

gserver v5.02

Desarrollado por gmatic

2007, gmatic, *electrónica de máquinas industriales.*



Electrónica de máquinas industriales
Electronics for industrial machinery

Joan Garcia Castillo
Francesc Casajoana, 63
08280 Calaf, Barcelona
Tel: 93 868 01 72 y 686 63 81 28

Acerca de gServer

Esta aplicación fue desarrollada por D. Juan García Castillo, trabajador autónomo dedicado al desarrollo de productos electrónicos y automatización industrial con el nombre comercial **gmatic**. Inicialmente se proyectó para uso privado y fue adaptada para monitorizar datos de proceso de una industria cerámica pero más tarde, se decidió distribuirla de forma libre y gratuita a todos aquellos desarrolladores electrónicos y para el uso docente, (no siendo gratuita para un usuario final industrial).

“gServer”, podríamos definirlo como un servidor de comunicaciones entre un PC o red de PCs y un PLC o red de PLCs

“gServer”, simplifica el código de nuestras aplicaciones ya que no debemos preocuparnos de la comunicación con los dispositivos conectados, y utilizándolo, podemos crear nuestras propias aplicaciones cliente con total libertad, incluso podríamos crear nuestro propio SCADA creando sus controles y librerías de código, o para usar en nuestras aplicaciones.

“gServer” ofrece gran seguridad y consistencia debido a que las aplicaciones cliente están en proceso con gServer, esto permite un alto rendimiento y seguridad para su uso con automatización remota. Se ha probado utilizando el subsistema DCOM (Distributed Component Object Model) que Windows incorpora de serie desde la aparición de Windows 95.

Usando automatización remota con gServer podremos ejecutar y desarrollar aplicaciones que interactúen con un proceso industrial desde cualquier PC conectado en una red local o Internet.

El medio de comunicación entre gServer y el PLC o red de PLCs es a través de RS232, de este modo, gServer se convierte en la opción más barata del mercado, en el caso de grandes distancias físicas entre dispositivos, añadiremos pasarelas RS422. Actualmente, se está trabajando en la implementación el medio de comunicación *ethernet* para futuras versiones de gServer.

La interface de gServer permite realizar ajustes para adaptar su funcionamiento a nuestras necesidades, todo ello de forma sencillísima.

Con gServer, podemos utilizar para escribir nuestras propias aplicaciones cliente cualquier lenguaje de programación para Windows (MFC y COM). Al instalar gServer, queda en su disco duro el directorio de instalación en la carpeta *Examples* diversos ejemplos para su uso con Visual Basic y Microsoft Excel y también se ha logrado integrar con éxito len aplicaciones para el producto LabVIEW de National Instruments mediante VI's o instrumentos virtuales.

El creador de gServer, apuesta por la libre distribución de gServer como medio de difusión del producto, valorando cualquier opinión o sugerencia que el usuario quiera hacer sobre esta aplicación. Pruebe gServer y si tiene alguna duda, contacte con gmatic@telefonica.net.

1 El objeto gConnection

Al crear este objeto, Windows ejecuta la aplicación gServer.exe. Y el objeto *gConnection* creado presenta una interface o lista de propiedades y procedimientos que nos permitirá interactuar con la aplicación gServer.

Propiedad	Status	Devuelve un entero a tipo de enumeración que indica el estado de la conexión.
Función	addItem	Devuelve un objeto de tipo <i>gItem</i> que es enlazado a una Dirección de PLC.
Propiedad	ConnectedSince	Devuelve un tipo Fecha/hora que indica la hora de inicio de la conexión.
Propiedad	ConnectionName	Valor de tipo string que define un nombre identificativo para la conexión.
Función	GetgServerSettingInterface	Devuelve un objeto. <i>GetgServerSettingInterface</i> que ofrece una interface que permite el ajuste del funcionamiento de gServer.
Procedimiento	Disconnect	Desconecta y destruye el Objeto <i>gConnection</i> .

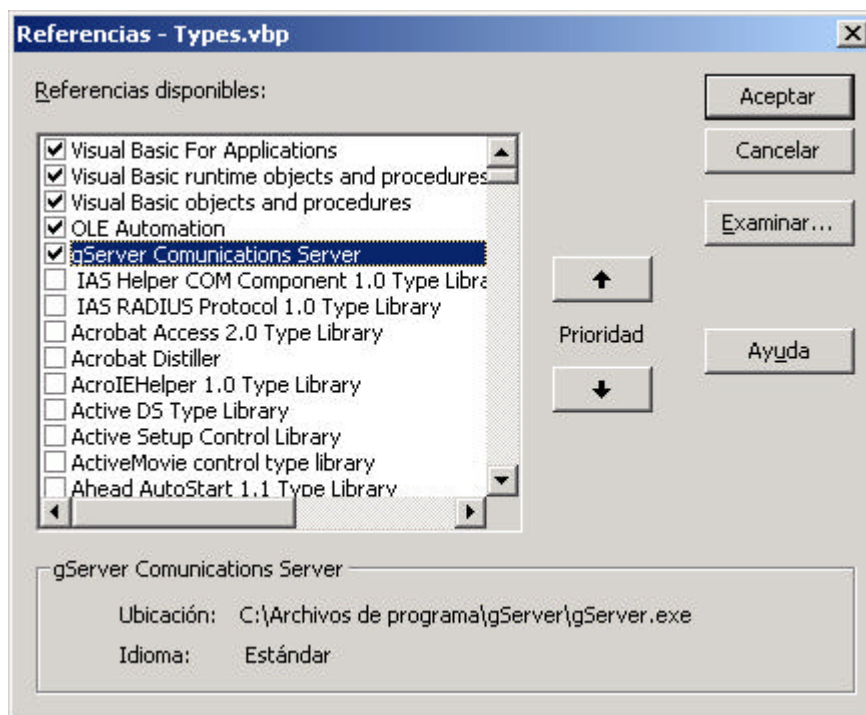
Lista de los métodos que componen la interface *gServer.gConnection*. (Todos los conceptos mostrados en esta lista serán explicados a lo largo de este libro).

La clase *gConnection* es de uso múltiple y se pueden establecer varias instancias de esta clase pudiéndose crear varios objetos *gConnection* desde una o varias aplicaciones cliente. Esto, ofrece grandes ventajas que más adelante se explicarán en este libro.

1.1 Crear un objeto gConnection

Hay varias formas de crear un objeto *gConnection* desde una aplicación cliente. Pero una cosa es imponderable, deberemos hacer referencia desde nuestra aplicación al componente *gServer* . Por ejemplo, si deseamos escribir código desde un editor de Visual Basic, deberemos crear un proyecto y después...

- Hacer desplegar el menú “Herramientas”
- Hacer clic en “Referencias...”
- Nos aparecerá una ventana que contiene la lista de componentes y librerías que se utilizarán en nuestro proyecto.



Buscar en la lista la entrada “**gServer Communications Server**” y seleccionarlo, acto seguido, hacer clic en “Aceptar” para validar nuestra selección.

Ahora, nuestra aplicación ya tiene referenciado el componente *gServer*, y podremos utilizarlo.

En el editor de Visual Basic, debemos declarar y dimensionar las variables de objeto donde será referenciada la conexión.

Escribiremos en el módulo del formulario el siguiente código.

```
Option explicit  
Dim MyConnection as gConnection
```

....Resto del código...

Así, ya tenemos dimensionada la variable de objeto *MyConnection* como objeto *gConnection*.

A continuación escribiremos el código siguiente para que se ejecute cuando de cargue el formulario de nuestra aplicación.

```
Private Sub Form1_Load ()  
Set MyConnection = CreateObject ("gServer.gConnection")  
  
....Resto del código...
```

Después de ejecutar esta línea, la variable de objeto *MyConnection* ya estará establecida y podremos utilizar sus métodos y procedimientos, necesarios para trabajar con gServer.

Aunque no queremos profundizar demasiado en las técnicas y posibilidades para programar nuestra aplicación cliente, debemos decir que también podríamos utilizar automatización remota indicando el nombre del servidor donde queremos ejecutar gServer, la sintaxis que deberíamos escribir es:

```
Private Sub Form1_Load ()  
Set MyConnection = CreateObject ("gServer.gConnection", "NombreMaquinaRemota")  
  
....Resto del código...
```

Nota: Si se omite el parámetro opcional 'ServerName' de la instrucción CreateObject (Class as string, optional ServerName as string) se toma como ServerName la máquina local donde se ejecuta la aplicación cliente. Se ha probado gServer utilizando automatización remota mediante DCOM con gran éxito.

1.2 La propiedad *gConnection.ConnectionName*.

Esta propiedad permite leer y escribir un nombre identificativo de tipo string para la conexión.

Public Property ConnectionName As String

Sintaxis escritura: gConnection.ConnectionName="Nombre Conexión"

Sintaxis Lectura: sNombreConexion=gConnection.ConnectionName

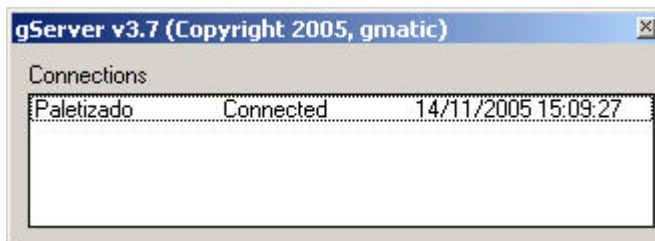
Por ejemplo, si añadimos el siguiente código:

```
Private Sub Form1_Load ()  
Set MyConnection = CreateObject ("gServer.gConnection",)  
MyConnection.ConnectionName = "Paletizador 1"  
  
....Resto del código...
```

El valor de la propiedad es el que se muestra en la lista de conexiones activas en la ventana del monitor de gServer.

Nota aclarativa. Esta propiedad solo es utilizada a modo de identificación por el usuario o aplicación cliente, y puede ser perfectamente omitida.

El nombre "Paletizador 1" se muestra en la lista de conexiones activas.



1.3 La Propiedad *gConnection.ConnectedSince*.

Esta propiedad solo de lectura permite obtener la fecha y hora en el formato del sistema operativo donde funcione la aplicación cliente.

Private propConnectedSince As Date

Sintaxis: dtHoraConexion=gConnection.ConnectedSince

Por ejemplo, imaginemos el siguiente código, donde queremos saber la última hora en que se ha iniciado la conexión.

```
Dim HoraInicioConexion as Date
HoraInicioConexion=MyConnection.ConnectedSince
...Resto de código...
```

1.4 La Función *gConnection.AddItem*.

Esta función devuelve un objeto del tipo *gItem*, y se le deben pasar como argumentos los datos que definen al *gItem*

Public Function AddItem(eDeviceType As EnumDeviceTypes, iNetNode As Integer, sItemAddress As String, eTypeOfData As EnumTypesOfData) As gItem

Donde:

eDevicetype : Enumeración de tipo de PLC (Ver lista EnumDeviceTypes al final de este libro) al que pertenezca el ítem que se quiere añadir.

INetNode: Valor de tipo entero donde indicamos el nodo de comunicación del PLC al que pertenezca el ítem que queremos añadir.

sItemAddress: Valor de tipo string donde indicamos la dirección (area de memoria válida) del ítem que queremos añadir.

eTypeOfData: Enumeración(EnumTypesOfData) del tipo de dato o formato en el que queremos expresar el valor del ítem que queremos añadir, UBCD, UINT, BOOL, etc...

Enumeración *EnumTypesOfData*

BOOL_	0	Tipo de datos booleano rango 0 a 1.
-------	---	-------------------------------------

UBCD_	1	Tipo de datos BCD sin signo, rango 0 a 9999.
UINT_	2	Tipo de datos entero sin signo rango 0 a 65535.
INT_	3	Tipo de datos entero con signo, rango -32768 a 32767.
UDBCD_	4	Tipo de datos BCD de tipo doble (32Bit), rango 0 a 99999999.
UDINT_	5	Tipo de datos entero doble (32Bit) sin signo, rango 0 a 4294967295.
DINT_	6	Tipo de datos entero doble (32Bit) con signo, rango -2147483648 a 2147483647.
REAL_	11	Tipo de datos en formato de coma flotante de precisión simple, rango -3.40282399999999E+38 a -1.40129899999999E-45 para números negativos y 1.40129899999999E-45 3.40282399999999E+38 para número positivos.

Lista de valores de la enumeración *EnumTypesOfData*.

El objeto *gItem* que nos devuelve esta función, hace referencia directamente a una dirección de memoria de un PLC conectado.

Por ejemplo: Sabemos que estamos conectados a un PLC del tipo CJ1G_CPU42H que tenemos configurado con el nodo host link = 0 que gobierna un paletizador. Y queremos tener acceso al valor del registro D05150 en formato UBCD que contiene en el PLC la posición del eje X del pórtico del paletizador y el estado del BIT H12.07 de dicho PLC que indica el estado abierto o cerrado de la electroválvula de la pinza de agarre de dicho paletizador.

Escribiremos el siguiente código en la inicialización de nuestra aplicación.

```
Dim ConnPaletizador as gConnection
Dim PosCoordenadaX as gItem
Dim EstadoPinza as gItem

Private Sub Form1_Load ()

    Set ConnPaletizador = CreateObject ("gServer.gConnection",)
    ConnPaletizador.ConnectionName = "Paletizador 1"
    Set PosCoordenadaX = ConnPaletizador. AddItem (CJ1G_CPU24H, 0, "D5150", UBCD_)
    Set EstadoPinza = ConnPaletizador. AddItem (CJ1G_CPU24H, 0, "H12.7", BOOL_)

    ....Resto del código...
```

A Partir de este momento, los Ítem definidos como *PosCoordenadaX* y *EstadoPinza* serán refrescados automáticamente por el componente *gServer* y nosotros podremos consultar cuando queramos el valor de las direcciones de PLC definidas llamando a las propiedades del objeto *gItem* de la siguiente forma.

Supongamos que queremos mostrar en un control label de nuestro formulario el valor de la coordenada del eje X del paletizador definida en la variable de objeto *PosCoordenadaX*, escribiremos el siguiente código.

```
...Codigo...
Label1.Caption = PosCoordenadaX. Value

...Resto del código...
Nota: la propiedad gItem.Value devuelve el valor en el formato previamente definido en el ítem.
```

Si queremos visualizar el valor de la coordenada X de dicho paletizador de una forma periódica, podemos incluir un control temporizador *Timer()* a nuestro formulario, también podemos visualizar el estado de la pinza.

```
...Codigo...
Private Sub Timer1_Timer()

    Label1.Caption=posCoordenadaX.Value & "mm"
    If EstadoPinza.Value =1 then
        Label2.Caption="Cerrada"
    Else :Label2.Caption="Abierta"
    End If

End Sub

...Resto del código...
```

El objeto *gItem* se explicará con más detalle a lo largo de este libro.

1.5 La Propiedad *gConnection.Status*.

Esta propiedad solo de lectura, permite obtener el estado de la conexión.

Public Property Status As EnumConnectionStatus
Sintaxis: eEstadoConexión=gConnection.Status

Cuando consultamos esta propiedad desde nuestra aplicación cliente, nos devuelve un tipo de dato entero relacionado a una enumeración.

Enumeración *EnumConnectionStatus*:

<i>Initialized</i>	=0
<i>Connected</i>	=1
<i>Disconnecting</i>	=2

El estado *initialized* es devuelto si se ha creado el objeto *gConnection* pero aún no se ha añadido ni creado ningún ítem (*gltem*).

El estado *Connected* es devuelto si se ha añadido algún ítem a dicha conexión.

El estado *Disconnecting* es devuelto si se ha llamado al procedimiento *gConnection.Disconnect* que destruye y establece a nothing al objeto *gConnection* mientras se está "desconectando".

Por ejemplo:

```
Dim ConnPaletizador as gConnection
Dim PosCoordenadaX as gltem
Dim EstadoPinza as gltem

Private Sub Form1_Load ()

    Set ConnPaletizador = CreateObject ("gServer.gConnection",)
    'Aquí el valor de ConnPaletizador.Status es Initialized

    ConnPaletizador.ConnectionName = "Paletizador 1"
    'Aquí el valor de ConnPaletizador.Status sigue siendo Initialized

    Set PosCoordenadaX = ConnPaletizador. AddItem (CJ1G_CPU24H, 0, "D5150", UBCD_)
    'Aquí el valor de ConnPaletizador.Status es Connected

    Set EstadoPinza = ConnPaletizador. AddItem (CJ1G_CPU24H, 0, "H12.7", BOOL_)
    'Aquí el valor de ConnPaletizador.Status sigue siendo Initialized
    ....Resto del código...
```

Cierre de nuestra aplicación cliente:

```
Private Sub Form1_Unload (Cancel as integer)
    ConnPaletizador.Disconnect
    'Aquí el valor de ConnPaletizador.Status es Disconnecting
    Set ConnPaletizador = Nothing
    Set PosCoordenadaX= Nothing
    Set EstadoPinza = Nothing
End Sub
```

Es muy importante que al final de la aplicación o cuando no se quieran refrescar más datos de una determinada conexión desconectemos la conexión, de esta forma, *gServer* dejará de consultar los ítems definidos en esa conexión y de esta forma se consigue mejor rendimiento en otras conexiones que puedan estar aún activas. (Recuerde que *gServer* admite varias conexiones de diversas aplicaciones simultáneamente).

1 El objeto gltem.

El objeto *gltem*, se crea mediante el retorno de la llamada a la función

`gConnection.AddItem()` de una conexión activa. Hemos visto antes como crear un objeto `gItem` llamado a la función `gConnection.AddItem`.

El objeto `gItem` ofrece las siguientes propiedades.

Propiedad	Value	Devuelve un número de tipo <code>double</code> que contiene el valor que contiene la dirección de PLC definida en la creación del ítem.
Propiedad	Status	Nos muestra el estado del ítem, en forma de entero representado por una enumeración.

Lista de los métodos que componen el objeto `gItem`. (Todos los conceptos mostrados en esta lista serán explicados a lo largo de este libro).

1.6 La propiedad `gItem.Value`.

Esta propiedad, es de lectura y escritura, y contiene el valor de la dirección de PLC que se especifica cuando se crea el objeto `gItem` mediante la llamada a `gConnection.AddItem`.

Public Property Value As Double

Sintaxis escritura: `gItem.Value= Valor o variable`

Sintaxis Lectura: `Variable=gItem.Value`

Nota: La propiedad `.Value` devuelve un dato de tipo `double` capaz de expresar el valor del ítem del PLC en el formato de datos deseado.

Por ejemplo, si escribimos el siguiente código:

```
Dim MyConnection as gConnection
Dim MyItem as gItem

Private Sub Form1_Load ()

    Set MyConnection = CreateObject ("gServer.gConnection",)

    Set MyItem = MyConnection.AddItem (CQM1H_CPU51, 0, "DM1030", UBCD)

    ....Resto del código...
```

Mediante este código creamos un ítem que está en un PLC de tipo `CQM1H_CPU51`, con el número de nodo de red `0`, que apunta al `DM1030` de dicho PLC y se leerá y escribirá en formato BCD sin signo.

Una vez creado el objeto `gItem`, no debemos preocuparnos de nada más, cada vez que consultemos la propiedad `.Value` a lo largo de nuestra aplicación cliente, leeremos el valor de esta dirección de memoria.

1.7 La propiedad *gItem.Status*.

Esta propiedad, es solo de lectura, y contiene un valor entero que viene definido por una enumeración de valores, siendo:

Sintaxis: *EstadoItem=gItem.Status*

EnumItemStatus:

Not_Initialized	= 0
Device_Value	= 10
On_Fly_Value	= 20
Conversion_INT_To_BCD_Error	= 30
Overflow_Value	= 40
Unable_To_Write_In_RUN_Mode	= 50
Comunications_Device_Error	= 60

Valor *Not_Initialized*, se devuelve este valor = 0 cuando el ítem aún no se ha refrescado por el motivo que sea. Cabe decir que si utilizamos gServer con un método de refresco de tipo *No_Determinate* (detallado en la sección métodos de refresco en el capítulo 3.5 de este libro), la propiedad *Item.Status* contendrá este valor hasta que efectuamos la primera o escritura lectura de su valor desde nuestra aplicación cliente mediante la propiedad *Item.Value*.

Valor *Device_Value*, se devuelve este valor =10 cuando el valor leído en la propiedad *.Value* es un valor directamente leído del PLC conectado, es decir, el dato está físicamente presente en la memoria del PLC, esto puede ser utilizado para comprobar si se ha efectuado un escritura de valor correctamente.

Valor *On_Fly_Value*, se devuelve este valor =20 despues de realizar una escritura de un nuevo valor para el ítem y mientras el valor espera en cola para ser escrito. (gServer no escribe los valores en el PLC de forma inmediata, ya que espera a terminar antes la tarea de comunicación).

Por ejemplo supongamos el siguiente código, donde queremos escribir en la dirección definida por el ítem *MyItem* el valor 1234.

```

...Código...
MyItem.Value = 1234

'En este momento el valor de la propiedad MyItem.Status es 1001 (On_Fly_Value)
'Si queremos esperar a que el ítem esté escrito en el PLC para asegurar la escritura podemos
'hacer...

Do While MyItem.Value<> 1234 and MyItem.Status <> Device_Value 'Device_Value=1000
  Do Events
  If MyItem.Value<> 1234 then
    ... (Mostrar error de escritura,)
  Exit Do
Loop

...Resto del código...

```

La consulta de esta propiedad *.Status* variará según se ajuste el método de determinación mediante la herramienta *gServerConfigurator.exe* que incluye *gServer*. (Esto se explicará mas adelante en este libro)

Valor *Conversion_INT_To_BCD_Error*, se devuelve este valor = 30 cuando el un ítem definido como tipo de dato *UBCD_* o *UDBCD*, contiene un valor que no se puede expresar en un objeto.

Imaginemos el siguiente caso:

Tenemos un ítem definido como *UBCD* y el valor contenido en la dirección de memoria del PLC de este ítem es = *#12AB*, el valor *#12AB*, no se puede representar en modo *BCD*, por lo tanto, se nos notificará mediante la propiedad *.Status* con el valor *Conversion_INT_To_BCD_Error = 30*

Valor *Overflow_Value*, se devuelve este valor = 40 cuando se intenta escribir un valor en el ítem que supera los límites inferior o superior para el tipo de datos definido en este ítem.

Imaginemos el siguiente caso:

Tenemos un ítem definido como *UBCD* y queremos escribir en el el valor *10567*, como el rango para datos *UBCD* es de 0 a 9999, cuando intentemos escribir este valor, se nos devolberá a través de la propiedad *.Status* el valor *Overflow_Value = 40*. Después de esto, el valor de la propiedad *.Value* seguirá indicando el valor físico existente en la memoria del PLC.

Valor *Unable_Write_in_RUN_Mode*, se devuelve este valor = 50 cuando se pretende escribir un ítem del PLC y este está funcionando en modo *RUN*, en algunos tipos de PLC, se permiten escribir valores en modo *RUN*, y esto no nos afectará, (para más información consulte el manual de referencia del PLC al cual se quiere conectar). Normalmente, para permitir escritura de datos a través de comunicación, el PLC debe estar en modo *MONITOR*, o también lógicamente en modo *PROGRAM* o *STOP*. Después de devolver este valor, el valor de la propiedad *.Value* de nuestro ítem, vuelve a ser el valor anterior contenido en la memoria del PLC.

Valor *Comunications_Device_Error*, se devuelve este valor = 60 cuando ha surgido un error de comunicación o de cualquier otra naturaleza tanto en la escritura como en la lectura de la propiedad ítem. (Por ejemplo, si desenchufamos el cable de comunicación *PC-PLC*, la propiedad *.Status* nos dará *Comunications_Device_Error*.

2 El objeto gServerSettingsInterface

Este objeto, se obtiene haciendo una llamada a la función *gConnection.GetgServerSettingsInterface*. Y una vez establecido nos ofrece una interface de métodos y propiedades que nos permitirá configurar algunos parámetros de funcionamiento de gServer.

gConnection.GetgServerSettingsInterface (opcional [Login] as string, opcional [Password] as string) as gServerSettingsInterface

2.1 Crear un objeto gServerSettingsInterface

En el editor de Visual Basic escribimos el siguiente código.

```
...Código...
Dim MyConnection as gConnection
Dim SETI as gServerSettingsInterface
Private Sub Form1_Load()
    Set MyConnection = CreateObject ("gServer.gConnection")
    Set SETI= MyConnection.GetgServerSettingsInterface ()
...Resto del código...
```

A partir de este momento, la variable de objeto llamada SETI en nuestra aplicación cliente, nos ofrece la interface con los métodos y propiedades que nos permitirán ajustar el funcionamiento de gServer a nuestras necesidades. A continuación se muestra la lista de métodos que componen la interface del objeto *gServerSettingsInterface*.

Propiedad	AvailablePorts	Devuelve un string que contiene la descripción de los puertos de comunicación disponibles en el PC (servidor en el caso)
Propiedad	CommunicationsPortName	Valor de tipo string que devuelve o establece el nombre del puerto del PC (servidor en el caso) que se utilizará para realizar las comunicaciones.
Propiedad	CommunicationsSettings	Valor de tipo string que devuelve o establece la configuración de velocidad, control de paridad, tamaño de byte y bits de stop para transmisiones RS232.
Propiedad	HandshakeMethod	Valor de tipo enumerado que

		devuelve o establece el método de control de acceso de los dispositivos a una red de PLCs.
Propiedad	DeterminatingMode	Valor de tipo enumerado que devuelve o establece el método de determinación que se utilizará para lectura y escritura de datos de la red de PLCs.
Función	GetSensitiveTimeout	Devuelve un valor de tipo long que contiene el tiempo de respuesta en comunicación de los dispositivos conectados (red de PLCs) en unidades de milisegundos.
Procedimiento	LoadSettingsFromFile	Procedimiento que hace que gServer cargue la configuración especificada en el archivo de configuración.
Propiedad	MaxTimeout	Propiedad que devuelve o establece el tiempo máximo de espera de respuesta para transmisiones RS232 . El dato es de tipo long en unidades de milisegundos.
Propiedad	PoolRate	Propiedad que devuelve o establece la periodicidad del flujo de comunicaciones de gServer con la red de PLCs conectados. Valor de tipo long en unidades de milisegundos.
Procedimiento	SaveSettingsToFile	Procedimiento que hace que gServer guarde la configuración actual en un archivo de configuración. Esto provocará que gServer inicie con la configuración especificada en dicho archivo.
Propiedad	ShowMonitor	Propiedad que devuelve o

establece si se visualizará la ventana del monitor de gServer.

Propiedad	UnloadWhenConnectionsEmpty	Propiedad que devuelve o establece si se descargará o no gServer de la memoria cuando no exista ninguna conexión activa.
-----------	----------------------------	--

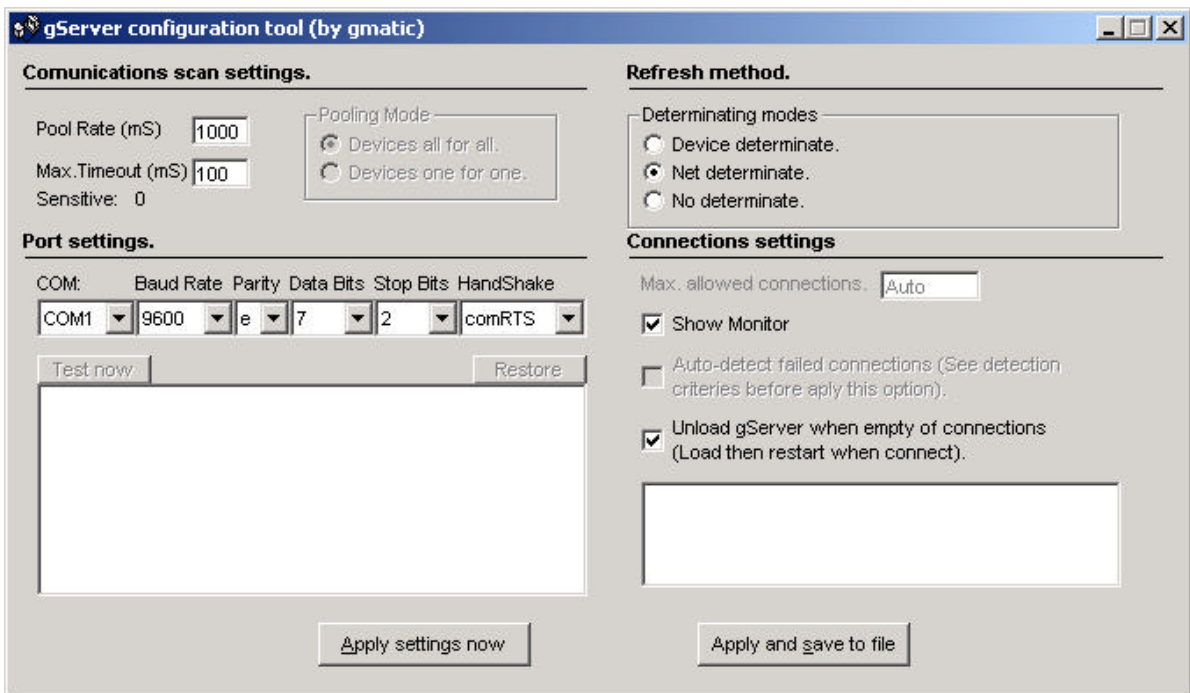
Lista de los métodos que componen el objeto *gServerSettingsInterface* (Todos los conceptos mostrados en esta lista serán explicados a lo largo de este libro).

Tal como podemos ver, gServer nos ofrece muchas posibilidades de ajuste para nuestras necesidades. No obstante, algunos parámetros de configuración son complejos y se debe estar profundamente familiarizado con el sistema de proceso de gServer para ajustarlos de forma óptima.

A continuación, haremos una descripción de la herramienta de configuración de gServer, la aplicación *gServerConfigurator.exe*.

2.2 La aplicación de configuración gServerConfigurator.

Esta aplicación, está ubicada en el directorio donde se halla instalado gServer, (normalmente *c:\WINDOWS\SYSTEM32*) pero observemos también que uno de los ejemplos para Visual Basic que incluye la aplicación es precisamente la herramienta *gServerConfigurator*, así pues, podemos ver el código fuente de esta aplicación y entender como se ajustan los parámetros de configuración tanto desde la herramienta *gServerConfigurator* como desde una aplicación cliente escrita por nosotros mismos. Aquí vemos la herramienta *gServerConfigurator*.



Ventana de la herramienta de configuración *gServerConfigurator.exe*

Acto seguido explicaremos cada uno de los parámetros que podemos utilizar para configurar el funcionamiento de gServer mediante la aplicación *gServerConfigurator*

2.3 Parámetros de ajuste de ciclo de comunicación.

Vemos dos casillas de texto:

Pool Rate (mS): El valor presente en esta casilla de texto indica el tiempo en milisegundos de cada periodo de lectura del PLC o red de PLCs, cuanto menor sea el tiempo de scan, con mayor celeridad se refrescarán los items añadidos a gServer.

NOTA: Cabe decir que a menor tiempo de scan o de PoolRate, mayor será el % de uso de CPU, esto debe tenerse en cuenta por el programador ya que un exceso de rendimiento del procesador implica el recalentamiento del hardware. Se recomienda llevar un régimen de rendimiento del procesador no superior al 40% en régimen continuo. Para más información consulte al proveedor del procesador del PC donde gServer es servidor)

Max.Timeout (mS): El valor presente en esta casilla de texto indica el tiempo en milisegundos que gServer se mantendrá en espera de respuesta durante la comunicación con los PLCs conectados.

Más abajo vemos:

Sensitive: A continuación se muestra un valor, este valor indica el tiempo real en milisegundos que gServer está en espera de respuesta durante la comunicación con los PLCs conectados. Se recomienda no impostar en el parámetro de configuración *Max.Timeout(ms)* un valor inferior al 50% por encima del valor de este dato. Por ejemplo...

Supongamos que:

El valor de *Sensitive* tiene un valor de 50, no se recomienda ajustar el parámetro *Max.Timeout(ms)* con un valor inferior a 75.

2.4 Parámetros de ajuste de puerto de comunicación.

Vemos 5 casillas de selección:

COM: El valor seleccionado en esta lista, contiene el nombre del dispositivo de comunicaciones que se utilizará para comunicar con el PLC o red de PLCs.

NOTA: Esta lista solo muestra los puertos disponibles para este fin instalados en el PC donde esté funcionando gServer como servidor.

Baud Rate: El valor seleccionado en esta lista, contiene la velocidad en baudios que se usará para la comunicación con el PLC o red de PLCs conectados.

NOTA: El valor establecido por defecto en PLCs de marca Omron es de 9600 baudios, pero pueden ajustarse otras velocidades consiguiendo así una mayor velocidad de comunicación. Esto se ha podido comprobar con gran éxito a velocidades de 57600 y 115200 baudios.

Parity: El valor seleccionado en esta lista, contiene el tipo de control de error sobre los telegramas de comunicación recibidos, control de paridad.

NOTA: El valor establecido por defecto en PLCs de marca Omron es de "E", (Even = Par),.

Data Bits: El valor seleccionado en esta lista, contiene el tamaño de los bytes que se reciben.

NOTA: El valor establecido por defecto en PLCs de marca Omron es 7 (7Bits), dado que se trata de telegramas ASCII es posible.

Stop Bits: El valor seleccionado en esta lista, contiene el número de bits de para en sincronismo de transmisión entre byte y byte.

NOTA: El valor establecido por defecto en PLCs de marca Omron es 2 (2Bits de parada).

2.5 Parámetros de control de flujo y handshaking.

Propiedad *HandshakeMethod*.

Vemos 4 tipos de control de flujo seleccionables

comNone (0): El dispositivo de comunicaciones utilizado no tiene control sobre el acceso protocolo de los dispositivos al bus de comunicación.

comXonXoff (1): Protocolo Xon/Xoff

comRTS (2): Protocolo RTS/CTS (Petición de envío / Preparado para enviar).

comRTSXonXoff (3): Ambos protocolos (RTS y XON/XOFF).

El protocolo utilizado por defecto para comunicar con redes de PLCs, es el comRTS (2), ya que permite controlar el acceso a varios dispositivos mediante las señales de permiso RTS y CTS. O siendo estas emuladas por conversores RS422

2.6 Parámetros de ajuste de método de refresco.

Vemos 3 opciones de refresco:

Device deteminate: Seleccionando este método de refresco los valores de items leídos mediante la llamada a la propiedad *gltem.Value* siempre son valores que se han leído directamente del PLC, todos los datos leídos son obtenidos al mismo tiempo, es decir existe un sincronismo común en el valor de los datos leídos del PLC. Este método de refresco puede ser útil para aplicaciones cliente que requieran alta fiabilidad de lectura o ejecutar acciones sobre el proceso.

Fijémonos en que si escribimos un valor en un ítem mediante la propiedad *.Value* y tenemos activado el método de refresco "Device Determinate", al leer de nuevo el valor del ítem, nos da el último valor que se leyó del PLC, (no el que hemos escrito), en cuanto el ítem se halla escrito con el valor que queremos y gServer vuelva a leer del PLC el valor de lectura de dicho ítem ya será el que nosotros hemos escrito.

NOTA: Este método de refresco es un poco más lento pero ofrece una altísima fiabilidad y seguridad.

Net deteminate: Este es el método que gServer utiliza por defecto cuando se instala por primera vez. Seleccionando este método de refresco los valores de ítem leídos mediante la llamada a la propiedad *gItem.Value* solo son obtenidos al mismo tiempo, es decir existe un sincronismo común en el valor siempre y cuando no se efectue una operación de escritura en alguno de los ítem,

Fijémonos en que si escribimos un valor en un ítem mediante la propiedad *.Value* y tenemos activado el método de refresco "Net Determinate", al leer de nuevo el valor del ítem, nos da el valor que se ha escrito, esto no significa que el valor ya se ha escrito, pero nosotros podemos creernoslo o efectuar un test sobre el ítem mediante la propiedad *gItem.Status*, que se mostrará en estado *On_Fly_Value=20* hasta que gServer lo haya leído directamente del PLC y la propiedad *gItem.Status* se ponga a *Device_Value= 10*.

NOTA: Este método es más rápido que "Device determinate" pero es necesario controlar la correcta escritura de valores en el ítem, si se desea un nivel de alta fiabilidad.

No deteminate: Seleccionando este método de refresco los valores de ítem leídos mediante la llamada a la propiedad *gItem.Value* no son obtenidos ni leídos al mismo tiempo y no existe ningún tipo de sincronismo común en el valor.

NOTA: Este método es mucho más rápido que los otros, es optimo para aplicaciones de solo visualización, el método de escritura es igual al modo "Net_Determinate".

Nota muy importante: gServer no debe usarse con fines de actuación directa sobre elementos mecánicos que puedan causar daños a personas o a seres vivos. Para estos fines, deberán utilizarse medios apropiados para este efecto y la ética de funcionamiento de maquinaria.

2.7 Parámetros de ajuste de conexión.

Vemos 3 casillas de verificación:

Show Monitor: Seleccionando esta casilla, permitiremos que se muestre o no la ventana del *Monitor de gServer*.

Auto_detect failed connections: Seleccionando esta casilla, permitiremos *gServer* desactive aquellos clientes de los que no se recibe respuesta ya sea debido a una terminación anormal en un módulo de la aplicación cliente o por una desconexión física de red o otros... (Esa opción no está incorporada aún en *gServer* pero se está trabajando en ello)

Unload gServer when empty of connections: Seleccionando esta casilla, permitiremos que Windows libere a *gServer* de la memoria (lo quite) cuando no tenga conexiones activas ni tareas de comunicación. Si no se selecciona esta casilla *gServer* permanecerá en memoria como proceso. "*gServer.exe*".

2.8 Botones de aplicación de cambios

Vemos 2 botones:

Apply settings now: Accionando este botón, activamos de forma inmediata los parámetros de configuración seleccionados. (Validación)

Apply and save to file: Accionando este botón, activamos de forma inmediata los parámetros de configuración seleccionados y estos datos serán además guardados en un fichero de configuración haciendo que gServer se ajuste a los parámetros deseados cada vez que inicie.