que representan dígitos hexadecimales. Por ejemplo, "C080".
FS	Caracter ASCII 28 decimal. Separador de campos de información.
Estado Fiscal	Segundo campo de información. Existe siempre; incluso si el comando fue rechazado. Contiene cuatro caracteres ASCII que representan dígitos hexadecimales. Por ejemplo, "B620".

Del tercer campo en adelante la impresora fiscal HASAR entrega, de corresponder, información asociada al comando enviado, sea éste de consulta, o no. Por ejemplo, el tercer campo de la respuesta al comando de cierre de documento fiscal contiene el número del comprobante que se ha cerrado.

Analizando estados

Para proceder al análisis de la información de estados el software de gestión para puntos de venta debe separar en campos el string de respuesta, convertir a representación binaria la notación hexadecimal reportada y examinar los bits de ambos campos.

El análisis de la respuesta debe ser realizado teniendo en cuenta el contexto en el cual el comando es enviado a la impresora fiscal. Por ejemplo:

+ Al arrancar el programa la intención es cancelar cualquier comprobante que pudiese estar abierto en la impresora fiscal HASAR. Si la respuesta al comando de cancelación es "Rechazado por inválido", se interpreta como la inexistencia de un comprobante abierto.

+ Si habiendo un documento fiscal abierto en la impresora fiscal HASAR el programa envía el comando de cancelación y es "Rechazado por inválido", en este caso la interpretación es que existe un pago parcial registrado y el comprobante debe cerrarse para luego emitir una nota de crédito.

Rompiendo mitos

Es una creencia muy arraigada, no sólo entre desarrolladores de software de gestión para puntos de venta, sino también entre usuarios de sus productos, que se puede cambiar de modelo y/o versión de impresora fiscal HASAR en el punto de venta sin que se requieran ajustes en el programa de gestión.

Es cierto que al comparar dos sets de comandos, soportados por sendas impresoras fiscales HASAR, puede haber un altísimo grado de compatibilidad. Lo que no es posible garantizar es la compatibilidad al 100%. Tener en cuenta longitudes de textos, prestaciones disponibles, cambios en la normativa fiscal vigente, etc..

Debido a lo anterior, es muy importante el control de errores mediante el análisis de las respuestas a los comandos enviados. Ante la prueba de fuerza bruta de cambiar el modelo de impresora fiscal HASAR en el punto de venta es vital atrapar y tratar adecuadamente rechazos de comandos por: comando inválido, campo de datos inválido, desborde de total, comando desconocido, etc..

Algunas ideas

Ya se ha dicho que el string de respuesta comienza con los siguientes caracteres: cuatro de estado impresora, uno como separador de campos, y cuatro como estado fiscal.
Si se consulta el manual de comandos de la impresora fiscal HASAR, se verá que estas posiciones son las únicas fijas en un string de respuesta cualquiera. El resto de los campos, o no existen, o son de longitud variable.

Para el esbozo de las ideas se empleará el lenguaje Visual Basic debido a su popularidad y sencillez de escritura e interpretación.

Estado Impresora				FS	*Estado Fiscal*				*Resto del string de respuesta*
1	2	3	4	5	6	7	8	9	*n* posiciones más
P	**P**	**P**	**P**	∟	**F**	**F**	**F**	**F**	…

∟ = *ASCII 28 decimal ; Separador de campos*

Suponiendo que el string de respuesta a un comando enviado a la impresora fiscal HASAR se encuentra almacenado en una variable llamada “Respuesta”, y suponiendo también que el contenido de la misma es “C080∟3600” (“C080” representado por “PPPP” en la tabla anterior, y “3600” representado por “FFFF” en la misma tabla), para obtener cada campo de estado se puede recurrir al uso de la función *Left$( )* -para los primeros cuatro caracteres (estado impresora)-, y el uso de la función *Mid$( )* -para los cuatro caracteres que siguen (estado fiscal) al separador de campo ubicado en la quinta posición-.

Por ejemplo,

```
Dim PrinterStatus as String
Dim FiscalStatus as String

PrinterStatus = Left$(Respuesta, 4)      '// Se obtiene el string “C080”
FiscalStatus = Mid$(Respuesta, 5, 4)     '// Se obtiene el string “3600”
```

Si se fuese necesario obtener información contenida en campos de longitud variable (como podría ser cualquier otro campo adicional de la respuesta), se puede recurrir al uso de la función *InStr( )* para buscar la posición del siguiente separador de campos.

Por ejemplo,

```
Dim n as Integer, m as Integer
Dim UnCampo as String,  SigCampo as String
Dim FS As String

FS = Chr(28)
n = InStr( 11, Respuesta, FS )          '// 11 es la posición de comienzo del tercer campo, si existiese
m = InStr( n + 1, Respuesta, FS )

UnCampo = Left$(Respuesta, n - 1 )
SigCampo = Mid$( Respuesta, n + 1, m - n - 1 )
```

Volviendo al análisis de los campos de estado…
El siguiente paso a realizar (con cada campo de estado) es convertir estos strings ASCII que representan dígitos hexadecimales a valores numéricos enteros (2 bytes, 16 bits). Asumiendo que las variables "PrinterStatus" y "FiscalStatus" contienen la información a convertir, se puede recurrir al uso de la función *Val( )* para concretar la conversión.

```
Dim Printer as Integer
Dim Fiscal as Integer
Printer = Val( "&H" + PrinterStatus )      '// "C080" es 49280 como valor entero
Fiscal = Val( "&H" + FiscalStatus )        '// "3600" es 13824 como valor entero
```

El argumento "&H" fuerza a la función *Val( )* a considerar el string como representación hexadecimal. Ahora solo resta analizar –en formato binario- cada uno de los bits de los valores enteros obtenidos. En el manual de comandos correspondiente a cada modelo de impresora fiscal HASAR, en los apéndices para tal fin, se describe como interpretar cada uno de estos bits.

Suponiendo que el software de gestión para puntos de venta debiese atrapar el bit que es puesto a uno, en el campo de estado impresora, cuando en la respuesta se reporta error mecánico de impresora, se puede proceder del siguiente modo (recordar que la variable "Printer" contiene el valor 49280):

```
Const ERRORPRINTER = 4                                  '// 4 = 0100 en binario

if (( Printer And ERRORPRINTER ) = ERRORPRINTER ) then  '// ¿ (49280 And 4) = 4 ?
      Print "Error mecánico de impresora"
end if
```

Tener presente para el análisis que el bit cero –valor menos significativo- es el que se encuentra más a la derecha en el valor binario. Para el caso de '0100' se encuentra en '1' el bit '2' (tercera posición de derecha a izquierda).

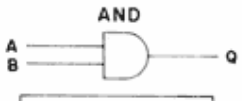

TABLA DE VERDAD
AND

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

La imagen de la izquierda muestra como se comporta el operador AND a nivel de bits (da como resultado '1' si y solo si ambos bits comparados valen '1').

Por ejemplo,

C080 1100000010000**0**00
AND
0004 0000000000000**1**00
=
0004 0000000000000**0**00

Este tipo de operaciones con máscaras pueden ser combinadas, mediante el operador OR. Por ejemplo,

```
Const FISCALMEMFAIL = &H1          '// Bit '0' en '1'
Const WORKINGMEMFAIL = &H2         '// Bit '1' en '1'
Const FISCALMEMFULL = &H80         '// Bit '7' en '1'

Dim ErrorCritico AS Integer

ErrorCritico = FISCALMEMFULL OR WORKINGMEMFAIL OR FISCALMEMFAIL

if (( Fiscal And ErrorCritico ) <> 0 ) Then      '// Recordar que "Fiscal" contiene el valor 13824
      Print "Hay un error fiscal crítico"
end if
```

Una mirada alternativa

Una variante del ejemplo expuesto hasta ahora, es no preguntar por el valor de un bit determinado, sino recorrer los 16 bits de cada campo de estado.

Suponiendo que en la variable entera "FiscalStatus" se encuentra el estado fiscal devuelto por la impresora fiscal HASAR (13824 en nuestro caso), entonces:

```
Dim Mask As Long                        '// Para poder superar la frontera del entero con signo (32767)
Dim i As Integer
Dim FiscalStatusError (16) as String    '// Puede variar según el modelo de impresora fiscal HASAR

FiscalStatusError (1)  = "Fiscal Memory Fail"
FiscalStatusError (2)  = "Working Memory Fail"
FiscalStatusError (3)  = ""
FiscalStatusError (4)  = "Unrecognized Command"
FiscalStatusError (5)  = "Invalid Field Data"
```

```
FiscalStatusError (6)  = "Invalid Command"
FiscalStatusError (7)  = "Total Overflow"
FiscalStatusError (8)  = "Fiscal Memory Full"
FiscalStatusError (9)  = "Fiscal Memory Near Full"
FiscalStatusError (10) = "Fiscal Terminal Certified"
FiscalStatusError (11) = "Fiscal Terminal Fiscalized"
FiscalStatusError (12) = "Date Set Fail"
FiscalStatusError (13) = "Receipt/Slip Open"
FiscalStatusError (14) = "Slip Open"
FiscalStatusError (15) = "Receipt Open"
FiscalStatusError (16) = ""

Mask = 1

For i = 1 To 16
      if (( FiscalStatus And Mask ) <> 0 ) then
            Print FiscalStatusError( i )
      end if
      Mask = Mask * 2
Next i
```

1	0001
2	0010
4	0100
8	1000
…	…

Multiplicar por 2, equivale a desplazar una posición a la izquierda el bit en '1' por cada operación. Notar que el bit en '1' se encuentra , al realizar la primera multiplicación en la posición '0'.

La tabla de la izquierda muestra en su primera fila el valor original de la variable "Mask". Cada nueva fila muestra el resultado de "Mask * 2".

El OCX Fiscal HASAR

Para aquellos desarrolladores de software de gestión para puntos de venta que en sus productos utilicen el OCX Fiscal HASAR (en cualquiera de sus versiones), el análisis de simplica bastante. Cada vez que se invoca a un método miembro del OCX Fiscal HASAR, con la información suministrada se genera internamente el string de comando a enviar.

Cuando el software de gestión para puntos de venta recibe el string de respuesta al comando enviado, el análisis de los bits en los campos de estado (impresora y fiscal) es realizado por el propio OCX Fiscal HASAR.

Por cada bit en '1' detectado en los campos de estado, el OCX Fiscal HASAR dispara el evento asociado que corresponda, reportando un código de evento perfectamente identificable. El software de gestión para puntos de venta deberá atrapar los eventos que pueden ser disparados por el OCX Fiscal HASAR, identificando el código de evento, y realizando un adecuado tratamiento de la situación presentada.

Supongamos que se intenta emitir una factura 'A', para lo cual es necesario enviar a la impresora fiscal HASAR el comando *SetCustomerData* para informar los datos del comprador –requisito obligatorio para poder abrir un comprobante del tipo de nuestro interés-. El comando en cuestión se genera en el OCX Fiscal HASAR invocando al método miembro llamado *DatosCliente( )*. Ahora supongamos, también, que el comando enviado es rechazado por la impresora fiscal HASAR por "campo de datos inválido".

Para el caso expuesto en el párrafo anterior, supondremos que el campo de estado fiscal contenido en la respuesta al comando *SetCustomerData –DatosCliente( )*- reporta la siguiente información: "8610".

El OCX Fiscal HASAR hace un análisis del campo de estado fiscal que determina:

Valor hexadecimal:	8 6 1 0		
Valor binario:	1000 0110 0001 0000	→	Bit '4' en '1'
Valor entero:	34336		Campo de datos inválido

En consecuencia, el OCX Fiscal HASAR dispara el evento *ErrorFiscal( )* reportando el rechazo del comando mediante el código de evento *F_INVALID_FIELD_DATA* (valor 16 decimal).

Una rutina como la siguiente (en su más paupérrima expresión) podría formar parte del código de escrito para el software de gestión para puntos de venta.

```
Private Sub HASAR1_ErrorFiscal(ByVal Flags As Long)

    If (Flags = F_INVALID_FIELD_DATA) Then
        MsgBox "Campo de datos inválido"
    End If

End Sub
```

De la forma expuesta en el ejemplo anterior (por lo menos en Visual Basic) es posible atrapar uno de los eventos que puede disparar el OCX Fiscal HASAR.