MANUAL PARA USO DEL GRID CON GML

INDICE

Visual PRO/5 Grid Management Library **403**
GML Grid Management Template: GML$ **403**
Atributos GML y Celdas de datos (texto). Plantilla GML_TPL$ **404**
Variables que afectan los atributos de cada grid celda. **405**
Variables que afectan la edicíon del grid **405**
Uso de Variables Misceláneas **405**
GML_TPL$: Inicialización de una plantilla **406**
Manejo de librerías grid: Nombre de Variables/Llamados de Programas GML **406**
GML Nombre de Variables/Llamados a programas GML **407**
GML: Pre-asignación de Valores **407**
GML Start-Up: Llamando programa GML_I **408**
Ejecutando un Procedimiento GML **409**
Procedimiento GML: Index **10**
Procedimiento GML TPL_PREP **11**
Procedimiento GML CLEAR **11**
Procedimiento GML CURRENT **12**
Procedimiento GML DELETE_ROW **12**
Procedimiento GML DESELECT **13**
Procedimiento GML DISABLE **13**
Procedimiento GML ENABLE **14**
Procedimiento GML END_EDIT **14**
Procedimiento GML END_EDIT **15**
Procedimiento GML FETCH **16**
Procedimiento GML FIND **17**
Procedimiento GML FOCUS **17**
Procedimiento GML GET **17**
Procedimiento GML GET_STORED_ROW_DATA **18**
Procedimiento GML GOTO **419**
Procedimiento GML INSERT_ROW **20**
Procedimiento GML LOAD **20**
Procedimiento GML MODE **21**
Procedimiento GML NEXT **22**
Procedimiento GML POPULATE **23**
Procedimiento GML PUP **23**
Procedimiento GML REFRESH **24**
Procedimiento GML RESTORE **25**
Procedimiento GML SERIES **26**
Procedimiento GML SET_COLUMN_WITDH **27**
Procedimiento GML SET_DEFAULT **27**
Procedimiento GML SET_EDIT_MASK **28**
Procedimiento GML SET_HIGHLIGHT_METHOD **28**
Procedimiento GML SET_IMAGELIST **429**

Procedimiento GML SET_LINE_MODE 30
Procedimiento GML SET_OUT_MASK 30
Procedimiento GML SET_ROWS 31
Procedimiento GML SET_ROW_HEIGHT 32
Procedimiento GML SHOW..... 33
Procedimiento GML SORT 33
Procedimiento GML START..... 34
Procedimiento GML START_EDIT..... 36
Procedimiento GML STORE_ROW_DATA..... 36
Procedimiento GML UPDATE 37
Procedimiento GML UPDATE 38
Interpretando La notificación de eventos : GML:MOTIFY%..... 439
Interpretando Eventos grid notify..... 41
Manejo de Librerías Grid: Manejo productos Look-up (LUM)..... 42
Grid Management Library : Look-Up Manager (LUM) 43
Información Adicional 43
GML Look-Up Manager (LUM) : Start-Up..... 44
GML Look-Up Manager (LUM) : Desplegando el The Look-Up Control..... 45
GML Look-Up Manager (LUM) : Respondiendo a eventos 46
Listando la subrutina LUM_LOOK_UP 48
LUM_EVENT Subroutine:..... 449
GML LUM PROCEDIMIENTO: Index 449
GML LUM PROCEDIMIENTO INIT_LUM_TPL..... 50
GML LUM PROCEDIMIENTO LUM_DISPLAY 50
GML LUM PROCEDIMIENTO LUM_EVENT 51
GML LUM PROCEDIMIENTO LUM_FIND..... 52
GML LUM PROCEDIMIENTO LUM_GET_TEXT 53
GML LUM PROCEDIMIENTO LUM_HIDE..... 53
GML LUM PROCEDIMIENTO LUM_MODE..... 53
GML LUM PROCEDIMIENTO LUM_START 54
Asociando un archivo con un grid control 55
Características Modo de archivo Attached 56
Ejemplo de programa que captura datos sin estarAttached 57

Visual PRO/5 Grid Management Library

EL GML provee un método compresivo de control de manejo del Grid en ambiente VPRO/5. EL manejo se refiere al proceso de actualización de atributos del GRID e información de texto mientras administra eventos en el Grid.

EL GML contiene una librería de programas y rutinas asociadas que permiten un desarrollo para crear códigos de interface con Grid, a través del uso de un comando como, un set de instrucciones referentes a un procedimiento. Cuando el procedimiento es ejecutado dentro de programas GML ellos son expandidos de manera similar a ejecutar instrucciones macro.

Una plantilla Grid es usada para la interface entre el Grid y el desarrollo del código. Cuando se ejecutan procedimientos GML las plantillas GML son usadas para la interface con el grid en una celda base, de la celda mientras permite el desarrollo para trabajar con el Grid en una fila por base de la fila, con información de columna (celda) contenida dentro de la plantilla.

En efecto el manejo de librerías Grid crea una capa entre las instrucciones codificadas por un desarrollador y las instrucciones actualmente requeridas para complementar la tarea seleccionada.

En VPRO5 los grid pueden contener arriba de tres controles. EL grupo de GML los tres controles dentro del set de grid con el control individual se refiere a miembros. Un set siempre contiene un grid Principal (miembro uno) y puede también contener un encabezado de columna (miembro 2) y un encabezado de fila (miembro tres).

EL GML consiste en tres programas públicos (llamados) con funciones específicas como las siguientes:

GML_I: es llamado durante el startup para inicializar las variables GML.

GML_M: Administra la notificación de eventos asociados con objetos tipo 107 (grid control).

Los programas GML proveen una interface entre el desarrollador y el control grid a través del uso de plantillas GML y procedimientos GML asociados.

GML Grid Management Template: GML$

El manejo de información Grid es mantenido por el GML en la plantilla de manejo de grid GML$. Esta plantilla contiene los valores necesarios para proveer el continuo manejo de grid. Los campos de la plantilla que son aplicables para la interface con un desarrollador.

Ejemplo:

GML.SET%: Se refiere al grid presente a ser manejado por el GML.

Se refiere al valor relacionado al estatus de un evento particular.

GML.NOTIFY%: Contiene un valor que se relaciones a un evento grid particular.

Los valores de los campos dentro de la plantilla GML$ son mantenidos por el GML y son solo de tipo lectura, como comparación a la plantilla GML_TPL$ la cual es usada por el desarrollador a través de procedimientos para una interface con el grid para obtener o cambiar los atributos del grid una celda texto.

Atributos GML y Celdas de datos (texto). Plantilla GML_TPL$

La plantilla GML_TLP$ contiene parámetros necesarios para la interface con los controles grid. Cuando un procedimiento GML es ejecutado, los campos dentro de la plantilla contienen los valores que son apropiados para la tarea que será realizada. No todos los campos dentro de la plantilla son usados para cada procedimiento. Las variables contenidas en la plantilla son descritas en los siguientes grupos.

Variables referenciado set de grids, filas y posiciones de columnas:

GML_TPL.SET% se refiere a un grid, set de procedimientos relacionados, con un valor de 1 al máximo set de grids que sean manejados por el GML.

GML_TPL.MEMBER% se refiere al grid específico de control dentro de un set de grid. Un valor de uno indica un grid control principal, un dos es un encabezado de columna y un tres es un encabezado de fila.

GML_TPL.ROW_N% se refiere e un número de fila en un rango de uno a un máximo de filas en un grid.

GML_TPL.COL_N% se refiere a un número de columna de un rango de un máximo de columnas en un grid.

Un GML_TPL.ROW_N% o GML_TPL.COL_N% valor de menos de uno tiene un especial significado explicado en la sección procedimiento GML update de este manual.

Variables contenidas en grid data (texto).
GML_TPL.COL$[c] contienen el grid, celdas texto con el valor de <c> indicando el número de columna. El número de rango desde uno a la máxima columna en el grid.

GML_TPL.REL$[c] contiene texto el cual es relacionado al grid celda texto. Como ejemplo un grid celda contiene un número de cuenta que podría usar la posición GML_TPL.REL$[c]
Para almacenar la actual llave de registro en las variables GML.

Variables que afectan los atributos de cada grid celda.

GML_TPL.T_COLOR$[c] se refiere al color de texto de la columna.

GML_TPL.B_COLOR$[c] se refiere al color de fondo de la columna.

GML_TPL.STYLE%[c] se refiere al estilo de la celda (entrada, botón, etc.).

GML_TPL.ALIGN%[c] se refiere a la alineación de la celda (derecha, izquierda, centrada).

GML_TPL.IMAGE%[c] se refiere al número de índice de una imagen desplegada dentro de la celda.

GML_TPL.DEFAULT$[c] contiene los atributos por defecto de la columna.

Variables que afectan la edición del grid

GML_TPL.E_MODE%[c] permite o restringe la edición del grid principal.

GML_TPL.MASK$[c] se refiere a la máscara de la columna.

Uso de Variables Misceláneas

GML_TPL.OPTION se refiere al uso de una opción particular para un procedimiento específico.
GML_TPL.DEF_FLAG% se refiere al uso de un producto particular para un procedimiento específico.

GML_TPL.FLAG% contiene un valor relacionado a la ejecución de un procedimiento.

GML_TPL.TEXT$ contiene texto usado para múltiples propósitos.

GML_TPL.ARG% variable numérica usada con ciertos propósitos.

GML_TPL.SUP% variable numérica usada con ciertos propósitos.

GML_TPL.INFO$ contiene información relacionada a ciertos procedimientos.

GML_TPL.ROW_STAT% contiene información del estatus de la fila.

GML_TPL$: Inicialización de una plantilla

Cada vez que el GML_TPL$ es usado debe ser inicializado. Esta inicialización prepara la plantilla para ser usada y capturar los parámetros de la plantilla a ser especificados en la interface del grid. Inicialización también fija variables pre-establecidas las cuales son necesarias para la correcta realización de un procedimiento GML. El procedimiento usado para la inicialización es perfilado en GML.

Manejo de librerías grid: Nombre de Variables/Llamados de Programas GML

Todas las variables GML empiezan con la letra GML. Las variables asociadas con el llamado dos de los tres programas GML son como los siguientes y son usados como parte de la lista de argumentos del llamado de un programa.

GML_INIT$: Inicialización de parámetros.

GML$ Plantilla maestra de grid.

GML_TPL$ Atributos y datos de plantilla de texto.

GML_GM$ GML celda de datos principal del array.

GML_GC$ GML encabezado de columna en array.

GML_GR$ Encabezado de fila en array.

GML_SET$ Variable string

El programa GML_I es llamado por el programa de espacio de trabajo una vez durante un programa de startup como se indica adelante:

```
CALL
"GML_I",SYSGUI,GML_INIT$,GML$,GML_GM$[ALL],GML_GC$[ALL],GML_GR$[ALL],GML_SET$
```

SYSGUI es el número de canal de la venta SYSGUI

El programa GML_M es llamado por el programa de espacio de trabajo cuando el notificador de eventos asociado con el objeto tipo 107 ocurre como en el siguiente ejemplo.

```
IF EVENT.OBJTYPE%=107 THEN IF EVENT.CODE$="N" THEN GOSUB GML_M

GML_M:
CALL "GML_M",SYSGUI,EVENT$,NOTICE$,GML_SET$,GML_TPL$,GML$,
            GML_GM$[ALL],GML_GC$[ALL],GML_GR$[ALL]
```

EVENT$ contiene el valor del evento recibido desde la cola de la plantilla

NOTICE$ contiene el valor recibido desde la plantilla grid de notificación.

El programa GML provee de interfaces al grid control a través del uso de GML plantillas y procedimientos asociados. El procedimiento GML ha sido creado de manera que minimice la cantidad de código que un desarrollador requiere para escribir cuando llama un programa GML. El nombre del procedimiento es además la etiqueta reverenciada en el GML (llamado), como aparece en el siguiente ejemplo:

Cargue la variable **GML.SET%** con el apropiado set número grid.

GML.SET%=1 Inicialice la plantilla **GML_TPL$ CALL "GML::TPL_PREP"**

GML Nombre de Variables/Llamados a programas GML

Desde el llamado el GML no usa lista de argumentos, el programa GML comparte todas las variables con el llamado de programa. Por esta razón el GML solo contiene variables que empiezan con las letras GML. Por eso las variables que empiezan con GML deben ser consideradas como reservadas para ser usadas por el GML y consecuentemente no asignadas por los desarrollados para uso en el código de aplicación.

Si el programa GML es llamado desde el programa, ese programa se debe haber llamado con las siguientes variables como parte de la lista de argumentos:

GML_TPL$,GML$,GML_GM$[ALL],GML_GC$[ALL],GML_GR$[ALL]

Ejemplo:

```
REM MAIN_PROGRAM
.
CALL "PROGRAM_1",A$,B$,GML_TPL$,GML$,GML_GM$[ALL],GML_GC$[ALL],GML_GR$[ALL]

REM PROGRAM_1
.
ENTER A$,B$,GML_TPL$,GML$,GML_GM$[ALL],GML_GC$[ALL],GML_GR$[ALL]
.
.
GML.SET%=1
CALL  "GML::TPL_PREP"
```

GML: Pre-asignación de Valores

Cuando el programa GML_I es llamado durante el startup del programa, la plantilla GML$ es cargada con valores que pueden ser usados con ciertos procedimientos que se relacionan para cambiar atributos de las celdas. Los nombres de campos son:

GML.BLACK$	**GML.BLUE$**	**GML.CYAN$**
GML.DKGRAY$	**GML.GRAY$**	**GML.GREEN$**
GML.LTGRAY$	**GML.MAGENTA$**	**GML.RED$**
GML.WHITE$	**GML.YELLOW$**	
GML.CENTERED%	**GML.LEFT%**	**GML.RIGHT%**
GML.CHECKED_CHECKBOX%		**GML.INPUTE%**
GML.RAISED_BUTTON%		**GML.RECESSED_BUTTON%**
GML.UNCHECKED_CHECKBOX%		

GML Start-Up: Llamando programa GML_I

Si un valor o un problema es encontrado durante el startup cuando está llamado el GML_I, la variable GML.FLAG% será asignada a valor negativo que le corresponde a uno de los siguientes:

Valor	Significado.
-401	**GML_INIT$** la plantilla no ha sido inicializada.
-402	Valor inválido en la variable **GML_INIT.GR_TOT%**
-403	Valor inválido en la variable **GML_INIT.LUM_QTY%**
-404	problema obteniendo la cuenta grid columna principal
-405	problema obteniendo la cuenta grid fila principal
-406	problema obteniendo la cuenta grid visual
-407	problema deseleccionando el grid principal.
-408	reservado para uso futuro
-409	MOD(grid principalID,5) no igual a 1
-410	ID encabezado de columna no igual al grid principal ID +1
-411	ID encabezado de fila no igual al grid principal ID +2
-412	celdas máximas excedidas
-413	problema obteniendo las dimensiones del grid principal
-414	problemas obteniendo las dimensiones del encabezado de columna
-415	problemas obteniendo las dimensiones del encabezado de fila
-416	asociación inválida entre el grid control principal y el control de encabezado de columna
-417	asociación inválida entre el grid control principal y el control de encabezado de fila

El valor **GML.FLAG%** podría ser chequeado por la aplicación inmediatamente seguido del llamado al programa GML_I

Ejecutando un Procedimiento GML

Un procedimiento es siempre ejecutado por el llamado de un programa GML. Si un valor inválido o un problema es encontrado durante la ejecución de un procedimiento o ir un programa GML, la variable GML.FLAG% será asignada a un valor negativo de la siguiente lista:

Valor	Significado
-1	Valor inválido en variable **GML.SET%**
-2	Inválido número de fila en variable **GML_TPL.ROW_N%**
-3	operación no exitosa.
-4	Plantilla **GML_TPL$** no inicializada
-5	**GML_TPL.MEMBER%** contiene un valor inválido.
-6	Procedimiento no válido para valor en **GML_TPL.MEMBER%**
-7	Incorrecta inicialización de plantilla para el valor contenido en **GML.SET%**
-8	Valor inválido en variable **GML_TPL.OPTION%**
-9	procedimiento **START_EDIT** ejecutado en celda no editable.
-10	inválido número de columna en variable **GML_TPL.COL_N%**
-11	valor inválido en variable **GML_TPL.DEF_FLAG%**
-12	procedimiento **START_EDIT** ejecutado en grid de solo lectura
-13	Procedimiento **RESTORE** ejecutado en fila incorrecta.
-14	(reservado)
-15	(reservado)
-16	(reservado)
-17	Procedimiento **RESTORE** ejecutado con un valor incorrecto en valor G**ML_TPL.DEF_FLAG%**
-18	Problema encontrado cuando ejecuta el procedimiento **SORT** variable (**GML_TPL.FLAG%**) contiene el VPRO5 número de y **GML_TPL.ROW_N%** el número de fila que causa el error.
-19	Procedimiento **SORT** ejecutado en una celda mientras estaba en modo edit
-20	valor inválido en variable **GML_TPL.ROW_N%**
-21	valor inválido en variable **GML_TPL.COL_N%**
-22	valor inválido en variable **GML_TPL.COL$[GML_TPL.COL_N%]**
-23	valor inválido en variable **GML_TPL.ARG%**
-24	valor inválido en variable **GML_TPL.SUP%**

Procedimiento GML: Index

PAG.	NOMBRE PROCEDIMIENTO
411	CALL "GML::CLEAR"
412	CALL "GML::CURRENT"
412	CALL "GML::DELETE_ROW"
413	CALL "GML::DESELECT"
413	CALL "GML::DISABLE"
414	CALL "GML::ENABLE"
414	CALL "GML::END_EDIT"
416	CALL "GML::FETCH"
416	CALL "GML::FIND"
417	CALL "GML::FOCUS"
417	CALL "GML::GET"
418	CALL "GML::GET_STORED_ROW_DATA"
419	CALL "GML::GOTO"
419	CALL "GML::HIDE"
420	CALL "GML::INSERT_ROW"
420	CALL "GML::LOAD"
421	CALL "GML::MODE"
422	CALL "GML::NEXT"
423	CALL "GML::POPULATE"
423	CALL "GML::PUT"
424	CALL "GML::REFRESH"
425	CALL "GML::RESET"
425	CALL "GML::RESTORE"
426	CALL "GML::SERIES"
427	CALL "GML::SET_COLUMN_WIDTH"
427	CALL "GML::SET_DEFAULT"
428	CALL "GML::SET_EDIT_MASK"
428	CALL "GML::SET_HIGHLIGHT_METHOD"
429	CALL "GML::SET_IMAGELIST"
430	CALL "GML::SET_LINE_MODE"
430	CALL "GML::SET_OUT_MASK"
431	CALL "GML::SET_ROWS"
432	CALL "GML::SET_ROW_HEIGHT"
432	CALL "GML::SHOW"
433	CALL "GML::SORT"
434	CALL "GML::START"
436	CALL "GML::START_EDIT"
436	CALL "GML::STORE_ROW_DATA"
411	CALL "GML::TPL_PREP"
437	CALL "GML::UPDATE"
5,6	GML.FLAG% values

Procedimiento GML TPL_PREP

Sintaxis
CALL "GML::TPL_PREP"

Propósito:
Inicializa (prepara) la plantilla GML_TPL$ para ser usada. La inicialización del set de variables para pre-fijar valores los cuales son necesarios para la correcta ejecución del procedimiento que será seguido a la plantilla de inicialización.

Nota:
Cada vez que este procedimiento es usado la variable GML.SET% debe ser cargada con el valor del grid específico que tendrá la interface. En la inicialización de la plantilla GML_TLP$ el valor de **GML_TPL.MEMBER es** asignado al valor de 1 por el programa GML

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
```

Procedimiento GML CLEAR

Sintaxis
CALL "GML::CLEAR"

Propósito:
Limpiar el contenido de la celda grid principal y redibujar el grid control.

Nota:
Este procedimiento es válido solo cuando es aplicado al grid principal.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 CALL "GML::CLEAR"
```

Opción:
El procedimiento CLEAR normalmente redibuja el grid. Seteando el GML_TPL.DEF_FLAG% para un valor de 1 prevendrá la realización de la operación de redibujo.

Seteando GML_TPL.OPTION% a uno ejecutará el procedimiento CLEAR de la celda de todos los miembros del set de grid.

Procedimiento GML CURRENT

Sintaxis
CALL "GML::CURRENT"

Propósito:
Trae el número de fila y columna de la celda marcada en el grid principal. Las variables retrieve son localizadas dentro de la plantilla el **GML_TPL$** variables **GML_TPL.ROW_N%** y **GML_TPL.COL_N%**

Nota:
Un **GML_TPL.ROW_N%** valor de -1 indica que no hay celdas dentro del grid seleccionado.
Este procedimiento es válido solo cuando de aplica al grid principal.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 CALL "GML::CURRENT"
```

Procedimiento GML DELETE_ROW

Sintaxis
CALL "GML::DELETE_ROW"

Propósito:
Borra una fila de datos del grid principal en el set de grid seleccionado. Los datos en las celdas son movidos una fila arriba de la fila eliminada.

Nota:
La plantilla **GML_TPL$ variable GML_TPL.ROW_N** es usada para diseñar cual de las filas de datos traerá.

Si no hay celdas dentro del grid seleccionado entonces la celda seleccionada es fijada como la primera celda en el grid. Este procedimiento es válido solo cuando se aplica al grid principal **(GML_TPL.MEMBER%=1).**

Borra 3 filas en el grid principal del set de grid 1.

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.ROW_N%=3
2430 CALL "GML::DELETE_ROW"
```

Opción:
EL procedimiento DELETE_ROW normalmente redibuja el grid. Fijar el GML_TPL.DEF_FLAG% a un valor 1 prevendrá el procedimiento desde la ejecución de la operación redibujar.

Procedimiento GML DESELECT

Sintaxis
CALL "GML::DESELECT"

Propósito:
Deselecciona todos las celdas grids del set de grid principal fila -1

Notas:
Este procedimiento es válido solo cuando aplica al grid principal.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 CALL "GML::DESELECT"
```

Procedimiento GML DISABLE

Sintaxis
CALL "GML::DISABLE"

Propósito:
Deshabilitar el grid control.

Nota:
El valor en **GML_TPL.MEMBER%** determina cual control dentro del set de grid es dehabilitado.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 CALL "GML::DISABLE"
```

Opción:
Fijar el valor de **GML_TPL.OPTION%** a 1 dehabilitará todos los grid control (principal, encabezado de columna y fila) en el set de grid.

Procedimiento GML ENABLE

Sintaxis
CALL "GML::ENABLE"

Propósito:
Habilitar el grid control

Nota:
El valor en GML_TPL.MEMBER% determina cual set de grid control se habilitará.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 CALL "GML::ENABLE"
```

Opción:
Fijar el valor de GML_TPL.OPTION% a 1 habilitará todos los grid control (principal, encabezado de fila y columna) en el set de grid.

Procedimiento GML END_EDIT

Sintaxis
CALL "GML::END_EDIT"

Propósito:
Terminar el modo de edición de una celda grid principal y actualizar la celda grid texto del array de datos.

Nota:
Este procedimiento es ejecutado después que el programa ha recibido un GML.NOTIFY% valor de 6, el cual indica que una tecla especial fue presionada por el usuario durante el proceso de edición de la celda. Cuando el programa GML_M localiza el valor de 6 en la variable GML.NOTIFY también localiza la fila de datos presente dentro de la plantilla GML_TPL$ con el texto recientemente entrado en la variable de plantilla GML_TPL.TEXT$. En adición, un valor correspondiente a una tecla que fue presionada es localizado en la variable GML_TPL.FLAG% Esos valores son como los siguientes:

```
        <Enter> = 1     <F1> = 11
          <Tab> = 2     <F2> = 12
<Shift> + <Tab> = 3     <F3> = 13
```

```
     <Up Arrow> = 4      <F4> = 14
   <Down Arrow> = 5      <F5> = 15
      <Page Up> = 6      <F6> = 16
    <Page Down> = 7      <F7> = 17
          <Esc> = 8      <F8> = 18
                         <F9> = 19
```

Ejemplo:

```
4720 SWITCH GML.NOTIFY%
4730 REM +----------------------+
4740 REM !  SPECIAL KEY PRESSED !
4750 REM +----------------------+
4760 CASE 6
4770 SWITCH GML_TPL.FLAG%
4780 REM +-------------+
4790 REM !  ENTER KEY  !
4800 REM +-------------+
4810 CASE 1
4820 CALL "GML::END_EDIT"
4830 BREAK
```

Opción:
Si, previo a ejecutar el procedimiento END_EDIT, una máscara es localizada dentro de la variable GML_TPL$ plantilla variable GML_TPL.MASK$[c] (donde **[c] es el valor** GML_TPL.COL_N%) será usada para formatear el texto que es contenido en el grid de celda del array de datos.

Procedimiento GML END_EDIT

Sintaxis
CALL "GML::END_EDIT"

El valor en la variable GML_TPL.DEF_FLAG% es aditivo.

El procedimiento END_EDIT normalmente mueve a celda marcada a la siguiente celda editable en el grid. Fijando el GML_TPL.DEF_FLAG%a un valor de 1 prevendrá el procedimiento desde la ejecución de esta acción.

Fijando la variable **GML_TPL.DEF_FLAG% a 2 prevendrá el procedimiento** END_EDIT desde la ejecución de la rutina GML que administra la cantidad de filas visibles que pueden se vistas más allá de la última fila de datos que contiene texto.

Nota Adicional:
Si la siguiente celda editable seleccionada por el procedimiento END_EDIT es está una fila diferente, entonces la variable GML_TPL.FLAG% contendrá la fila vieja (previa)

Procedimiento GML FETCH

Sintaxis
CALL "GML::FETCH"

Propósito:
Recuperar una fila de un grid de celda de datos con atributos. La recuperación de variables son localizadas dentro de la plantilla **GML_TPL$.**

Nota:
El GML_TPL$ variable plantilla GML_TPL.ROW_N% es usada para diseñar cual de las filas de datos es recuperada. Si todas las variables de columna GML_TPL.COL$[c] no contienen datos (vacías) entonces subsecuentemente la ejecución del procedimiento FETCH la variable GML_TPL.ROW_STAT% contendrá el valor de 1, indicando que la fila está vacía.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.ROW_N%=2
2430 CALL "GML::FETCH"
```

Opción:
Si la variable GML_TPL.ROW_N% es fijada a -1 anterior a la ejecución del procedimiento, entonces la fila de datos seleccionada será regresada.

Si la variable GML_TPL.DEF_FLAG% es fijada a 1 previo a la ejecución el procedimiento, solo una fila de datos regresará en la plantilla GML_TPL$ (los atributos de la celda de datos no serán localizados en la plantilla).

Procedimiento GML FIND

Sintaxis
CALL "GML::FIND"

Procedimiento:
Recupera él más bajo número en un grid de fila vacío. La recuperación del número de fila es localizada en la plantilla GML_TPL$ variable GML_TPL.ROW_N%.

Nota:
Un GML_TPL.ROW_N% valor de -1 indica que no hay filas vacías dentro del grid seleccionado.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 CALL "GML::FIND"
```

Opción:
El procedimiento FIND normalmente mueve la celda marcada en el grid principal a la fila vacía (sí la encuentra). Fijando GML_TPL.DEF_FLAG% a valor de 1 encontrará una fila vacía pero no mueve la celda marcada a esa fila.

El procedimiento FIND considera aun celda conteniendo solo espacios como una celda que contiene texto. Si la variable GML_TPL.SUP% es fijada a 1 cuando se ejecuta el procedimiento FIND, la función VPRO5 CVS será usada para tirar espacios. Cuando esta opción es usada, una celda conteniendo solo espacios será considerada como una celda vacía.

Procedimiento GML FOCUS

Sintaxis
CALL "GML::FOCUS"

Propósito:
Fija el enfoque del teclado a una celda específica en el grid principal.

Nota:
Mueve la celda marcada en el grid principal a la celda grid en la fila y encabezado basado en los valores contenidos en la plantilla GML_TPL$ variables GML_TPL.ROW_N% y GML_TPL.COL_N%.

Nota:
Este procedimiento es válido solo cuando se aplica al grid principal.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.ROW_N%=24
2430 GML_TPL.COL_N%=4
2440 CALL "GML::FOCUS"
```

Procedimiento GML GET

Sintaxis
CALL "GML::GET"

Propósito:
Recupera una fila de un grid de celda de datos con atributos. Las variables recuperadas son localizadas dentro de la plantilla GML_TPL$.

Nota:
La plantilla GML_TPL$ variable GML_TPL.ROW_N% es usada para diseñar cual fila de datos será recuperada. Si todos las variables de columna en GML_TPL.COL$[c] no contienen (vacías) entonces subsecuentemente la ejecución del procedimiento GET la variable GML_TPL.ROW_STAT% contendrá el valor de 1, indicando que la fila está vacía.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.ROW_N%=2
2430 CALL "GML::GET"
```

Opción:
Si la variable GML_TPL.ROW_N% es fijada a -1 antes de la ejecución del procedimiento, entonces la fila seleccionada de datos será regresada.

Si la variable GML_TPL.DEF_FLAG% es fijada a 1 antes de la ejecución del procedimiento, solo la fila de datos será regresada en la plantilla GML_TPL$ (los atributos de la celda de datos no serán localizados en la plantilla).

Procedimiento GML GET_STORED_ROW_DATA

Sintaxis
CALL "GML::GET_STORED_ROW_DATA"

Propósito:
Recupera una fila de datos que fue previamente almacenada a través del uso de procedimiento STORE_ROW_DATA. Los datos recuperados son localizados en la plantilla GML_TPL$.

Nota:
Este procedimiento es válido solo cuando es aplicado al grid principal.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL"GML::GET_STORED_ROW_DATA"
```

Opción:
El GML automáticamente almacena fila de datos que más tarde son usadas por el procedimiento RESTORE para restaurar una fila de una celda de datos a los valores que estaban en las filas antes de que nadie haya editado la celda. Fijando la variable GML_TPL.OPTION% a valor 1 recuperará la fila de datos en cambio de los que había almacenado previamente a través del uso del procedimiento STORE_ROW_DATA

Procedimiento GML GOTO

Síntaxis
CALL "GML::GOTO"

Propósito:
Mueve la celda del grid principal a la celda del grid de fila y columna basado en los valores contenidos en la plantilla GML_TPL$ variable GML_TPL.ROW_N% y GML_TPL.COL_N%.

Nota:
Este procedimiento es válido solo cuando es aplicado al grid principal.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.ROW_N%=24
2430 GML_TPL.COL_N%=4
2440 CALL "GML::GOTO"
```

Procedimiento GML HIDE

Sintaxis
CALL "GML::HIDE"

Propósito:
Ocultar un grid control.

Nota:
EL valor en **GML_TPL.MEMBER%** determina cual control dentro del set del grid para ocultarlo.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 CALL "GML::HIDE"
```

Opción:
Fija el valor de GML_TPL.OPTION% a 1 ocultando todos los grid controles (principal, encabezado de columna y fila) en el grid set.

Procedimiento GML INSERT_ROW

Sintaxis
CALL "GML::INSERT_ROW"

Propósito:
Inserta una fila de datos vacía en el grid principal del set de grid seleccionado.

Nota:
La plantilla GML_TPL$ variable GML_TPL.ROW_N% es usada para designar donde va a insertar una fila vacía. Los datos en la celda son movidos 1 fila abajo, si los datos existen en la última fila serán descartados. Este procedimiento es válido solo cuando es aplicado al grid principal. (GML_TPL.MEMBER%=1).

Ejemplo:
Insertar una fila vacía en la fila 1 del grid principal en el grid set 1.

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.ROW_N%=1
2430 CALL "GML::INSERT_ROW"
```

Opción:
El procedimiento INSERT_ROW normalmente redibuja el grid. Fijando el GML_TPL.DEF_FLAG% al valor 1, prevendrá el procedimiento desde la ejecución de la operación de redibujo.

Procedimiento GML LOAD

Sintaxis
CALL "GML::LOAD"

Propósito:
Carga variables desde la plantilla GML_TPL$ dentro de las variables GML apropiadas y posiciones correspondientes. La plantilla GML_TPL$ variable GML_TPL.ROW_N% es usada para designar cual fila de datos será cargada.

Nota:
Este procedimiento no redibuja el grid. Subsecuentemente va cargando todos los datos, el procedimiento REFRESH es usado para causar que el grid se redibuje.

Ejemplo:
Cargando unas tres columnas del grid con datos desde un archivo. Este grid es un miembro del set grid dos. La tecla del archivo asociada con cada registro ha sido almacenada en el campo GML_TPL$.REL[c]. Después el archivo es cerrado, el procedimiento REFRESH redibuja el grid.

```
2300 DIM ITEM$:"NUM:C(6),DESC:C(30),PRICE:N(10)"
.
2400 GML.SET%=2
2410 CALL "GML::TPL_PREP"
2420 ROW%=0
.
2500 READ RECORD(4,END=2590)ITEM$
2510 ROW%=ROW%+1
2520 GML_TPL.COL$[1]=ITEM.NUM$
2530 GML_TPL.COL$[2]=ITEM.DESC$
2540 GML_TPL.COL$[3]=STR(ITEM.PRICE:"#,###,##0.00")
2550 GML_TPL.REL$[1]=KEYP(4)
2560 GML_TPL.ROW_N%=ROW%
2570 CALL "GML::LOAD"
2580 GOTO 2500
2590 CLOSE (4)
2600 CALL "GML::TPL_PREP"
2610 CALL "GML::REFRESH"
```

Procedimiento GML MODE

Sintaxis
CALL "GML::MODE"

Propósito:
Cambia el grid principal a modo "solo lectura".

Nota:
EL valor del GML_TPL.OPTION% determina el modo del grid. Un valor de 0 indica un modo normal con el grid de celda estatus edición, determinado por el valor del GML_TPL.E_MODE% para cada celda individual. Un GML_TPL.OPTION% valor de 1, localiza el grid en modo solo lectura temporalmente sobre escribiendo el valor de las variables GML_TPL.E_MODE% para las celdas, Este procedimiento es válido solo cuando se aplica a la celda principal (GML_TPL.MEMBER%=1).

Ejemplo A:
El grid principal en el set de grid es fijado a uno para modo solo lectura.

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.OPTION%=1
2430 CALL "GML::MODE"
```

Ejemplo B:
EL grid principal en el set de grid es fijado a 2 para modo solo lectura y el grid texto y colores de fondo también es cambiados para todas las celdas en el grid principal.

```
2400 GML.SET%=2
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.OPTION%=1
2430 GML_TPL.COL_N%=-1
2440 GML_TPL.ROW_N%=-1
2450 GML_TPL.T_COLOR$[1]=GML.DKGRAY$
2460 GML_TPL.B_COLOR$[1]=GML.LTGRAY$
2470 CALL "GML::MODE"
```

Opción:
EL procedimiento MODE normalmente redibuja el grid. Fijando GML_TPL.DEF_FLAG% a un valor de 1, prevendrá el procedimiento desde la operación de redibujo.

Procedimiento GML NEXT

Sintaxis
CALL "GML::NEXT"

Propósito:
Mueve la celda marcada en el grid principal a la siguiente celda del grid. Los nuevos valores de la celda marcada fila y columna son localizados dentro de la plantilla **GML_TPL$ variables GML_TPL.ROW_N%** y **GML_TPL.COL_N%.**

Nota:
Si no hay celdas dentro del grid seleccionado, entonces la celda marcada es fijada a la primer celda en el grid. Este procedimiento es válido solo cuando aplica al grid principal. (**GML_TPL.MEMBER%=1).**

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 CALL "GML::NEXT"
```

Opción:
El procedimiento NEXT normalmente mueve la celda marcada a la siguiente celda editable del grid. Fijando GML_TPL.OPTION% a valor 1 moverá la celda marcada a la siguiente celda editable o celda no editable.

Procedimiento GML POPULATE

Sintaxis
CALL "GML: POPULATE"

Propósito:
Localiza texto de la información en el grid de celdas principal como si hubiera entrado directamente a través del uso del procedimiento START_EDIT y END_EDIT

Notas:
Este procedimiento debe ser usado cuando un control como el Listbox, es usado para llenar una celda grid provista de una lista de texto de selecciones en lugar de la entrada de texto actual dentro de la celda grid.

Ejemplo:
Control ID 2402 es una lista de texto que fue generada por el evento de código 1, el cual indica que un click en el ítem de la lista de texto ha ocurrido. EL procedimiento gml CURRENT es ejecutado para determinar la fila del grid asociado con la lista de texto. El texto del ítem de la lista seleccionado es obtenido usando la función CTRL y entonces localizado dentro del grid en la columna 2 (row_n%) moviendo el texto dentro de la plantilla GML_TPL$ variable GML_TPL.COL$[2] .

```
3660 IF EVENT.CODE$= 1 AND EVENT.ID%=2402 THEN GOSUB LB
.
.
7980 LB:
7990 GML.SET%=1
8000 CALL "GML::TPL_PREP"
8010 CALL "GML::CURRENT"
8020 LET ROW_N%=GML_TPL.ROW_N%
8030 CALL "GML::TPL_PREP"
8040 LET GML_TPL.ROW_N%=ROW_N%
8050 LET GML_TPL.COL$[2]=CTRL(SYSGUI,EVENT.ID%,1)
8060 CALL "GML::POPULATE"
```

Procedimiento GML PUT

Sintaxis
CALL "GML::PUT"

Propósito:
Cargar variables desde la plantilla GML_TPL$ dentro de las variables GML correspondientes y posiciones correspondientes. La plantilla GML_TPL$ variable GML_TPL.ROW_N% es usada para designar cual fila de datos va a cargar.

Notas:
Este procedimiento no redibuja el grid. Subsecuentemente va cargando todos los datos, el procedimiento REFRESH es usado para causar que el grid sea redibujado.

Ejemplo:
Cargando columnas del grid con datos desde un archivo. EL grid es un miembro del set de grid 2. El archivo lleva asociado a cada registro leído el cual va almacenando en el campo GML_TPL$.REL[c]. Después el archivo es cerrado, el procedimiento REFRESH redibuja el grid.

```
2300 DIM ITEM$:"NUM:C(6),DESC:C(30),PRICE:N(10)"
.
2400 GML.SET%=2
2410 CALL "GML::TPL_PREP"
2420 ROW%=0
.
2500 READ RECORD(4,END=2590)ITEM$
2510 ROW%=ROW%+1
2520 GML_TPL.COL$[1]=ITEM.NUM$
2530 GML_TPL.COL$[2]=ITEM.DESC$
2540 GML_TPL.COL$[3]=STR(ITEM.PRICE:"#,###,##0.00")
2550 GML_TPL.REL$[1]=KEYP(4)
2560 GML_TPL.ROW_N%=ROW%
2570 CALL "GML::PUT"
2580 GOTO 2500
2590 CLOSE (4)
2600 CALL "GML::TPL_PREP"
2610 CALL "GML::REFRESH"
```

Procedimiento GML REFRESH

Sintaxis
CALL "GML::REFRESH"

Propósito:
Redibuja el grid control con los actuales contenidos y atributos, mueve la celda seleccionada a la primera fila y columna, entonces deselecciona la celda.

Notas:
Este procedimiento es válido solo cuando aplica el grid principal. (GML_TPL.MEMBER%=1).

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 CALL "GML::REFRESH"
```

Opción:
Fijando GML_TPL.OPTION% a 1 antes de ejecutar el procedimiento REFRESH, causará que el grid sea redibujado, pero la celda presente no será cambiada o deseleccionada.

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.OPTION%=1
2430 CALL "GML::REFRESH"
```

Procedimiento GML RESET

Sintaxis
CALL "GML::RESET"

Propósito:
Restablece los atributos de la celda a otros por default. Fija la variable GML.S_STAT%[GML.SET%] a 0, y entonces limpia y redibuja el grid.

Nota:
Este procedimiento es válido solo cuando aplica al grid principal o el control de encabezado de columna (GML_TPL.MEMBER%=1 o GML_TPL.MEMBER%=2).

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 CALL "GML::RESET"
```

Opción:
El procedimiento RESET normalmente redibuja el grid. Fijando GML_TPL.DEF_FLAG% a valor 1 prevendrá el procedimiento desde el grid de redibujo.

Procedimiento GML RESTORE

Sintaxis
CALL "GML::RESTORE"

Propósito:
Restablece una fila de una celda de datos principal a los valores originales; la fila que es restablecida es con los datos que existían en la celda antes de que nadie la haya editado. La plantilla GML_TPL$ variable GML_TPL.ROW_N% es usada para designar cual fila de datos es restaurada.

Nota:
Cuando el enfoque se mueve a una fila nueva (fila cambiada) y la edición empieza en una celda dentro de la fila, la fila de datos en ese tiempo se refiere a los datos originales. Cuando el enfoque cambia a una nueva fila, los datos originales llegan a ser la fila de datos editada. Si el enfoque fue movido a parte de la fila y regresado a la misma fila, la variable GML_TPL.DEF_FLAG% debe ser fijada a -1 antes de la ejecución del procedimiento RESTORE.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.ROW_N%=2
2430 CALL "GML::RESTORE"
```

Opción:
Si la variable GML_TPL.OPTION% es fijada a 1, el procedimiento RESTORE restaurará los datos que fueron almacenados a través del uso del procedimiento STORE_ROW_DATA. Los datos son restaurados incondicionalmente, el valor en la variable GML_TPL.ROW_N% en los datos almacenados determinará cual de las filas será restaurada.

Procedimiento GML SERIES

Sintaxis
CALL "GML::SERIES"

Propósito:
Llena las celdas grid con números en una serie que corresponde a la secuencia de las celdas basadas en su posición de fila y columna.

Nota:
Este procedimiento es normalmente usado durante el desarrollo de una aplicación.

Ejemplo: A:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 CALL "GML::SERIES"
```

Opción_
Fijando GML_TPL.OPTION% a 1 antes de ejecutar el procedimiento SERIES, llenará todos lo miembros del grid con una serie de números.

Ejemplo: B:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.OPTION%=1
2430 CALL "GML::SERIES"
```

Procedimiento GML SET_COLUMN_WITDH

Sintaxis
CALL "GML::SET_COLUMN_WIDTH"

Propósito:
Fija el ancho de una columna en el grid.

Nota:
El valor en la plantilla GML_TPL$ variable GML_TPL.COL$[c] es usado para designar el ancho de cada columna individual en el grid.

Ejemplo:
Fija las primeras 2 columnas en el grid a un ancho de 50 pixeles, y deja 2 en 20 pixeles.

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.COL$[1]="50"
2430 GML_TPL.COL$[2]="50"
2440 GML_TPL.COL$[3]="20"
2450 GML_TPL.COL$[4]="20"
2460 CALL "GML::SET_COLUMN_WIDTH"
```

Procedimiento GML SET_DEFAULT

Sintaxis
CALL "GML::SET_DEFAULT"

Propósito:
Fija los atributos default al grid.

Nota:
Este procedimiento es válido solo cuando aplica al grid principal o control de encabezado de columna (GML_TPL.MEMBER%=1 o GML_TPL.MEMBER%=2) y es usado para fijar atributos default; el procedimiento RESET es usado para establecer todas las celdas grid a los valores default, limpiar y redibujar el grid después de la ejecución del procedimiento SET_DEFAULT.

Ejemplo:
EL color de fondo y alineamiento del texto default para una columna 1 en el grid principal en el set grid 2 es cambiado por el uso del procedimiento SET_DEFAULT. El grid es redibujado por el uso del procedimiento RESET.

```
2400 GML.SET%=2
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.B_COLOR$[1]=GML.LTGRAY$
```

```
2430 GML_TPL.ALIGN%[1]=0
2440 CALL "GML::SET_DEFAULT"
2450 CALL "GML::TPL_PREP"
2460 CALL "GML::RESET"
```

Procedimiento GML SET_EDIT_MASK

Sintaxis
CALL "GML::SET_EDIT_MASK"

Propósito:
Fija a la columna grid default una máscara editable.

Nota:
Este procedimiento es válido solo cuando aplica al grid principal (GML_TPL.MEMBER%=1).

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.MASK$[1]="00/00/0000"
2460 CALL "GML::SET_EDIT_MASK"
```

Procedimiento GML SET_HIGHLIGHT_METHOD

Sintaxis
CALL "GML::SET_HIGHLIGHT_METHOD"

Propósito.
Fija a la celda default colores destacados.

Nota:
Fijando la variable GML_TPL.OPTION% a 0 apagará el destacado. Fijando GML_TPL.OPTION% a 1 dibujará un rectángulo alrededor de la celda sin cambiar el color, mientras fija GML_TPL.OPTION% a 2 será marcada la celda presente con los colores destacados.

Este procedimiento es válido solo cuando aplica al grid principal.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.OPTION%=1
2430 CALL "GML::SET_HIGHLIGHT_METHOD"
```

Opción:
Cuando el marcado es activo, fijando la variable GML_TPL.DEF_FLAG% a 1, aplicará el marcado a una celda individual. Fijando GML_TPL.DEF_FLAG% a 1 aplicará el marcado a la fila de celdas.

Procedimiento GML SET_IMAGELIST

Sintaxis
CALL "GML::SET_IMAGELIST"

Propósito:
Fija él ID de la lista de imagen que contiene las imágenes a ser desplegadas en el grid control. El GML_TPL$ variable GML_TPL.OPTION% contiene el actual valor de los ID control. GML_TPL.IMAGE% aunque el uso del procedimiento UPDATE es entonces para fijar el imagelist índex de la imagen a ser desplegada dentro de la celda grid.

Nota:
Antes de usar este procedimiento, el mnemónico IMAGELIST debe ser usado para fijar parámetros y recuperar archivo de imágenes del bitmap de imágenes. EL ID usado con este procedimiento es el mismo ID usado con el mnemónico IMAGE. Este procedimiento es usado para inicializar la asignación del ID o cambiar un ID previamente asignado.

Ejemplo:
En este ejemplo el mnemónico IMAGELIST asigna un ID control de 2501 al archivo imagelist llamado images.bmp e indica el ancho de cada imagen dentro del archivo de 15 pixeles. El procedimiento SET_IMAGELIST entonces liga él ID imagelist al grid específico que desplegará la imagen contiene dentro el archivo imagelist. En la última parte del ejemplo, el procedimiento UPDATE asigna la imagen al índice 2, el cual es la tercera imagen en el archivo imagelist (las posiciones de la imagen son basadas en 0) a la celda grid en al fila 4, columna 3 en el grid set 1.

```
.
1500 PRINT (SYSGUI)'IMAGELIST'(2501,15,"images.bmp")
.
.
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.OPTION%=2501
2430 CALL "GML::SET_IMAGELIST"
.
.
3010 GML.SET%=1
3020 CALL "GML::TPL_PREP"
3030 GML_TPL.ROW_N%=4
3040 GML_TPL.IMAGE%[3]=2
3050 CALL "GML::UPDATE"
```

Procedimiento GML SET_LINE_MODE

Sintaxis
CALL "GML::SET_LINE_MODE"

Propósito:
Despliega o oculta el grid principal las líneas de separación horizontal y verticales.

Nota:
La variable de plantilla GML_TPL.ROW_N% es usada para fijar la separación de líneas del grid principal (línea de fila) a invisible o visible. Los valores válidos para la variable GML_TPL.ROW_N% son:

0 no cambia
1 Oculta la línea de separar horizontal
2 muestra la línea de separar horizontal

La variable de plantilla **GML_TPL.COL_N** es usada para fijar la línea de separación del grid vertical (línea de columna) a visible o invisible. Los valores válidos para la variable GML_TPL.COL_N% son:

0 no cambia
1 Oculta la línea de separar vertical
2 muestra la línea de separar vertical

Ejemplo:
Despliega líneas de separación verticales y horizontales.

```
GML.SET%=1
GML_TMP.ROW_N%=2
GML_TPL.COL_N%=2
CALL "GML::SET_LINE_MODE"
```

Procedimiento GML SET_OUT_MASK

Sintaxis
CALL "GML::SET_OUT_MASK"

Propósito:
Fija la máscara de salida de la columna usada para desplegar datos en el grid.

Nota:
Una máscara conforma el formato perfilado para la función VPRO/5 STR( ), es localizada dentro de la plantilla GML_TPL$ variable GML_TPL.MASK%[c]. Si la manipulación del string de datos es requerida cuando aplica la máscara, entonces la correspondiente variable GML_TPL.STYLE%[c] debe ser cargada con un valor que es el mismo usado con la función CVS string de conversión.

Si un error es encontrado con la máscara durante la actividad del redibujado del grid, entonces los datos para la celda serán desplegados sin el uso de la máscara.

Cuando una máscara contiene un carácter 0 (decimal 48) el dato en la celda grid es convertido a numérico antes de aplicar la máscara y entonces la manipulación del string es realizada si la variable GML_TPL.STYLE%[c] contiene un valor más grande que 0.

Fijando la variable GML_TPL.COL_N% a -1 las instrucciones GML para aplicar la máscara y el valor contenido en las plantillas variables GML_TPL.MASK$[1] y GML_TPL.STYLE%[1] para todas las columnas en el grid. Fijando una variable GML_TPL.MASK%[c] a nulo ("") removerá una máscara existente.

Este procedimiento es válido solo cuando aplica al grid principal. (**GML_TPL.MEMBER%=1**).

Ejemplo:
El perfil de máscara para la columna 5 en el set grid 2 es fijado a "###,###0,00" y el correspondiente valor del string es fijado a 1 (tira espacios).

```
2400 GML.SET%=2
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.MASK$[5]= "###,##0,00"
2430 GML_TPL.STYLE%[5]=1
2440 CALL "GML::SET_OUT_MASK"
```

Opción:
Fijando GML_TPL.OPTION% a 1 antes de ejecutar el procedimiento SET_OUT_MASK redibujará el grid con el valor de máscara actualizado.

Procedimiento GML SET_ROWS

Sintaxis
CALL "GML::SET_ROWS"

Propósito:
Fija el número de filas que pueden ser vistas más allá del último grid de fila que contiene texto y un mínimo número de filas visibles para desplegar.

Nota:
La plantilla GML_TPL$ variable GML_TPL.ROW_N% es usada para designar el número de filas que pueden ser vistas más allá del último grid de filas que contiene texto. El valor de esta variable debe ser en el rango de -1 al máximo número de filas específicas cuando el grid fue creado. Especificando un valor de -1 indica que el máximo de filas a ser vistas más allá de la última fila que contiene texto es igual a las filas actuales del grid. Este procedimiento es válido solo cuando aplica al grid principal (**GML_TPL.MEMBER%=1).**

Ejemplo:
En este ejemplo el grid desplegado mostrará una fila más allá de la última fila en el grid que contiene texto. Si la última fila en el grid conteniendo texto es la fila número 15, entonces el máximo de filas que pueden ser vistas es 16.

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.ROW_N%=1
2430 CALL "GML::SET_ROWS"
```

Opción:
Fijando GML_TPL.OPTION% a 1 antes de ejecutar el procedimiento SET_ROWS fija el número de filas visibles para desplegar. Fijando el valor de la variable GML_TPL.ROW_N% a -1 indica que el número mínimo de filas a ser desplegadas es igual al número de filas visibles que pueden ser desplegadas en el grid control. Recíprocamente, fijando el GML_TPL.ROW_N% a valor 1 causará que despliegue solo una línea del grid cuando la primera fila del grid está vacía.

Fijando la variable GML_TPL.OPTION% a 2 aplicará el valor en GML_TPL.ROW_N% para ambos la cantidad de filas a ser vistas más allá de la última fila conteniendo texto y para la mínima cantidad de filas visibles para desplegar. Un GML_TPL.OPTION% de 2 es equivalente a ejecutar el procedimiento SET_ROWS con una opción de 0 y entonces una opción de 1.

Procedimiento GML SET_ROW_HEIGHT

Sintaxis
CALL "GML::SET_ROW_HEIGHT"

Propósito:
Fija la altura del grid principal.

Nota:
El valor en GML_TPL.OPTION% determina la altura de la fila en pixeles.

Ejemplo:
Fija la altura del grid principal a 18 pixeles.

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2430 GML_TPL.OPTION%=18
2440 CALL "GML::SET_ROW_HEIGHT"
```

Procedimiento GML SHOW

Sintaxis
CALL "GML::SHOW"

Propósito:
Muestra un grid control.

Nota:
El valor en GML_TPL.MEMBER% determina cual control dentro del set grid para mostrarlo.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 CALL "GML::SHOW"
```

Opción:
Fijando el valor de GML_TPL.OPTION% a 1 mostrará todos los grid control (principal, encabezado de columna y fila) en el set de grid.

Procedimiento GML SORT

Sintaxis
CALL "GML::SORT"

Propósito:
Ordena los datos del grid principal basados en los datos contenidos en una columna específica.

Nota:
La plantilla GML_TPL$ variable GML_TPL.ROW_C% es usada para designar cual columna es usada para el ordenamiento de la llave. Después de ordenar los datos, el grid sé redibuja, el número de columna relata la más reciente ejecución del procedimiento SORT que es retenido en GML.S_STAT%[GML.SET%]. Un valor positivo indica que el ordenamiento fue acordado en orden ascendente, mientras que un valor negativo indica un ordenamiento descendente. Cuando el procedimiento RESET es ejecutado, el valor en GML$ variable GML.S_STAT%[GML.SET%] es fijado a 0.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.COL_N%=3
2430 CALL "GML::SORT"
```

Opción: Los valores en la variable **GML_TPL.OPTION%** son aditivos.

Si GML_TPL.OPTION% es fijado a 1, el GML_TPL.REL$[sort column] valor será usado como llave para el ordenamiento.

Si GML_TPL.OPTION% es fijado a 2, el ordenamiento será acordado en orden descendente (orden inverso).

Fijando GML_TPL.OPTION% a 3 usará la GML_TPL.REL$[sort column], con el ordenamiento en orden descendente.

GML_TPL.OPTION% fijado a 4 actualizará los encabezados de columna después de que el procedimiento SORT es ejecutado. Esta opción es la más usada cuando la plantilla GML_TPL$ fue usada para ejecutar el procedimiento SORT, además contiene cambios en las variables GML_TPL$ afectando imágenes agregadas o removidas del encabezado de columna.

GML_TPL.ARG% es usada para especificar el uso de la función CVS (string de conversión) cuando está construyendo las llaves de ordenamiento. Los valores GML_TPL.ARG% son los mismos que son usados con la función CVS.

Si una máscara es provista en la variable GML_TPL.MASK$[sort column] y la máscara contiene cero y un punto decimal (opcional) la columna será ordenada como un numérico usando la máscara.

Nota Adicional:
Si hay un problema cuando se ejecuta el procedimiento SORT, la variable **GML.FLAG%** contendrá un -18. En adición la variable GML_TPL.FLAG% contendrá un número de error de VPRO/5, y la **GML_TPL.ROW_N%** contendrá el número de fila que causa el error.

Si el procedimiento SORT es ejecutado mientras una celda está en modo edit una GML.FLAG% valor de -19 será regresado (el procedimiento SORT no se ejecutará).

Procedimiento GML START

Sintaxis
CALL "GML::START"

Propósito:
Identificar el grid a ser manejado por el Manejo de librerías Grid.

Nota:
Este procedimiento es usado una vez en el código de aplicación del startup, solo antes del llamado al programa GML_I Cuando el procedimiento START es ejecutado una plantilla **GML_INIT$** es regresada al llamado de programas con las variables dimensionadas basadas en el valor de la variable **GML_QTY%**, la cual denota que la cantidad de set de grid a ser manejados por el GML. Las variables de plantillas son las siguientes:

GML_INIT.MAIN_ID%[s]	Id del grid principal
GML_INIT.COL_HEADER_ID%[s]	Id del control de encabezado de columna
GML_INIT.ROW_HEADER_ID%[s]	Id del control de encabezado de fila.
GML_INIT.SYSGUI%[s]	Número canal dispositivo del sistema
GML_INIT.CONTEXT%[s]	Control contexto
GML_INIT.LUM_QTY%	Cantidad de controles look-up

Un ID control del grid principal debe contener un valor que regrese un número 1 cuando está usando la función MOD(n,5), como ejemplo donde MOD(5001,5)=1. Un ID control encabezado de columna para el mismo grid debe ser igual al grid principal control ID+1, mientras él ID control encabezado de fila debe ser igual a el ID+2 control del grid principal. Si un grid principal no tiene un control de encabezado de columna asociado o de fila, entonces las variables para estos ID controles la plantilla GML_INIT$ deben ser fijados a 0.

Ejemplo:
Dos grids deben ser manejados por el GML. Desde que el grid uno se fija a 1 no contiene un control de encabezado de fila, el ID encabezado de fila es fijado a 0.

```
4130 LET GML_QTY%=2
4140 CALL "GML::START"
4150 LET GML_INIT.MAIN_ID%[1]=5001
4160 LET GML_INIT.COL_HEADER_ID%[1]=5002
4170 LET GML_INIT.ROW_HEADER_ID%[1]=0
4180 LET GML_INIT.SYSGUI%[1]=SYSGUI%
4190 LET GML_INIT.CONTEXT%[1]=1
4200 LET GML_INIT.MAIN_ID%[2]=5021
4210 LET GML_INIT.COL_HEADER_ID%[2]=5022
4220 LET GML_INIT.ROW_HEADER_ID%[2]=5023
4230 LET GML_INIT.SYSGUI%[2]=SYSGUI%
4240 LET GML_INIT.CONTEXT%[2]=2
4250 LET GML_INIT.LUM_QTY%=1
4260 CALL "GML_I",SYSGUI,GML_INIT$,GML$,GML_GM$[ALL],GML_GC$[ALL],
                GML_GR$[ALL],GML_SET$
```

Procedimiento GML START_EDIT

Sintaxis
CALL "GML::START_EDIT"

Propósito:
Iniciar el modo edición en el grid principal basado en los valores en la plantilla GML_TPL$ variables GML_TPL.ROW_N% o GML_TPL.COL_N%

Nota:
Este procedimiento solo es válido cuando aplica al grid principal.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2430 GML_TPL.ROW_N%=4
2440 GML_TPL.COL_N%=2
2450 CALL "GML::START_EDIT"
```

Opción:
Localizando texto en la variable GML_TPL.TEXT$ antes de ejecutar el procedimiento START_EDIT localizará el valor dentro de la celda grid seleccionada cuando la celda cambie al modo edit. Si la variable GML_TPL.MASK$[c] contiene una máscara y [c] es igual al valor en GML_TPL.COL_N%, será usada como una máscara para datos de entrada dentro de las celdas.

Si el procedimiento START_EDIT es ejecutado y la celda a ser editada es fijada a 0 en modo no editables, el procedimiento regresará un -9 o -12 en la variable GML.FLAG% indicando que el procedimiento START_EDIT no fue exitosamente ejecutado. Fijando la plantilla GML_TPL$ variables GML_TPL.DEF_FLAG% a -1 permitirá la edición de una celda no editable.

Procedimiento GML STORE_ROW_DATA

Sintaxis
CALL "GML::STORE_ROW_DATA"

Procedimiento:
Almacena los datos en la plantilla en la posición en array GML_GM$ que pueda ser recuperado un poco después a través del uso del procedimiento GET_STORED_ROW_DATA.

Nota:
Este procedimiento es válido solo cuando aplica al grid principal.

Ejemplo:

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.ROW_N%=4
2430 CALL"GML::FETCH"
2440 CALL"GML::STORE_ROW_DATA"
```

Procedimiento GML UPDATE

Sintaxis
CALL "GML::UPDATE"

Propósito:
Actualiza los datos del grid y atributos con valores contenidos en la plantilla GML_TPL$ y redibuja el grid control. La plantilla GML_TPL$ variable GML_TPL.ROW_N% es usada para designar cual de la fila de datos es actualizada.

Nota:
Este procedimiento es normalmente usado para actualizar un grid sencillo de celdas en una fila del grid. Cuando carga muchas celdas como cuando lee de un archivo, el procedimiento LOAD podría ser usado.

Ejemplo A:
Actualizando la celda grid localizada en el grid set1, fila 3, columna 2 con texto y fijando el grid de celda color a rojo.

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.ROW_N%=3
2430 GML_TPL.COL$[2]="John Smith"
2440 GML_TPL.T_COLOR$[2]=GML.RED$
2450 CALL "GML::UPDATE"
```

Opción:
Fijando GML_TPL.OPTION% a 1 antes de ejecutar el procedimiento UPDATE, actualizará los datos del grid, pero no redibuja la celda que será seleccionada, cambiada o deseleccionada. Un GML_TPL.OPTION% fijado a 2 causa que el grid sea redibujado, pero los datos de la celda y atributos no serán cambiados.

Ejemplo B:
Este ejemplo usa dos procedimientos para exhibir cuando el procedimiento UPDATE es usado con la GML_TPL.OPTION% fija un valor de 2. Usando el procedimiento REFRESH en lugar del procedimiento UPDATE logrará los mismos resultados.

```
2500 READ RECORD(4,END=2590)ITEM$
2510 ROW%=ROW%+1
2520 GML_TPL.COL$[1]=ITEM.NUM$
2530 GML_TPL.COL$[2]=ITEM.DESC$
2540 GML_TPL.COL$[3]=STR(ITEM.PRICE:"#,###,##0.00")
2550 GML_TPL.REL$[1]=KEYP(4)
2560 GML_TPL.ROW_N%=ROW%
2570 CALL "GML::LOAD"
2580 GOTO 2500
2590 CLOSE (4)
2600 CALL "GML::TPL_PREP"
2610 GML_TPL.OPTION%=2
2620 CALL "GML::UPDATE"
```

Procedimiento GML UPDATE

Sintaxis
CALL "GML::UPDATE"

Ejemplo C:
Normalmente la variable GML_TPL.ROW_N% contiene un valor entre 1 y el máximo de filas en el grid, una variable GML_TPL.COL_N% contiene el valor de 0. Fijando GML_TPL.ROW_N% a -1 es interpretado por un programa GML como una instrucción para aplicar el update a todas las columnas del grid. Por eso fijando GML_TPL.ROW_N% a -1 y GML_TPL.COL_N% a -1 aplicará los cambios en la columna [1] a cada celda en el grid. Este producto es normalmente usado para cambiar los atributos del grid.

Los siguientes cambios en todas las celdas del grid principal set 1 para desplegar el grid celda de texto con color de fondo gris y texto color gris oscuro. En este ejemplo la variable GML_TPL.DEF_FLAG% es fijada a 1, instrucción que programa el GML para actualizar el color de fondo del grid principal al color de fondo en la variable GML_TPL.B_COLOR$[1]. Usando este producto mejora la lista del grid en las celdas que son desplegadas. Si por ejemplo el color de fondo era gris claro y el de fondo fue blanco, el grid podría aparecer a "flash" como el grid haya sido redibujado. Los valores válidos para variables GML_TPL.DEF_FLAG% son:

Valor	Significado
0	no cambia
1	Fija el color de texto default al valor GML_TPL.T_COLOR$[1]
1	Fija el color de fondo default al valor en GML_TPL.B_COLOR$[1]
3	Aplica ambos colores 1 y 2 al grid de colores default.

```
2400 GML.SET%=1
2410 CALL "GML::TPL_PREP"
2420 GML_TPL.ROW_N%=-1
2430 GML_TPL.COL_N%=-1
2440 GML_TPL.B_COLOR$[1]=GML.LTGRAY$
2450 GML_TPL.T_COLOR$[1]=GML.DKGRAY$
2460 GML_TPL.DEF_FLAG%=2
```

2470 **CALL "GML::UPDATE"**

Opción Adicional:
Fijando la variable GML_TPL.DEF_FLAG% a 4 prevendrá el procedimiento actualización de ejecutar una rutina GML que administre la cantidad de filas visibles que pueden ser vistas más allá de la última fila de datos que contienen texto.

Interpretando La notificación de eventos: GML:MOTIFY%

A través del uso de la plantilla GML_TPL$ los procedimientos GML son usados para iniciar cambios a controles dentro de un grid específico. Cuando un notificador de eventos del grid control ocurre (código de evento en N con un tipo de objeto 107), programas GML_M maneja el evento por ejecución específica de operaciones basadas en el número de código de evento.

Ciertos eventos como códigos de evento 22 (TABLEUPDATE) son procesados por el programa GML_M de manera que no requiere ser procesado por la aplicación. Otros eventos, como el código de evento 18 (RCLICKED) son pre-procesados por un programa GML_M y entonces pasa a la aplicación para procesos opcionales adicionales. La determinación de cual evento notificador en el acto y que tipo de procesamiento adicional para aplicar una aplicación específica.

Cuando un evento grid es pre-procesado por el programa GML_M, la plantilla GML_TPL$ es inicializada y entonces sé cargada con los datos de fila que corresponden a la fila del grid que era el origen del evento. En adición a otras variables dentro de la plantilla GML_TPL$ son actualizadas con información suplementaria que relaciona al evento específico. En ciertos casos varias notificaciones de eventos son combinadas dentro del valor del GML.NOTIFY%. El propósito de pre-procesar el evento es para preparar la plantilla GML_TPL$ en una forma que permita a la plantilla ser ejecutada con procedimientos GML sin requerir una plantilla adicional.

Otros eventos como el código de evento 2 (COLCHANGE) son considerados por el programa GML_M como eventos asesores. En estos casos, el programa GML:M inicializa la plantilla GML_TPL$, pero la plantilla queda en el estado inicializado como indicado por la variable GML_TPL.ROW_N% conteniendo un valor de 0.

Se debe notar que el programa GML_M usa la información contenida en la plantilla notice event para manejar eventos a través del uso de la variable GML.NOTIFY% y asociada a la plantilla. La información adicional contenida en la plantilla notice no es cambiada por el GML. La aplicación puede usar otra plantilla notice evento de información, GML.NOTIFY%, o ambos para el acto en el evento del grid.

El programa GML_M usa la plantilla GML$ variable GML.NOTIFY% para notificar a la aplicación que un evento ha ocurrido. Para todos los eventos en pre-procesos la plantilla GML_TPL$ contiene la siguiente información, la cual es similar a usar el FETCH para obtener datos de la fila.

Nombre Variable	Contenido de Variable
GML_TPL.ROW_N%	Número de evento de la fila
GML_TPL.COL_N%	Número de evento de la columna (cuando aplica)
GML_TPL.COL$[c]	texto de celda
GML_TPL.REL$[c]	texto relacionado
GML_TPL.T_COLOR$[c]	color de texto de la celda columna.
GML_TPL.B_COLOR$[c]	color de texto del fondo.
GML_TPL.STYLE%[c]	Estilo de la celda
GML_TPL.ALIGN%[c]	Alineación de la celda
GML_TPL.IMAGE%[c]	No. de índice de la imagen desplegada dentro de la celda.
GML_TPL.E_MODE%[c]	celda en modo editable
GML_TPL.ROW_STAT%	El valor de 1 si todas las celdas dentro de la fila están vacías o 0 si las celdas no están vacías.

Otras variables dentro de la plantilla GML_TPL$ contienen valores suplementarios basados en códigos de eventos específicos y son como los siguientes:

GML.NOTIFY% 6: Tecla especial que cuando se presiona edita una celda:
GML_TPL.TEXT$ Contiene el control de editor de texto. GML_TPL.FLAG% contiene un valor que corresponde a una tecla especial que fue presionada. Los valores son los siguientes:

<Enter> = 1	<F1> = 11
<Tab> = 2	<F2> = 12
<Shift> + <Tab> = 3	<F3> = 13
<Up Arrow> = 4	<F4> = 14
<Down Arrow> = 5	<F5> = 15
<Page Up> = 6	<F6> = 16
<Page Down> = 7	<F7> = 17
<Esc> = 8	<F8> = 18
	<F9> = 19

GML.NOTIFY% 9 : Inicia la demanda de edit.
GML_TPL.FLAG% = 3 Inicia la demanda del edit iniciado por el click del mouse.
GML_TPL.FLAG% = 9 Inicia la demanda del edit iniciado presionando el enter
GML_TPL.FLAG% = 12 inicia la demanda del edit iniciado por una tecla presionada, con el valor ASCIII que contiene la tecla en la plantilla variable GML_TPL.OPTION%.

GML.NOTIFY% 12:Teclas teclado:
Si la tecla space fue presionada, GML_TPL.FLAG% contendrá un 27; GML_DEF_FLAG% contendrá un 0 si la tecla fue presionada cuando la celda no estaba en modo edición o un 1 si la tecla fue presionada cuando la celda había sido editada.

Si la tecla delete fue presionada GML_TPL_FLAG% contendrá un 127.

GML.NOTIFY% 14:Click Izquierdo:
GML_TPL.FLAG% contiene un 1 cuando el botón del mouse es presionado y 0 cuando se suelta.

GML.NOTIFY% 15:Click Izquierdo (celda seleccionada)
Si el control o teclas shift son presionados cuando el botón del mouse es presionado, GML_TPL.FLAG% contendrá uno de los siguientes valores.

GML_TPL.FLAG% = 1 tecla control presionada
GML_TPL.FLAG% = 2 tecla shift presionada
GML_TPL.FLAG% = 3 teclas shift y control presionadas.

GML.NOTIFY% 18:Click derecho
GML_TPL.FLAG% contiene un 1 cuando el botón del mouse es presionado y 0 cuando se suelta.

GML.NOTIFY% 19:Cambio fila
GML_TPL.FLAG% contiene el número de fila antes del cambio de fila.

Interpretando Eventos grid notify.

Ejemplo Interpretando GML.NOTIFY% = 9 / GML_TPL.FLAG% = 3 Inicia celda edit:
En este ejemplo el mouse tuvo un doble click izquierdo en la celda localizada en la fila 2, columna 4 en el grid principal del set 1. La plantilla GML_TPL$ ha sido preparada por el programa GML_M y además también contiene la fila seleccionada y los atributos de los datos. En este caso, el pre-proceso ha realizado el equivalente a lo siguiente:

```
GML.SET%=1
CALL "GML::TPL_PREP"
GML_TPL.ROW_N%=2
GML_TPL.COL_N%=4
CALL"GML::FETCH"
```

La aplicación ha sido realizada para interpretar este evento como un inicio de petición de edit. Desde que el pre-proceso realizado por el programa GML_M ha preparado la plantilla GML_TPL$ y la pasó directamente al programa GML con un llamado al procedimiento START_EDIT:

```
IF EVENT.OBJTYPE%=107 THEN IF EVENT.CODE$="N" THEN GOSUB GRID_MANAGER
.
GRID_MANAGER:
CALL "GML_M",SYSGUI,EVENT$,NOTICE$,GML_SET$,GML_TPL$,GML$,
             GML_GM$[ALL],GML_GC$[ALL],GML_GR$[ALL]

SWITCH GML.NOTIFY%
  .
  CASE 9
         SWITCH GML_TPL.FLAG%
               .
                CASE 3
                       CALL "GML::START_EDIT"
                       BREAK
         .
         SWEND
  .
SWEND
```

Si la aplicación fue programada para iniciar el inicio de edit en respuesta al doble click del mouse, presionando la tecla enter o presionando una tecla en una celda, la lógica aparece como sigue:

```
SWITCH GML.NOTIFY%
  .
  CASE 9
         SWITCH GML_TPL.FLAG%
               .
                CASE 3
                CASE 9
                CASE 12
                       CALL "GML::START_EDIT"
                       BREAK
         .
         SWEND
   .
SWEND
```

Manejo de Librerías Grid: Manejo productos Look-up (LUM)

Las librerías grid pueden ser configuradas para asociar un tipo de lista de control, como una lista de botón o una lista edit, con un evento particular que ocurre en una columna específica dentro del grid principal. Este producto ofrece la capacidad para mostrar temporalmente, la posición del grid de celda principal, como una lista tipo control a ser usada por el look-up t seleccionar información desde una lista de Opción presentada dentro del control. El GML administra las tareas asociadas con el movimiento, tamaño, control, mostrando y ocultando el look-up control. La opción de GML que manejan esas tareas es referida como el LUM.

Una plantilla llamada GML_LUM$ es usada para almacenar parámetros look-up. El GML LUM es capaz de manejar más de un look-up, es por eso que un array subscript[n] es usado para designar la asignación de valores look-up. Las variables dentro de esta plantilla requieren ser inicializadas por la aplicación de la siguiente forma:

GML_LUM$ Variable de plantilla ([n] indica número look-up)
GML_LUM.LU_ID%[n] ID del look-up control
GML_LUM.SET%[n] Fija el número del grid asociado
GML_LUM.COL%[n] Número de columna dentro del grid principal que será usado para activar (desplegar) look-up control.

GML_LUM.LOC%[n] EL valor en esta variable indica donde el control look-up va a ser desplegado, relativo al número de columna asignado a la variable GML_LUM.COL%[n] Como por ejemplo , si una variable GML_LUM.COL%[n] contiene una valor de 4 y una variable GML_LUM.LOC%[n] tiene asignado un valor de-1, entonces la columna 4 es desplegada en el grid activo del look-up y el look-up control es desplegado en la celda in la columna 3. Si un GML_LUM.LOC%[n] fue asignado un valor de 0, entonces el look-up control podría ser activado y desplegado dentro de la misma celda dentro de la columna. Los valores válidos para la GML_LUM.LOC%[n] son -1,0 y 1.

GML_LUM.SCOPE%[n] cada vez que un look-up control es desplegado, este es movido y reajustado al overlay en la columna identificada por la suma de variables GML_LUM.COL%[n] y GML_LUM.LOC%[n]. EL valor contenido en la variable GML_TPL.SCOPE%[n] determina el ancho del look-up control.

GML_LUM.X_ADJ%[n]
GML_LUM.Y_ADJ%[n]
GML_LUM.W_ADJ%[n] las variables ADJ%[n] son usadas para ajustar la ubicación del look-up control. Los valores en esas variables son iniciados a +3 para X_ADJ%[n], +1 para Y_ADJ%[n], y -1 para W_ADJ%[n], pero pueden ser cambiadas por la aplicación.

Grid Management Library : Look-Up Manager (LUM)

Información Adicional

El efecto de los valores en la variable GML_LUM.SCOPE%[n] en el ancho del look-up control están basados en valores que son asignados a variables GML_LUM.LOC%[n].

EV = Columna evento
LU = Columna Look-Up control
X = Mira (ancho) del control

GML_LUM.COL%[n]=3
GML_LUM.LOC%[n]=0

	Grid Columns				
GML_LUM.SCOPE%[n]			**EV LU**		
0			**X**		

1		X	X		
2			X	X	
3		X	X	X	
4 (width of grid control)	X	X	X	X	X

GML_LUM.COL%[n]=3
GML_LUM.LOC%[n]=-1

	Grid Columns				
GML_LUM.SCOPE%[n]		**LU**	**EV**		
0		X			
1		X	X		
2		X	X	X	
4 (width of grid control)	X	X	X	X	X

GML_LUM.COL%[n]=3
GML_LUM.LOC%[n]=1

	Grid				
GML_LUM.SCOPE%[n]			**EV**	**LU**	
0				X	
1			X	X	
2		X	X	X	
4 (width of grid control)	X	X	X	X	X

GML Look-Up Manager (LUM) : Start-Up

EL GML LUM es configurado durante la aplicación de start-up como sigue: Paso1: AL start-up, asigna la GML_INIT$ variable GML_INIT.LUM_QTY% la cantidad de look-up control a ser usados por la aplicación.

```
4130 LET GML_QTY%=2
4140 CALL "GML::START"
4150 LET GML_INIT.MAIN_ID%[1]=5001
4160 LET GML_INIT.COL_HEADER_ID%[1]=5002
4170 LET GML_INIT.ROW_HEADER_ID%[1]=0
4180 LET GML_INIT.SYSGUI%[1]=SYSGUI%
4190 LET GML_INIT.CONTEXT%[1]=1
4200 LET GML_INIT.MAIN_ID%[2]=5021
4210 LET GML_INIT.COL_HEADER_ID%[2]=5022
4220 LET GML_INIT.ROW_HEADER_ID%[2]=5023
```

```
4230 LET GML_INIT.SYSGUI%[2]=SYSGUI%
4240 LET GML_INIT.CONTEXT%[2]=2
4250 LET GML_INIT.LUM_QTY%=1
4260 CALL "GML_I",SYSGUI,GML_INIT$,GML$,GML_GM$[ALL],GML_GC$[ALL],
                 GML_GR$[ALL],GML_SET$
```

Paso 2: Siguiendo el llamado al programa GML_I, inicializa la plantilla LUM por la ejecución del siguiente procedimiento:

```
4340 REM +-----------------------+
4350 REM !  OBTAIN LUM TEMPLATE  !
4360 REM +-----------------------+
4370 CALL "GML::INIT_LUM_TPL"
```

Paso 3: Asigna valores a las variables de plantilla GML_LUM$ como son apropiadas:

```
4380 REM +---------------------+
4390 REM !  LU #1 INFORMATION  !
4400 REM +---------------------+
4410 GML_LUM.LU_ID%[1]=122
4420 GML_LUM.SET%[1]=1
4430 GML_LUM.COL%[1]=2
4440 GML_LUM.LOC%[1]=0
4450 GML_LUM.SCOPE%[1]=0
```

Nota:
El IC look-up control, valor de 122 en el GML_LUM.LU_ID%[1] se refiere a un tipo de lista botón o lista edit control que existe a la vez de la inicialización de plantilla GML_LUM$. Este control debe existir en el mismo contexto que el grid control en el set grid valor asignado a la variable GML_LUM.SET%[1]. En adición el control podría ser visible, pero localizado más allá del ancho de la pantalla. Como un ejemplo, si el ancho de la ventana es de 550 pixeles, el valor de x del look-up control podría ser más grande que 550.

Paso 4: Inicia el Look-up Manager

```
4620 REM +--------------------+
4630 REM !  START LU MANAGER  !
4640 REM +--------------------+
4650 CALL "GML::LUM_START"
```

GML Look-Up Manager (LUM) : Desplegando el The Look-Up Control

Cuando el evento grid 14 (click izquierdo), 15 (click izquierdo en la ceda) o 18 (click derecho) ocurre en el grid principal (miembro 1=}) del set grid, la plantilla GML_TPL$, variable GML_TPL.INFO$ contiene información relacionando la localización del click.

Esta información puede ser pasada en el look-up manager para desplegar el look-up control en la fila donde el evento fue generado. La actual posición del control es determinada por los valores de las variables contenidas en la plantilla GML_LUM$ que fueron inicializadas en la aplicación start-up.

En el siguiente ejemplo, GML.NOTIFY% 18 ha sido usado como el evento que activa (muestra) el look-up control. Si hay un control asociado con la información contenida en la plantilla GML_TPL$, ese control es desplegado. Sin embargo, si el look-up control no es asociado con la información GML_TPL$, la llamada al LUM_DISPLAY no tendrá efecto.

```
7170 REM +------------------+
7180 REM !  RIGHT CLICK     !
7190 REM !  GML.NOTIFY%=18  !
7200 REM +------------------+
7210 CASE 18
7220 SWITCH GML_TPL.MEMBER%
7230 CASE 1
7240 IF GML_TPL.FLAG%=1 THEN CALL "GML::LUM_DISPLAY"
7250 BREAK
7260 CASE 2
```

GML Look-Up Manager (LUM) : Respondiendo a eventos

El GML LUM debe ser llamado cuando un evento list edit o list butón ocurre ((GOSUB LUM_LOOK_UP). En adición cuando otros eventos ocurren, el LUM debe ser llamado para determinar cuando ocultar un look-up control GOSUB LUM_EVENT) que es visible (GML_LUM_FLAG%<>0), pero el enfoque ha sido movido a otra parte. La línea 3650 que sigue en el ejemplo que contiene el código necesario para accesar a las subrutinas que proveen acceso al GML LUM para administrar los eventos. Esta línea de código podría ser insertada en la aplicación inmediatamente seguida de la línea de código que transfiere el control programa la rutina que maneja los eventos del grid (3640)

```
3640 IF EVENT.CODE$="N" THEN IF NOTICE.OBJTYPE%=107 THEN GOSUB GML_M; CONTINUE

3650 IF EVENT.CODE$="N" AND NOTICE.OBJTYPE%>18 AND NOTICE.OBJTYPE%<21
       THEN GOSUB LUM_LOOK_UP
            ELSE IF EVENT.CODE$<>"m" AND GML.LUM_FLAG% THEN GOSUB LUM_EVENT
```

LUM_LOOK_UP Subrutina:
En la subrutina LUM_LOOK_UP , el Id del evento es localizado en GML_LUM.EVENT_ID% y el contexto del evento es localizado in variables GML_LUM.EVENT_CONTEXT%. El look-up manager es entonces llamado para ejecutar la llamada al GML para encontrar el índice del look-up control identificado por el ID evento y contexto.

```
7620 LUM_LOOK_UP:
```

```
7630 REM +----------------------+
7640 REM !  FIND LOOK-UP INDEX  !
7650 REM +----------------------+
7660 GML_LUM.EVENT_ID%=EVENT.ID%
7670 GML_LUM.EVENT_CONTEXT%=EVENT.CONTEXT%
7680 CALL "GML::LUM_FIND"
```

Si después de la ejecución de la llamada GML, la variable GML_LUM.LU_SET% contiene un 0 (el control no es un look-up de tipo control) la subrutina termina bifurcando para el RETURN la línea 8010.

```
7690 IF GML_LUM.LU_SET%=0 THEN GOTO 8010
```

En la siguiente porción de la subrutina, la variable de plantilla NOTICE.CODE% es usada con el verbo SWITCH para determinar como manejar el evento (ver VPRO/5 documentación relacionada al List Button y List Edit Notify Events).

Cuando un evento es generado por un list opened (CASE 1) no necesita acciones, y la subrutina termina.

```
7700 SWITCH NOTICE.CODE%
7710 REM +---------------+
7720 REM !  LIST OPENED  !
7730 REM +---------------+
7740 CASE 1
7750 BREAK
```

Cuando un ítem es seleccionado, el valor de la variable GML_LUM.LU_SET% es localizada dentro de la variable GML.SET%. GML es llamado para preparar la plantilla GML_TPL$, y el GML LUM es llamado para traer el texto del look-up control que generó el evento. GML LUM localiza el texto dentro de la plantilla GML_TPL$ variable GML_TPL.COL$[n]. El [n] es la suma de los valores en las variables GML_LUM.COL%[GML_LUM.LU_SET%] y GML_LUM.LOC%[GML_LUM.LU_SET%]. El número de la fila del evento es localizado en la variable GML_TPL.ROW_N%. El procedimiento POPULATE es entonces ejecutado para actualizar el texto en la celda del grid principal.

```
7760 REM +-----------------+
7770 REM !  ITEM SELECTED  !
7780 REM +-----------------+
7790 CASE 2
7800 GML.SET%=GML_LUM.LU_SET%
7810 CALL "GML::TPL_PREP"
7820 CALL "GML::LUM_GET_TEXT"
7830 CALL "GML::POPULATE"
7840 BREAK
```

Cuando el evento es generado por un list closed (CASE 3) o una cancelación del proceso de selección (CASE 4), GML LUM es llamado para ocultar el control.

```
7850 REM +---------------+
7860 REM !  LIST CLOSED  !
7870 REM +---------------+
7880 CASE 3
7890 REM +-------------------------------+
7900 REM !  SELECTION PROCESS CANCELLED  !
7910 REM +-------------------------------+
7920 CASE 4
7930 CALL "GML::LUM_HIDE"
7940 BREAK
```

Cuando el evento es generado por el cambio de selección (CASE 5) no son acciones necesarias, y la subrutina termina.

```
7950 REM +----------------------+
7960 REM !  SELECTION CHANGED  !
7970 REM +----------------------+
7980 CASE 5
7990 BREAK
8000 SWEND
8010 RETURN
```

Listando la subrutina LUM_LOOK_UP

```
7620 LUM_LOOK_UP:
7630 REM +-----------------------+
7640 REM !  FIND LOOK-UP INDEX  !
7650 REM +-----------------------+
7660 GML_LUM.EVENT_ID%=EVENT.ID%
7670 GML_LUM.EVENT_CONTEXT%=EVENT.CONTEXT%
7680 CALL "GML::LUM_FIND"
7690 IF GML_LUM.LU_SET%=0 THEN GOTO 8010
7700 SWITCH NOTICE.CODE%
7710 REM +---------------+
7720 REM !  LIST OPENED  !
7730 REM +---------------+
7740 CASE 1
7750 BREAK
7760 REM +-----------------+
7770 REM !  ITEM SELECTED  !
7780 REM +-----------------+
7790 CASE 2
7800 GML.SET%=GML_LUM.LU_SET%
7810 CALL "GML::TPL_PREP"
7820 CALL "GML::LUM_GET_TEXT"
7830 CALL "GML::POPULATE"
7840 BREAK
7850 REM +---------------+
7860 REM !  LIST CLOSED  !
7870 REM +---------------+
7880 CASE 3
7890 REM +-------------------------------+
7900 REM !  SELECTION PROCESS CANCELLED  !
7910 REM +-------------------------------+
```

```
7920 CASE 4
7930 CALL "GML::LUM_HIDE"
7940 BREAK
7950 REM +----------------------+
7960 REM !  SELECTION CHANGED  !
7970 REM +----------------------+
7980 CASE 5
7990 BREAK
8000 SWEND
8010 RETURN
```

LUM_EVENT Subroutine:

```
3650 IF EVENT.CODE$="N" AND NOTICE.OBJTYPE%>18 AND NOTICE.OBJTYPE%<21
       THEN GOSUB LUM_LOOK_UP
            ELSE IF EVENT.CODE$<>"m" AND GML.LUM_FLAG% THEN GOSUB LUM_EVENT
```

El control del programa es transferido a la subrutina LUM_EVENT cuando el valor de la variable GML.LUM_FLAG% no es igual a 0, y otro evento en un list control o evento del mouse ha ocurrido. En la subrutina LUM_EVENT, el Id evento es localizado en la variable GML_LUM.EVENT_CONTEXT%. EL look-up manager es entonces llamado ejecutando un llamado al GML para determinar si el look-up control perdió el enfoque y ocultó el control como lo requirió.

```
7420 LUM_EVENT:
7430 GML_LUM.EVENT_ID%=EVENT.ID%
7440 GML_LUM.EVENT_CONTEXT%=EVENT.CONTEXT%
7450 CALL "GML::LUM_EVENT"
7460 RETURN
```

GML LUM PROCEDIMIENTO: Index

Page	LUM PROCEDIMIENTO Name
50	INIT_LUM_TPL
50	LUM_DISPLAY
51	LUM_EVENT
52	LUM_FIND
53	LUM_GET_TEXT
53	LUM_HIDE
53	LUM_MODE
54	LUM_START

GML LUM PROCEDIMIENTO INIT_LUM_TPL

Sintaxis
CALL "GML::INIT_LUM_TPL"

Propósito:
Inicializar (preparar) la plantilla GML_LUM$ para ser usada. La inicialización fija la variable para pre—establecer valores los cuales son necesarios para iniciar el GML look-up manager.

Notas:
La plantilla GML_LUM$ es inicializada basada en un valor el cual ha sido previamente asignado a la plantilla GML_INIT$ variable GML_INIT.LUM_QTY% durante el start-up de la aplicación.

Ejemplo:

```
2400 CALL "GML:: INIT_LUM_TPL"
```

GML LUM PROCEDIMIENTO LUM_DISPLAY

Sintaxis
CALL "GML::LUM_DISPLAY"

Propósito:
Sobreponer un look-up control en un grid de celda principal.

Nota:
Cuando este procedimiento es ejecutado, la posición del control será determinada por los valores de las variables contenidas en la plantilla GML_LUM$ que será inicializada al start-up de la aplicación.

Ejemplo 1:
En el siguiente ejemplo GML.NOTIFY% 18 ha sido usado como el evento que activa (muestra) el lookup control. Si hay un control asociado con la información contenida en la plantilla GML_TPL$, ese control es desplegado. Sin embargo, si no hay lookup control asociado con la información GML_TPL$, el llamado a LUM_DISPLAY no tendrá efecto.

```
7170 REM +------------------+
7180 REM !  RIGHT CLICK     !
7190 REM !  GML.NOTIFY%=18  !
7200 REM +------------------+
7210 CASE 18
7220 SWITCH GML_TPL.MEMBER%
7230 CASE 1
7240 IF GML_TPL.FLAG%=1 THEN CALL "GML::LUM_DISPLAY"
7250 BREAK
```

GML LUM PROCEDIMIENTO LUM_DISPLAY

Sintaxis
CALL "GML::LUM_DISPLAY"

Opción:
Cuando GML_TPL.DEF_FLAG% es fijado a 0, el lookup control es posicionado y desplegado basado en parámetros reportados en el grid notify events 15 o 18 (ejemplo 1).

Si GML_TPL.DEF_FLAG% es fijada a 1, el lookup control será posicionado y desplegado basado en el número de columna y fila contenido en la plantilla GML_TPL$ variables GML_TPL.COL_N% y GML_TPL.ROW_N% (ejemplo 2)

Si el lookup control es un listbutton y GML_TPL.OPTION% es fijada a 0, el LUM manager ejecutará un LISTSEL basado en el valor en GML_TPL.ARG% cuando el control es desplegado.

Si el lookup control es un list control y GML_TPL.OPTION% es fijado a 1 y el texto es localizado dentro del GML_TPL.TEXT$, el texto será localizado en el listedit área seleccionada (edit) cuando el control es desplegado.

Fijando la variable GML_TPL.SUP% a 1 será habilitado el LUM auto-oculto para el lookup control, mientras fijando GML_TPL.SUP% a 2 dehabilitará el auto-oculto. EL modo LUM auto-oculto puede ser cambiado ejecutando el procedimiento LUM_MODE.

Ejemplo 2:
Un lookup es desplegado en el grid de las celdas principal localizado en la columna 2, fila 8 (el look up ha sido previamente definido como asociado con la columna 2):

```
4310 GML.SET%=1
4320 CALL "GML::TPL_PREP"
4330 GML_TPL.COL_N%=2
4340 GML_TPL.ROW_N%=8
4350 GML_TPL.DEF_FLAG%=1
4360 CALL "GML::LUM_DISPLAY"
```

GML LUM PROCEDIMIENTO LUM_EVENT

Sintaxis
CALL "GML::LUM_EVENT"

Propósito:
Determinar si el lookup control perdió el enfoque, oculta el control como sea requerido.

Nota:
Este procedimiento es ejecutado cuando el valor de variable GML.LUM_FLAG% no es igual a 0, y otro evento de un list control o evento mouse (code=m) ocurrió. El Id evento es localizado en GML_LUM.EVENT_ID% y el contexto del evento es localizado en la variable GML_LUM.EVENT_CONTEXT%. El lookup manager es entonces llamado ejecutando el GML para determinar si el lookup control perdió el enfoque y oculta el control como es requerido.

Ejemplo:

```
2400 GML_LUM.EVENT_ID%=EVENT.ID%
2410 GML_LUM.EVENT_CONTEXT%=EVENT.CONTEXT%
2420 CALL "GML::LUM_EVENT"
```

GML LUM PROCEDIMIENTO LUM_FIND

Sintaxis
CALL "GML::LUM_FIND"

Propósito:
Localizar el número de índice del lookup control.

Nota:
Cuando un evento ocurre en una lista tipo control y la variable GML.LUM_FLAG% no es igual a 0 (un lookup control ha sido desplegado), el lookup manager es llamado ejecutando el procedimiento LUM_FIND para encontrar el índice del lookup control identificado por él ID evento y contexto. Este procedimiento es también usado para localizar el número de índice del lookup control.

Ejemplo 1:
Encuentra un lookup índice basado en un evento.

```
2400 GML_LUM.EVENT_ID%=EVENT.ID%
2410 GML_LUM.EVENT_CONTEXT%=EVENT.CONTEXT%
2420 CALL "GML::LUM_FIND"
```

Ejemplo 2:
Encuentra un lookup índice basado en un control ID y contexto específico.

```
2400 GML_LUM.EVENT_ID%=122
2410 GML_LUM.EVENT_CONTEXT%=1
2420 CALL "GML::LUM_FIND"
```

GML LUM PROCEDIMIENTO LUM_GET_TEXT

Sintaxis
CALL "GML::LUM_GET_TEXT"

Propósito:
Recupera texto desde un lookup control.

Notas:
Cuando un ítem es seleccionado desde un lookp control, el valor de la variable GML_LUM.LU_SET% (lookup número índice obtenido con el procedimiento LUM_FIND) es localizada dentro de la variable GML.SET%. GML es llamado para preparar la plantilla GML_TPL$, entonces el procedimiento LUM_GET_TEXT es ejecutado para obtener él testo del lookup control que generó el evento, ubica el texto dentro de la plantilla GML_TPL$ y prepara la plantilla para ser usada con el procedimiento POPULATE.

```
2400 GML.SET%=GML_LUM.LU_SET%
2410 CALL "GML::TPL_PREP"
2420 CALL "GML::LUM_GET_TEXT"
2430 CALL "GML::POPULATE"
```

GML LUM PROCEDIMIENTO LUM_HIDE

Sintaxis
CALL "GML::LUM_HIDE"

Propósito:
Ocultar lookup controles.

Notas:
Cuando la aplicación esta manejando el ocultamiento del lookup control (auto-oculta es deshabilitado) el procedimiento LUM_HIDE es usada por la aplicación para ocultar el look up control.

Ejemplo:

```
2400 CALL "GML::LUM_HIDE
```

GML LUM PROCEDIMIENTO LUM_MODE

Sintaxis
CALL "GML::LUM_MODE"

Propósito:
Habilitar o deshabilitar el auto-oculto en el lookup manager.

Nota:
El LUM normalmente maneja el ocultamiento de lookup control basado en eventos que ocurren dentro de la aplicación. Si es requerido que la aplicación maneje el ocultamiento del lookup, entonces el procedimiento LUM_MODE, es ejecutado para cambiar el modo de auto-oculto a no auto-oculto.

Este producto es asociado con cada lookup control individual. Esto es, si hay dos controles, el control 1 podría tener un modo auto-oculto, mientras que el control 2 no es auto-oculto.

El valor en GML_TPL.ARG% identifica él ID lookup control, el valor en GML.SET% indica en cual set del grid el lookup control es asignado a, y el valor en GML_TPL.OPTION% determina el modo del lookup control.

Un GML_TPL.OPTION% valor de 1 indica que el LUM maneja el auto-oculto del control, mientras que un valor de 2 indica que la aplicación manejará el ocultamiento del control.

Ejemplo 1:
Ocultando un control manejado por la aplicación.

```
4300 GML.SET%=1
4310 CALL "GML::TPL_PREP"
4320 GML_TPL.OPTION%=2
4330 GML_TPL.ARG%=122
4340 CALL "GML::LUM_MODE"
```

Ejemplo 2:
El ocultamiento del control de manejado inverso por el LUM.

```
4300 GML.SET%=1
4310 CALL "GML::TPL_PREP"
4320 GML_TPL.OPTION%=1
4330 GML_TPL.ARG%=122
4340 CALL "GML::LUM_MODE"
```

GML LUM PROCEDIMIENTO LUM_START

Sintaxis
CALL "GML::LUM_START"

Propósito:
Inicia el GML lookup manager (LUM).

Nota:
Este procedimiento es ejecutado en la aplicación startup, siguiendo la ejecución del procedimiento INIT_LUM_TPL y cargando la información del lookup dentro de la plantilla GML_LUM$.

Ejemplo:

```
2400 CALL"GML::LUM_START"
```

Asociando un archivo con un grid control

EL GML puede ser configurado para asociar mkeyed archivos de registro a filas dentro del grid control principal. Esta relación, referida como "attaching" un archivo, es una en las cuales el GML maneja la lectura del canal de datos, mientras la aplicación individual formatear los datos del registro y pasa los datos formateados devuelta al GML para desplegarlos dentro del grid.

En la configuración del archivo attached, las teclas keyboard Arrow Up/Down, Page Up/Down, and Tab son usadas para navegar a través del grid / archivo asociado. El scroll vertical también provee navegación, con la funcionalidad basada en una de los siguientes modos de archivos attached.

Archivo Attached: Modo 1:
En la inicialización (attaching del archivo) el scroll vertical es porporcional a número de registros en el archivo entero. Si los registros son agregados o borrados después que el archivo es attached, el scroll vertical proporcionado es actualizado al nuevo número de registros dentro del archivo.

Archivo Attached: Modo 2:
Este modo es usado para llevar a través de una porción de un archivo. En la inicialización la aplicación provee la tecla start para el GML el cual es usado para identificar la fila 1 del grid. Debido al hecho que una tecla final no es provista el scroll bar vertical no puede ser proporcionado, como consecuencia limita su uso para mover el grid de la fila (registro) hacia atrás o adelante a través del archivo cada vez que se hace un click. Como los registros son leídos en dirección delantera la aplicación indica, a través de una bandera el GML, cuando el último registro para la porción de archivo ha sido desplegado. La lectura del archivo es entonces limitada a los registros entre la tecla start y la designación del último registro. La aplicación podrá cambiar el último registro designando cada vez que el archivo es leído en dirección delantera.

Archivo Attached: Modo 3:
El modo 3 es también usado para leer a través de una parte del archivo. En la inicialización la aplicación provee la tecla start y la tecla final a ser usadas como marcadores por el GML cuando lea adelante y atrás a través del archivo. Desde el inicio y final son provistos, el scroll vertical es proporcionado basado en el número de registros entre esos 2 puntos en el archivo. En el modo 3 la proporción es establecida en la inicialización y no puede ser cambiada, es por eso, que la aplicación debe controlar (prevenir) adiciones al archivo entre él las teclas de inicio y fin designadas.

Características Modo de archivo Attached

MODO ARCHIVO ATTACHED	Conducta barra scroll vertical	Tecla start requerida	Tecla final requerida	Rando registros	Permite Agregar y remover registros
1	Auto Proporcionada	No	No	Archivo completo	Sí
2	Not Proporcionada	Si	No	Porción de archivo	Si
3	Proporcionada	Si	Si	Porción de Archivo	No

Ejemplo de programa que captura datos sin estar Attached

A continuación un ejemplo de cómo incorporar de una forma sencilla los elementos necesarios para programar una aplicación que contenga un Control Grid utilizando la librería de administración del Grid GML que incorpora el Visual Pro/5.

Lo primero es diseñar la interfaz (pantalla) que nos servirá para interactuar con el control grid, para eso ejecutemos los siguientes pasos:

1. Ponga en ejecusión el utilitario Resbuilder
   a. Cree un nuevo Form
      i. Dé las siguientes propiedades al form construido
      ii. Form_Id = 101
      iii. Name = frm_Compras
      iv. Current Units = Semi-chars
      v. X Position = 20
      vi. Y Position = 62
      vii. Width = 356
      viii. Height = 235
      ix. Flags = Close box, Enter as Tab,Keyboard Navigation

2. Cree los siguientes elementos dentro del Form:

   a. Static Text
      i. Control Id = 100
      ii. Name = lbl_Empresa
      iii. Text = Ferreteria El Clavo Torcido S.A.
      iv. X Position = 107
      v. Y Position = 9
      vi. Width = 134
      vii. Height = 12
      viii. Font = MS Sans Serif Style: Bold Size: 12.
      ix. Justification = Center

   b. Static Text
      i. Control Id = 109
      ii. Name = lbl_Programa
      iii. Text = Entrada de Compras
      iv. X Position = 131
      v. Y Position = 24
      vi. Width = 74
      vii. Height = 12
      viii. Font = MS Sans Serif Style: Bold Size: 10.
      ix. Justification = Center

   c. InputE
      i. Control Id = 102
      ii. Name = Id_Codigo
      iii. Initial contents = Nada
      iv. X Position = 7
      v. Y Position = 55
      vi. Width = 41

vii. Height = 15
viii. Short cue = Digite un código de producto
ix. Max Length = 10

d. InputE
i. Control Id = 104
ii. Name = Id_Descripcion
iii. Initial contents = Nada
iv. X Position = 48
v. Y Position = 55
vi. Width = 149
vii. Height = 15
viii. Short cue = Digite la descripción del producto
ix. Max Length = 50

e. InputN
i. Control Id = 106
ii. Name = Id_Cantidad
iii. Initial value = Nada
iv. X Position = 197
v. Y Position = 55
vi. Width = 41
vii. Height = 15
viii. Short cue = Poner la cantidad
ix. Mask = ###,##0.00

f. InputN
i. Control Id = 108
ii. Name = Id_Costo
iii. Initial value = Nada
iv. X Position = 238
v. Y Position = 55
vi. Width = 50
vii. Height = 15
viii. Short cue = Debe poner el costo unitario
ix. Mask = #,###,##0.00

g. Grid
i. Control Id = 1001
ii. Name = Id_Grid
iii. Text = Nada
iv. X Position = 6
v. Y Position = 71
vi. Width = 345
vii. Height = 135
viii. Num Rows = 20
ix. Row height = 12
x. Num Columns = 5
xi. Row head = Des-chequeado
xii. Row head ID = 111
xiii. Row head width = 50
xiv. Col Head = Chequeado
xv. Col Head Id = 1002
xvi. Col head height = 12

xvii. Col Lines = Chequeado
xviii. Row Lines = Des-Chequeado
xix. Max cols = 5
xx. Vert scroll bar = Chequeado
xxi. Column prop:

1.	width: 40	title: Código
2.	width: 150	title: Descripción
3.	width: 40	title: Cantidad
4.	width: 50	title: Costo unitario
5.	width: 53	title: T o t a l

h. Tool Button
i. Control Id = 113
ii. Name = Id_Nuevo
iii. X Position = 4
iv. Y Position = 2
v. Width = 12
vi. Height = 15
vii. Short cue = Iniciar nuevo documento
viii. Face Type = Bitmap
ix. Bitmap File = \Basis\tools\guibuild\new.bmp

i. Tool Button
i. Control Id = 114
ii. Name = Id_Print
iii. X Position = 16
iv. Y Position = 2
v. Width = 12
vi. Height = 15
vii. Short cue = Presentación preliminar
viii. Face Type = Bitmap
ix. Bitmap File = \Basis\tools\guibuild\print.bmp

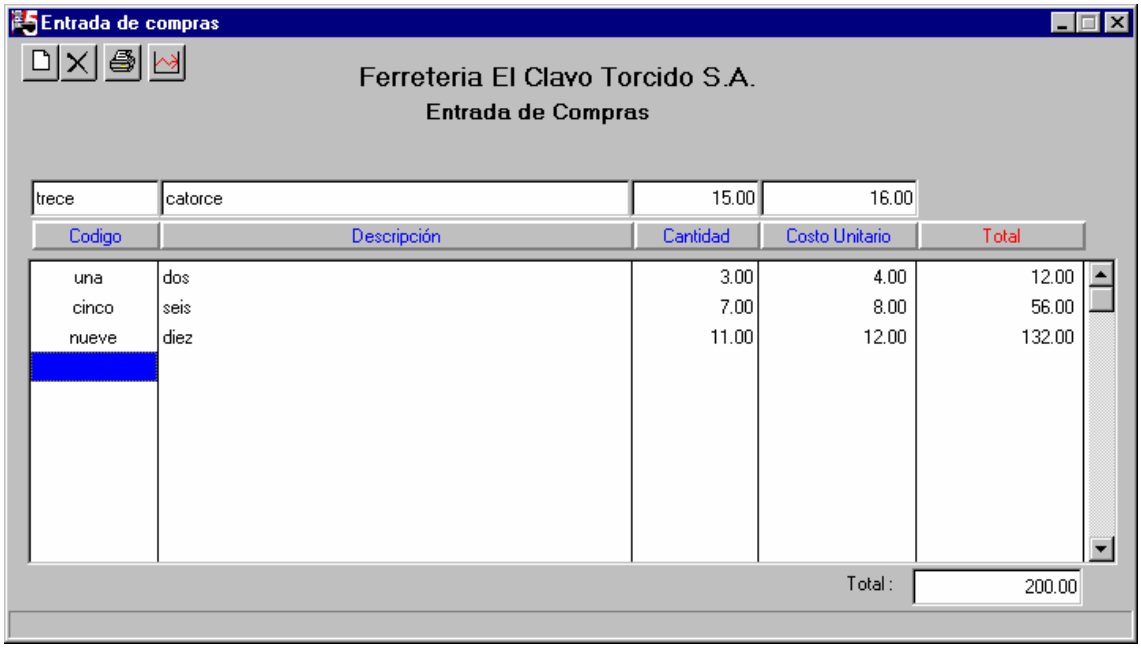
j. Tool Button
i. Control Id = 115
ii. Name = Id_Eliminar
iii. X Position = 28
iv. Y Position = 2
v. Width = 12
vi. Height = 15
vii. Short cue = Elimina datos en línea de ingreso de datos.
viii. Face Type = Bitmap
ix. Bitmap File = \basis\tools\guibuild\delcode.bmp

k. Tool Button
i. Control Id = 117
ii. Name = Id_Salir
iii. X Position = 40
iv. Y Position = 2
v. Width = 12
vi. Height = 15
vii. Short cue = Retornar al menú inicial.
viii. Face Type = Bitmap
ix. Bitmap File = \Basis\tools\guibuild\run.bmp

l. Static Text
   i. Control Id = 1006
   ii. Name = lbl_Total
   iii. Text = Total :
   iv. X Position = 265
   v. Y Position = 209
   vi. Width = 20
   vii. Height = 12

m. InputN
   i. Control Id = 1007
   ii. Name = Id_Total
   iii. Initial value = Nada
   iv. X Position = 286
   v. Y Position = 207
   vi. Width = 54
   vii. Height = 15
   viii. Back Color = Blanco
   ix. Mask = ###,###,##0.00
   x. Read Only = Chequeado

n. Pongamos un Status Bar a la forma. Para eso, en las propiedades de la forma, marque **Yes** en Has status bar y luego de estos valores a las propiedades de la barra de estado que le aparecerá:
   i. Control Id = 1008
   ii. Name = Id_StatusBar
   iii. Text = Nada



Una vez concluida la pantalla, guardemos el archivo con el nombre **Compras.brc** en el directorio `/Basis/cursovp5/` y pasemos a desarrollar el programa en Guibuilder. Para eso abra el GuiBuilder y presione el botón New GuiBuilder File, dele el nombre **Compras.gbf** y cuando le pregunte si desea crear un Resbuilder file dele **"NO"** y en su lugar seleccione el archivo **Compras.brc** que acabamos de crear.

Ahora en el menú principal seleccione la opción Program / Make Grid Code , esta opción crea dos subrutinas necesarias para manejar el control Grid, más adelante trabajaremos con estas subrutinas.

Del botón de lista Object seleccione la opción Initialization Code y digite lo siguiente en el area de edición:

```
Gosub Define_Constantes
Gosub Grid_Init
```

Del botón de lista object seleccione --- New Subroutine/Function –

Digite el Nombre: Define Constantes

Luego agregue el siguiente código:

```
Define_Constantes:

dim frm_screen$:fngb__template$(gb__win_id$)

Id_Codigo          =Num(Fattr(frm_screen$,"Id_Codigo","ID"))
Id_Descripcion     =Num(Fattr(frm_screen$,"Id_Descripcion","ID"))
Id_Cantidad        =Num(Fattr(frm_screen$,"Id_Cantidad","ID"))
Id_Costo           =Num(Fattr(frm_screen$,"Id_Costo","ID"))
Id_Grid            =Num(Fattr(frm_screen$,"Id_Grid","ID"))
Id_Total           =Num(Fattr(frm_screen$,"Id_Total","ID"))


Get_Text           =1
Get_Value          =2
Info_Icon          =64
Fatal_Icon         =16
Question_Icon      =32
Yes_No_Buttons     =4
No                 =7
true               =1
false              =0
m$                 ="####,###,##0.00-"

controlInputN      =104
controlInputE      =105
```

```
Dim
Datos$:"Codigo:C(6),Descripcion:C(50),Cantidad:N(10),Costo:N(15),Total:
N(15)"

Dim
Encabezado$:"Codigo:C(7),Descripcion:C(49),Cantidad:C(10),Costo:C(18),T
otal:C(16)"

Encabezado.Codigo$      ="Codigo "
Encabezado.Descripcion$ ="Descripcion"
Encabezado.Cantidad$    =pad("Cantidad",8,"R")
Encabezado.Costo$       =pad("Costo",18,"R")
Encabezado.Total$       =pad("Total",16,"R")

Return
```

Luego seleccionemos del botón de lista Object la Subrutina Grid Initialize
Y modifiquemos y agreguemos el siguiente código:

```
rem ' ------------------------------------------------------------------
rem ' Grid Initialize
rem ' ------------------------------------------------------------------

grid_init:

gml_qty%=1
call "gml::start"

rem ' copy this block for each grid set
gml_init.main_id%[1]=1001
gml_init.col_header_id%[1]=1002
gml_init.row_header_id%[1]=0
gml_init.sysgui%[1]=gb__sysgui
gml_init.context%[1]=fngb__context(gb__win_id$)

rem ' initialize gml
call
"gml_i",gb__sysgui,gml_init$,gml$,gml_gm$[all],gml_gc$[all],gml_gr$[all
],gml_set$

rem ' initialize gml template
gml.set%=1
call "gml::tpl_prep"

rem '--- Titulos de Columna ---
Gml_tpl.member%   =2
Gml_tpl.col$[1]   ="Codigo"
Gml_tpl.col$[2]   ="Descripción"
Gml_tpl.col$[3]   ="Cantidad"
Gml_tpl.col$[4]   ="Costo Unitario"
Gml_tpl.col$[5]   ="Total"
Gml_tpl.t_color$[1] =gml.blue$
Gml_tpl.t_color$[2] =gml.blue$
```

```
Gml_tpl.t_color$[3] =gml.blue$
Gml_tpl.t_color$[4] =gml.blue$
Gml_tpl.t_color$[5] =gml.red$
call "Gml::Update"

rem '--- Mascaras de Presentacion de Datos ---
call "Gml::tpl_prep"
Gml_tpl.Mask$[3]="##0.00"
Gml_tpl.Mask$[4]=m$
Gml_tpl.Mask$[5]=m$
call "Gml::set_out_mask"

rem '--- Alineamiento de los Datos ---
call "Gml::tpl_prep"
Gml_tpl.row_n%=-1

for i%=1 to 5

    switch i%
      case 1;  rem Codigo
               Gml_tpl.align%[i%]  =Gml.centered%
               Gml_tpl.e_mode%[i%] =true
               break

       case 2;  rem Descripcion
                Gml_tpl.align%[i%]  =Gml.left%
                Gml_tpl.e_mode%[i%] =true
                break

       case 3; rem Cantidad
        case 4; rem Costo
        case 5; rem Total
              Gml_tpl.align%[i%]   =Gml.right%
               Gml_tpl.e_mode%[i%] =true
              break
    swend

next i%

call "Gml::put"
return
```

Luego seleccionemos la subrutina Grid Event
Y agreguemos el siguiente código:

```
rem ' ---------------------------------------------------------------
rem ' Grid Event
rem ' ---------------------------------------------------------------

grid_event:

call "gml_m",gb__sysgui,gb__event$,gb__notice$,
:  gml_set$,gml_tpl$,gml$,gml_gm$[all],gml_gc$[all],gml_gr$[all]
```

```
switch gml.notify
   case 6; rem end edit
      switch gml_tpl.flag
         case 1; rem enter
            call "gml::end_edit"
            break
      swend
   break
   case 9; rem start edit
      switch gml_tpl.flag
         case 3; rem start edit (double click)
                Total               =num(gml_tpl.col$[5])
                TotalG
=Num(Ctrl(gb__sysgui,Id_Total,Get_Text))
                TotalG              =TotalG-Total
                Print (gb__sysgui)'title'(Id_Total,str(TotalG)),
:                                   'title'(Id_Codigo,gml_tpl.col$[1]),
:
'title'(Id_Descripcion,gml_tpl.col$[2]),
:                                   'title'(Id_Cantidad,gml_tpl.col$[3]),
:                                   'title'(Id_Costo,gml_tpl.col$[4])
                if gml_tpl.row_stat%=0 then
:               row%=row%-1
                call "gml::delete_row"


         case 9; rem start edit (enter)
            call "gml::start_edit"
            break
      swend
break
swend

return
```

Luego seleccionemos del botón de lista Object --- New Subroutine/Function ---
Y digitemos el nombre Limpiar Pantalla y digitemos el siguiente código.

```
rem ' ------------------------------------------------------------------
rem ' Limpiar Pantalla
rem ' ------------------------------------------------------------------

Limpiar_Pantalla:

Print (gb__sysgui)'clrtype'(controlInputN,controlInputE)

gml.set%=1
call "gml::tpl_prep"
call "gml::clear"

row%=0
```

```
Print (gb__sysgui)'focus'(Id_Codigo)

Return
```

Seleccionemos del botón de lista Object --- New Subroutine/Function ---
Y digitemos el nombre Grabar y digitemos el siguiente código:

```
rem ' ------------------------------------------------------------------
rem ' Grabar
rem ' ------------------------------------------------------------------

Grabar:

Datos.Codigo$       =Ctrl(gb__sysgui,Id_Codigo,Get_Text)
Datos.Descripcion$  =Ctrl(gb__sysgui,Id_Descripcion,Get_Text)
Datos.Cantidad      =Num(Ctrl(gb__sysgui,Id_Cantidad,Get_Text))
Datos.Costo         =Num(Ctrl(gb__sysgui,Id_Costo,Get_Text))

if cvs(Datos.Codigo$,3)="" then
:      Resp=msgbox("No ha digitado el Codigo",Info_Icon,"Información");
:       Return

if cvs(Datos.Descripcion$,3)="" then
:      Resp=msgbox("No ha digitado la Descripcion
",Info_Icon,"Información");
:       Return

if Datos.Cantidad=0 then
:      Resp=msgbox("No ha digitado la
Cantidad",Info_Icon,"Información");
:       Return

if Datos.Costo=0 then
:      Resp=msgbox("No ha digitado el costo",Info_Icon,"Información");
:       Return

Total   =(Datos.Cantidad*Datos.Costo)

row%=row%+1
gml_tpl.col$[1]=Datos.Codigo$
gml_tpl.col$[2]=Datos.Descripcion$
gml_tpl.col$[3]=str(Datos.Cantidad)
gml_tpl.col$[4]=str(Datos.Costo)
gml_tpl.col$[5]=str(Total)
gml_tpl.row_n%=row%
call "gml::put"

call "gml::tpl_prep"
call "gml::refresh"

rem '//Desplegar el Total General

TotalG  =Num(Ctrl(gb__sysgui,Id_Total,Get_Text))
```

```
TotalG  =TotalG+Total
Print (gb__sysgui)'title'(Id_Total,str(TotalG))

rem '//---------------------------

Print
(gb__sysgui)'clrtitle'(Id_Codigo,Id_Descripcion,Id_Cantidad,Id_Costo),
:                 'focus'(Id_Codigo)

return
```

Seleccionemos del botón de lista Object --- New Subroutine/Function ---
Y digitemos el nombre Imprimir Reporte y digitemos el siguiente código:

```
rem ' ---------------------------------------------------------------
rem ' Imprimir Reporte
rem ' ---------------------------------------------------------------

Imprimir_Reporte:

Print (Pchan,err=Cerrar_Impresora)'cp',"Ferreteria El Clavo Torcido
S.A.",@(54),"Sistema de Inventario",@(114),"Pagina :",1
Print (Pchan,err=Cerrar_Impresora)Pad("Entrada de Compras ",132,"C"),
:@(114),"Fecha  :",Date(0:"%Dz/%Mz/%Yl")
Print (Pchan,err=Cerrar_Impresora)@(114),"Hora   :",Date(0:"%h:%m:%p")
Print (Pchan,err=Cerrar_Impresora)fill(132,"-")
Print (Pchan,err=Cerrar_Impresora)Encabezado$
Print (Pchan,err=Cerrar_Impresora)fill(132,"-")

Total=0

for fila%=1 to 20

gml.set%=1
call "gml::tpl_prep"
gml_tpl.row_n%=fila%
call "gml::fetch"

if num(gml_tpl.col$[5])=0 then
:           Goto Siguiente_Fila

Dim
Datos$:"Codigo:C(6),Descripcion:C(50),Cantidad:N(10),Costo:N(15),Total:
N(15)"

Datos.Codigo$       =gml_tpl.col$[1]
Datos.Descripcion$  =gml_tpl.col$[2]
Datos.Cantidad$     =str(num(gml_tpl.col$[3]):"##0.00")
Datos.Costo$        =str(num(gml_tpl.col$[4]):m$)
Datos.Total$        =str(num(gml_tpl.col$[5]):m$)

Print (pchan,err=Cerrar_Impresora)Datos.Codigo$,"
",Datos.Descripcion$," :",Datos.Cantidad$," ",Datos.Costo$,"
",Datos.Total$
```

```
Total=Total+num(gml_tpl.col$[5])

Siguiente_Fila:
next fila%

if Total<>0 then
:     Print (pchan,err=Cerrar_Impresora)fill(132,"-");
:     Print (pchan,err=Cerrar_Impresora)"Total "+fill(79),Total:m$

Cerrar_Impresora:
Close (pchan)
Return
```

Seleccionemos del botón de lista Object --- New Subroutine/Function ---
Y digitemos el nombre Seccion de Mensajes y digitemos el siguiente código:

```
rem ' ------------------------------------------------------------------
rem ' Seccion de Mensajes
rem ' ------------------------------------------------------------------

Seccion_de_Mensajes:

Impresora_Ocupada:
Resp=msgbox("La Impresora no esta lista",Info_Icon,"Información")
return
```

Del botón de lista Object seleccione Form 101 frm_Compras, luego del botón de lista Control seleccione INPUTN 108 Id_Costo y después del botón de lista Event seleccione Control Lost Focus y adicione el siguiente código:

```
Gosub Grabar
```

Luego sigamos el mismo procedimiento para agregar más código de evento a los siguientes controles, del botón de lista Control seleccione el control Tool Button 113 Id_Nuevo y agregue lo siguiente al evento Tool Button Pushed:

```
Gosub Limpiar_Pantalla
```

Del botón de lista Control seleccione el control Tool Button 114 Id_Print agregue lo siguiente al evento Tool Button Pushed:

```
Pchan=unt
Open (Pchan,mode="PREVIEW,COLS=132",err=Impresora_Ocupada)"LP"
Goto Imprimir_Reporte
```

Del botón de lista Control seleccione el control Tool Button 115 Id_Eliminar agregue lo siguiente al evento Tool Button Pushed:

```
Print (gb__sysgui)'clrtitle'(Id_Codigo,Id_Descripcion,Id_Cantidad,Id_Costo)
```

Del botón de lista Control seleccione el control Tool Button 117 Id_Salir agregue lo siguiente al evento Tool Button Pushed:

```
gb__eoj = true
```

Del botón de lista Control seleccione Form 101 frm_Compras agregue lo siguiente al evento Window Closed:

```
gb__eoj = true
```

Para Finalizar del botón de lista Object selecione End of Job Code y agregue el siguiente código:

```
rem ' ------------------------------------------------------------
rem ' EOJ
rem ' ------------------------------------------------------------

release
```

Compilemos nuestro programa presionando el botón Build Program y luego corramos el programa presionando el botón Run Program.