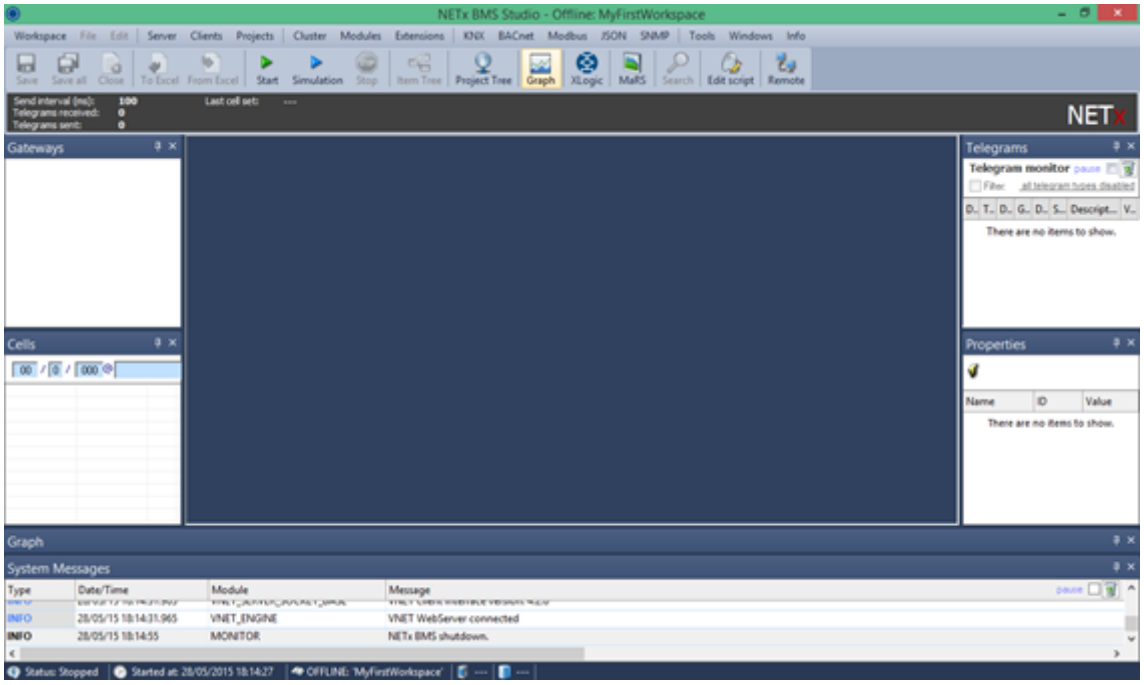
Tutorial para la comunicación TCP en el BMS Server

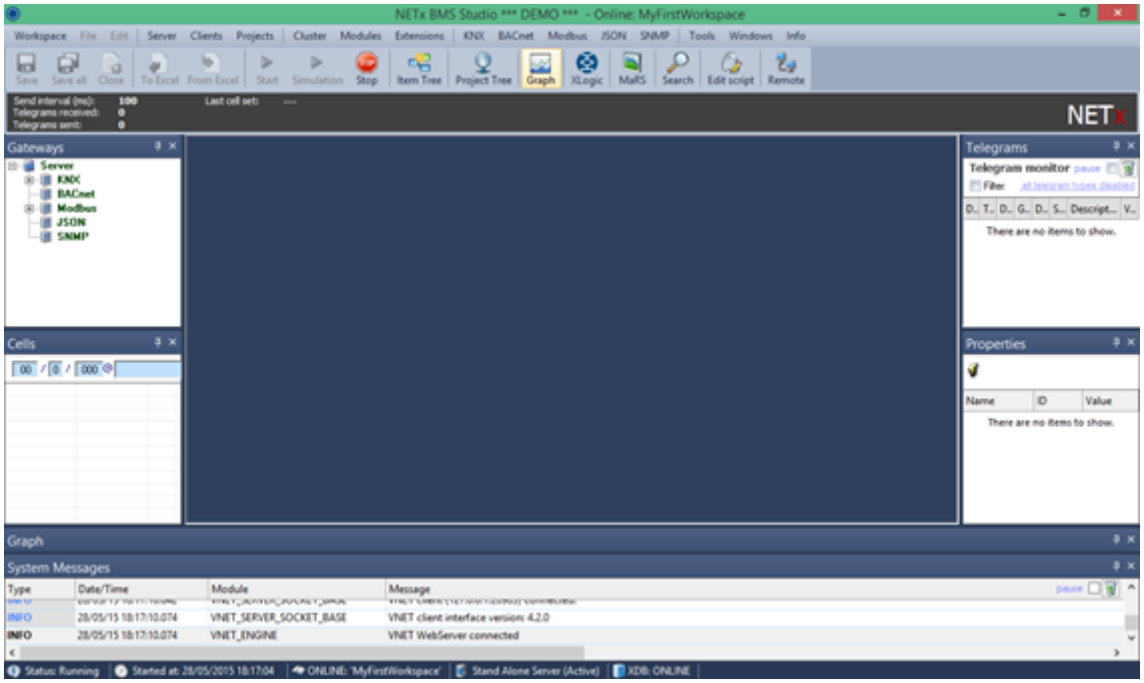
Contenido

1. Pasos iniciales para trabajar empleando conexión TCP

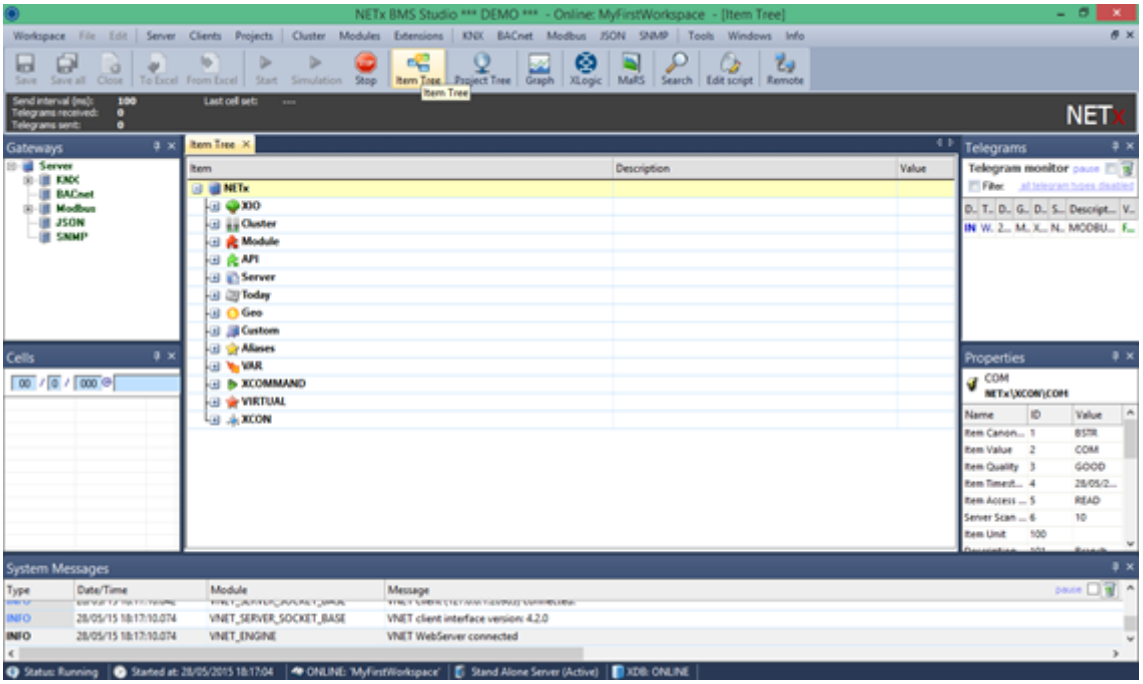
Tras iniciar el BMS Server nos encontraremos con la siguiente pantalla:



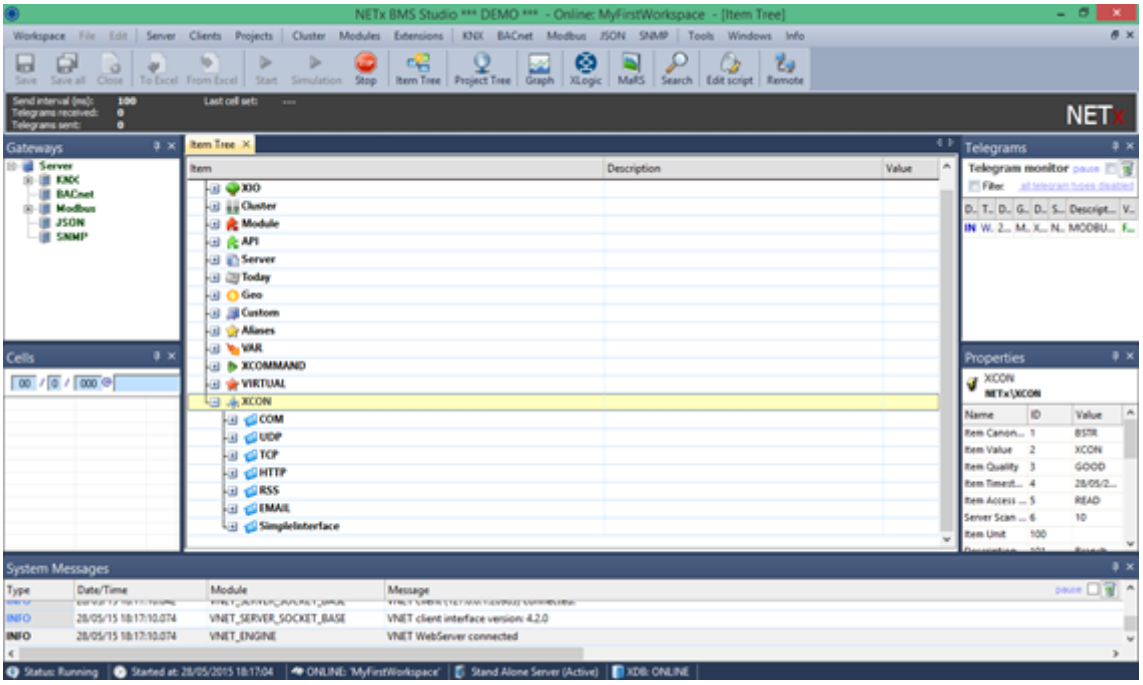
Clicamos sobre Workspace para seleccionar el deseado y tras abrirlo pulsamos en el botón “Start” para comenzar la simulación.



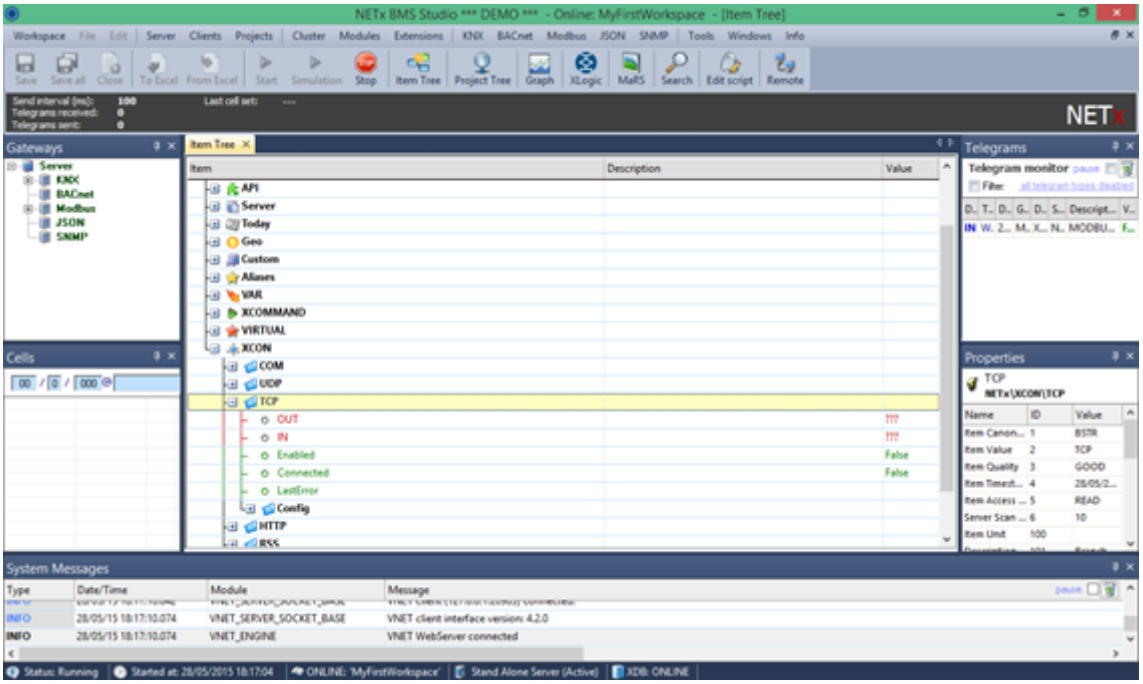
Tras abrir el Workspace seleccionado abrimos el “Item Tree” para tener acceso a todos los elementos de nuestro proyecto.



Desplegamos el menú de “XCOM” para mostrar las diferentes opciones proporcionadas por esta herramienta.

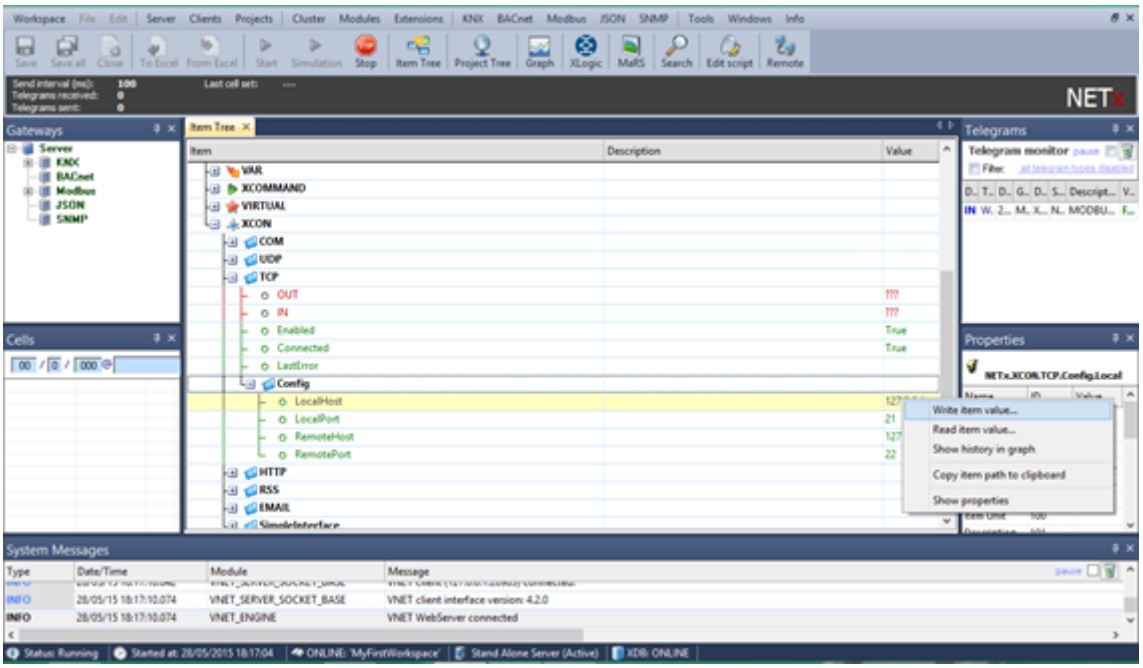


Desplegamos el menú de “TCP” para configurar los parámetros de conexión de forma adecuada.

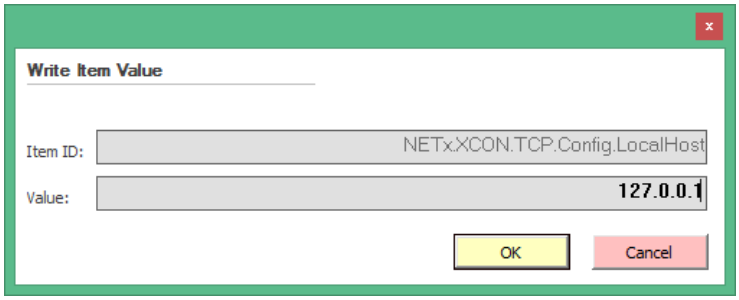


2. Configuración de la conexión TCP

Para configurar nuestra conexión desplegamos el menú “Config” del apartado “TCP” presente en el “Item Tree” e introducimos los parámetros de configuración apropiados clicando con el botón derecho en cada uno de ellos y seleccionando “Write item value…”



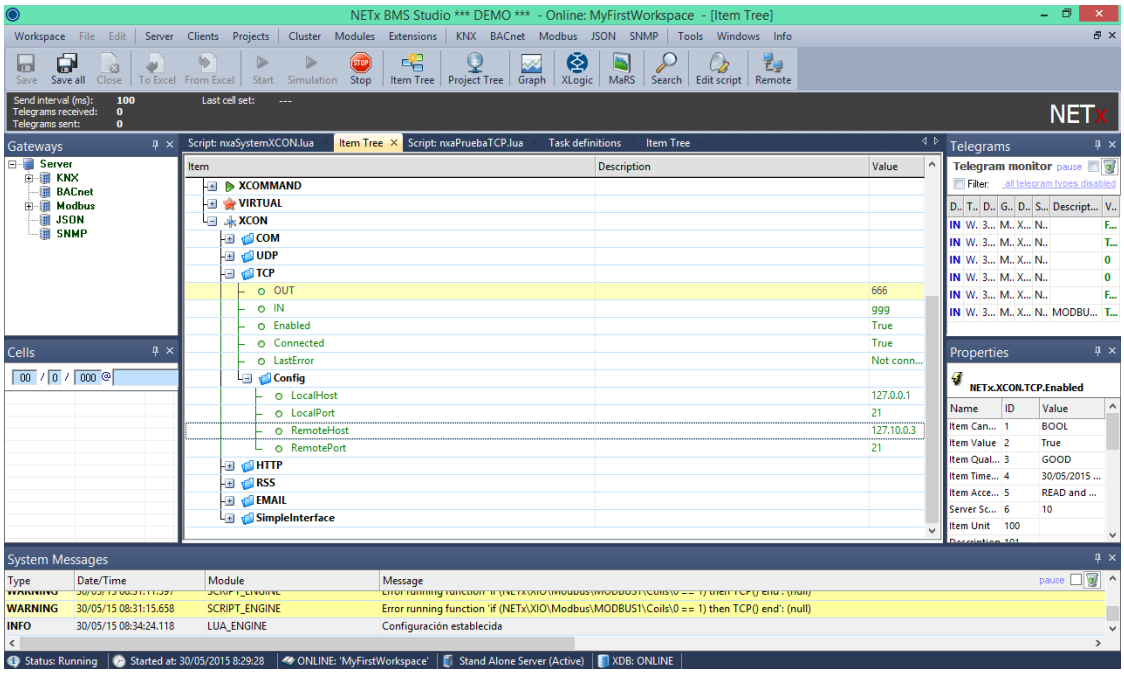
Tras realizar el paso anterior se nos abrirá una ventana en la aplicación con una apariencia similar a la siguiente:



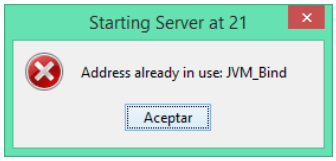
En nuestro caso configuramos la conexión de la siguiente forma:

- Dirección IP del LocalHost: 127.0.0.1
- Puerto de conexión LocalPort: 21
- Dirección IP del RemoteHost: 127.0.0.1
- Puerto de conexión RemotePort: 22

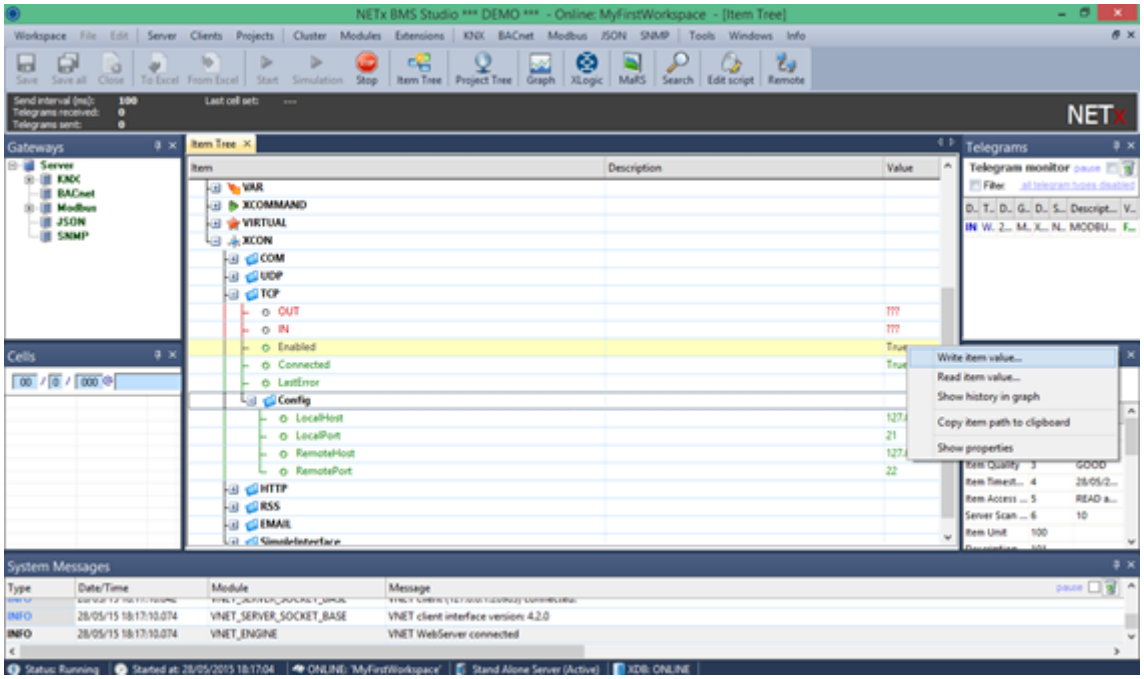
Nota: Las direcciones IP de LocalHost y RemoteHost pueden ser diferentes o iguales, ya que se emplean puertos diferentes. En el caso de que el BMS Server y la interfaz TCP emplearan el mismo puerto, no podrían utilizar la misma dirección IP, tendrían que ser diferentes.



Si se intenta emplear la misma IP, el programa mostrará un mensaje igual al presente en la siguiente imagen indicando que dicha dirección ya está en uso.

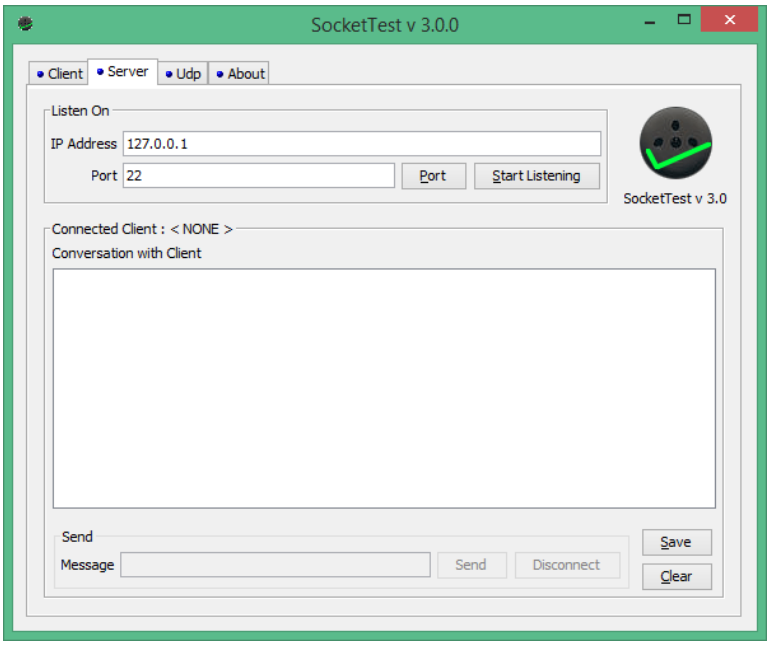


Tras realizar la configuración será necesario modificar el valor del parámetro “Enabled” para permitir la conexión. Esta modificación se realizará de la misma forma que los anteriores e introduciremos “1” o “True” para habilitar la conexión.

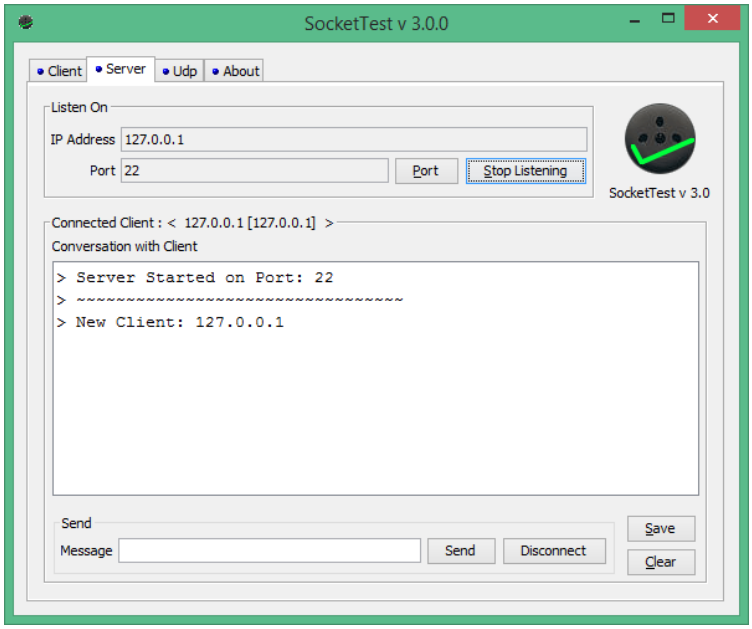


2.1. Comprobación de la conexión TCP

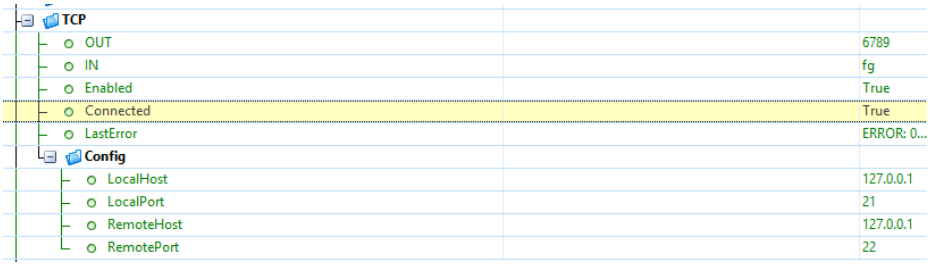
A fin de comprobar que la conexión se ha configurado y establecido correctamente, emplearemos un programa terminal que simulará la comunicación con el BMS Server, en nuestro caso emplearemos el programa "SocketTest".



Para que la comunicación se establezca deberemos configurar el programa terminal con la misma IP que introdujimos anteriormente en "RemoteHost" y con el mismo puerto de conexión que introdujimos en "RemotePort". Una vez configurada la conexión tanto en el BMS Server como en el terminal, presionamos el botón "Start Listening" sobre el SocketTest para verificar que la conexión se ha establecido correctamente.

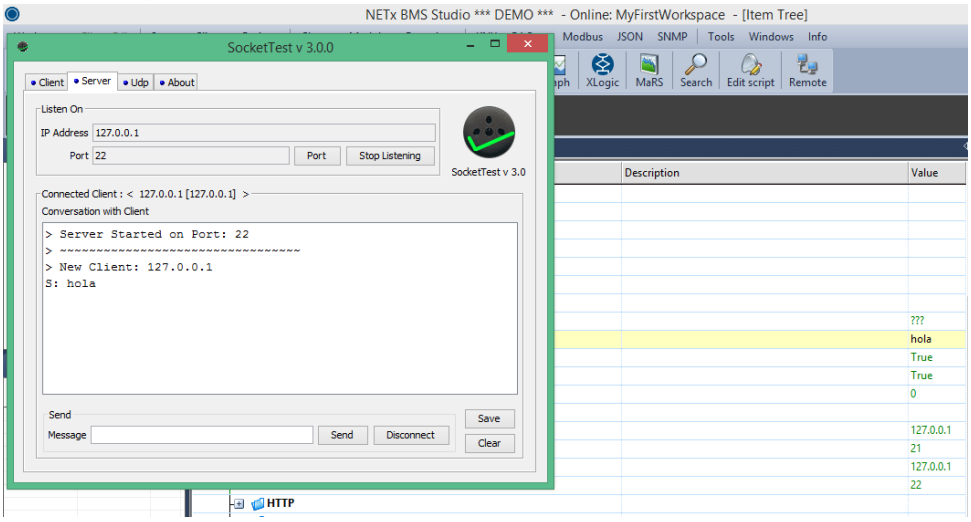


Si la comunicación se ha configurado y establecido de forma adecuada en el programa terminal visualizaremos el mensaje mostrado en la imagen superior, y en el BMS aparecerá la variable “Connected” con el valor “True” tal y como se aprecia en la siguiente imagen.

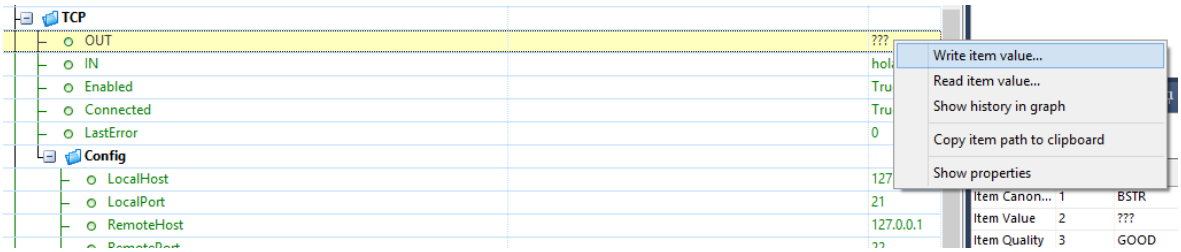


Una vez configurada, establecida y comprobada la conexión ya será posible el envío y recepción a través de ésta. Para enviar un mensaje escribiremos el mensaje deseado en “Send / Message” en el SocketTest y presionamos en el botón “Send”. En la siguiente imagen puede apreciarse el envío y recepción de la cadena de caracteres “hola”.

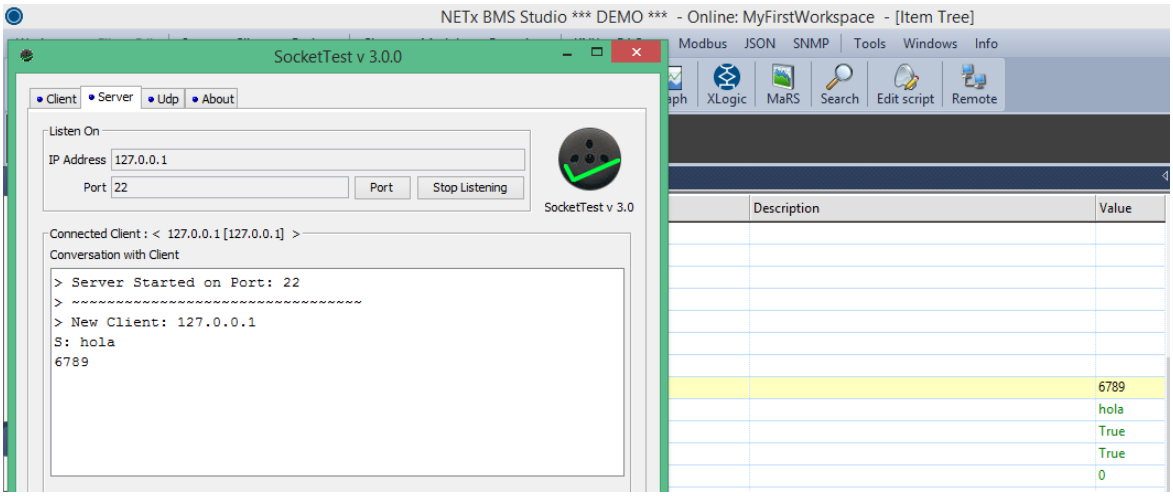
2.2. Ejemplo de comunicación TCP



La conexión TCP es una conexión bidireccional, por lo que también es posible enviar mensajes desde el BMS al terminal; para ello nos iremos al elemento “OUT” dentro del desplegable “TCP” y escribiremos el mensaje que deseamos enviar de la misma forma que hemos escrito anteriormente los parámetros de configuración.



En nuestro caso hemos enviado el valor numérico 6789 tal y como puede apreciarse en la siguiente captura de pantalla:



3. Parámetros de la Configuración TCP

Como se ha reflejado antes en las imágenes, para la comunicación TCP se utilizan los siguientes parámetros:

3.1. OUT

Este parámetro es el encargado de enviar los datos introducidos en el BNS Server empleando la comunicación TCP, es decir, es el buffer de bytes empleado para el envío por parte del BMS Server. Sus características son las siguientes:

- **Data Type (Tipo de Dato).** Es el tipo de datos que soporta, en este caso, una cadena de caracteres.
- **Default value (Valor por Defecto).** No tiene ningún valor por defecto.
- **Access Rights (Modos de Accesos).** Indica en qué modos podemos acceder al parámetro OUT. Los modos de este parámetros son lectura y escritura (Read and Write).
- **Standard Path (Ruta Estándar).** Es la ruta del Ítem del parámetro OUT; para poderla emplear, por ejemplo, con los comandos de la programación LUA. La ruta es NETx.XCON.<TCP>.OUT.

3.2. IN

Este parámetro es el encargado de enviar los datos introducidos mediante la comunicación TCP, es decir, es el buffer de bytes empleado para la recepción de datos por parte del BMS Server. Sus características son las siguientes:

- **Data Type (Tipo de Dato).** Es el tipo de dato que soporta, en este caso, una cadena de caracteres.
- **Default value (Valor por Defecto).** No tiene ningún valor por defecto.
- **Access Rights (Modos de Accesos).** Indica en qué modos podemos acceder al parámetro IN. El único modo de acceso a este parámetro es de lectura (Read).
- **Standard Path (Ruta Estándar).** Es la ruta del Ítem del parámetro IN, para poderla emplear, por ejemplo, con los comandos de la programación LUA. La ruta es NETx.XCON.<TCP>.IN.

3.3. Enabled

Este parámetro es el encargado de establecer la conexión del BMS Server mediante TCP. Es responsable de la creación de una conexión o desconexión de la comunicación TCP. Sus características son las siguientes:

- **Data Type (Tipo de Dato).** Es el tipo de dato que soporta. En este caso, un dato Booleano (BOOL), es decir, 0 ó 1.
    - El 0 indica Falso (False). La conexión no se ha realizado.
    - El 1 indica Verdadero (True). La conexión se ha realizado.
- **Default value (Valor por Defecto).** El valor por defecto que trae este parámetro es 0, Falso (False).
- **Access Rights (Modos de Accesos).** Indica en qué modos podemos acceder al parámetro Enabled. Los modos de este parámetros son lectura y escritura (Read and write).
- **Standard Path (Ruta Estándar).** Es la ruta del Item del parámetro Enabled, para poderla emplear, por ejemplo, con los comandos de la programación LUA. La ruta es NETx.XCON.<TCP>.Enabled.

3.4. Connected

Este parámetro es el encargado de establecer la conexión TCP, es decir, cuando este parámetro vale 1 indica que el BMS Server está conctado. Al igual que el parámetro Enabled, es el responsable de la creación de una conexión o desconexión con la interfaz TCP. Si uno de los dos parámetros se encontrara a 0 (False), la conexión TCP no se realizaría. Las características de este parámetro son las siguientes:

- **Data Type (Tipo de Dato).** Es el tipo de dato que soporta, en este caso, Booleano (BOOL), es decir, 0 ó 1.
    - El 0 indica Falso (False). La conexión no se ha realizado.
    - El 1 indica Verdadero (True). La conexión se ha realizado.
- **Default value (Valor por Defecto).** El valor por defecto que trae este parámetro es 0, Falso (False).
- **Access Rights (Modos de Accesos).** Indica en qué modos podemos acceder al parámetro Connected. El modo de acceso a este parámetro es solo lectura (Read).
- **Standard Path (Ruta Estándar).** Es la ruta del Ítem del parámetro Connected, para poderla emplear, por ejemplo, con los comandos de la programación LUA. La ruta es NETx.XCON.<TCP>.Connected.

3.5. LastError

Este parámetro es el encargado de indicar si se ha establecido algún error al establecer la conexión TCP. Las características de este parámetro son:

- **Data Type (Tipo de Dato).** Es el tipo de dato que soporta, en este caso, una cadena de caracteres.
- **Default value (Valor por Defecto).** El valor por defecto que trae este parámetro es una cadena vacía (Empty string).
- **Access Rights (Modos de Accesos).** Indica en qué modos podemos acceder al parámetro LastError. Los modos de acceso a este parámetro son lectura y escritura (Read and Write).
- **Standard Path (Ruta Estándar).** Es la ruta del Ítem del parámetro LastError, para poderla emplear, por ejemplo, con los comandos de la programación LUA. La ruta es NETx.XCON.<TCP>.LastError.

3.6. LocalHost

Este parámetro es el que contiene la dirección IP Local, en este caso sería la dirección IP del BMS Server. Las características de este parámetro son:

- **Data Type (Tipo de Dato).** Es el tipo de dato que soporta, en este caso, una cadena de caracteres.
- **Default value (Valor por Defecto).** No tiene valor por defecto.
- **Access Rights (Modos de Accesos).** Indica en qué modos podemos acceder al parámetro LocalHost. Los modos de acceso a este parámetro son lectura y escritura (Read and Write).
- **Standard Path (Ruta Estándar).** Es la ruta del Ítem del parámetro LocalHost, para poderla emplear, por ejemplo, con los comandos de la programación LUA. La ruta es NETx.XCON.<TCP>.Config.LocalHost.

3.7. LocalPort

Este parámetro es el que contiene el Puerto Local, en este caso sería el Puerto del BMS Server. Las características de este parámetro son:

- **Data Type (Tipo de Dato).** Es el tipo de dato que soporta. Es un entero de 4 bytes (-128 a +127 ó 0 a 255).
- **Default value (Valor por Defecto).** El valor por defecto es 0.
- **Access Rights (Modos de Accesos).** Indica en qué modos podemos acceder al parámetro LocalPort. Los modos de este parámetros son lectura y escritura (Read and Write).
- **Standard Path (Ruta Estándar).** Es la ruta del Ítem del parámetro LocalPort, para poderla emplear, por ejemplo, con los comandos de la programación LUA. La ruta es NETx.XCON.<TCP>.Config.LocalPort.

3.8. RemoteHost

Este parámetro es el que contiene la dirección IP del dispositivo o simulador remoto para poder establecer la comunicación TCP, en nuestro caso sería la dirección IP del Simulador TCP, SocketTest. Las características de este parámetro son:

- **Data Type (Tipo de Dato).** Es el tipo de dato que soporta, en este caso, una cadena de caracteres.
- **Default value (Valor por Defecto).** No tiene valor por defecto.
- **Access Rights (Modos de Accesos).** Indica en qué modos podemos acceder al parámetro RemoteHost. Los modos de este parámetros son lectura y escritura (Read and Write).
- **Standard Path (Ruta Estándar).** Es la ruta del Ítem del parámetro RemoteHost, para poderla emplear, por ejemplo, con los comandos de la programación LUA. La ruta es NETx.XCON.<TCP>.Config.RemoteHost.

3.9. RemotePort

Este parámetro es el que contiene el Puerto del dispositivo o simulador remoto para poder establecer la comunicación TCP, en este caso sería el Puerto del Simulador TCP, SocketTest. Las características de este parámetro son:

- **Data Type (Tipo de Dato).** Es el tipo de dato que soporta, en este caso, un entero de 4 bytes (-128 a +127 ó 0 a 255).
- **Default value (Valor por Defecto).** El valor por defecto es 0.
- **Access Rights (Modos de Accesos).** Indica en qué modos podemos acceder al parámetro RemotePort. Los modos de este parámetros son lectura y escritura (Read and Write).
- **Standard Path (Ruta Estándar).** Es la ruta del Ítem del parámetro RemoteHost, para poderla emplear, por ejemplo, con los comandos de la programación LUA. La ruta es NETx.XCON.<TCP>.Config.RemotePort.

4. Parámetros necesarios para la ejecución de los Scripts LUA

La función principal que utiliza el BMS Server para la ejecución de los Scripts LUA es la función xcon.CreateTCP.

Esta función viene por defecto en el BMS Server, en el archivo nxaSystemXCON.lua, es un Script que inicializa los parámetros antes mencionados y permite utilizarlos.

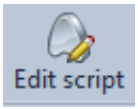
Los parámetros a tener en cuenta son:

- El identificador de la conexión dentro de la secuencia de comandos LUA.
- La dirección IP de la Interfaz Local, en este caso la dirección IP del BMS Server. Si el campo está vacío la detecta el sistema.
- El Puerto que se utiliza para la conexión, si es 0 el sistema selecciona uno.
- La dirección IP o el nombre Host del servidor remoto, en este caso el del Simulador TCP SocketTest.
- El Puerto del Servidor Remoto que se utiliza para la conexión.

5. Comunicación TCP empleando Scripts Lua

5.1. Configuración de la conexión a partir del Script por defecto

Desde el BMS Server podemos acceder a varios Scripts proporcionados por defecto simplemente clicando en “Edit script” (botón que se aprecia en la imagen adjunta). Dentro del Script nxaSystemXCON.lua, se pueden modificar los parámetros OUT, IN, Enabled, Connected, LastError, LocalHost, LocalPort, RemoteHost y RemotePort, para establecer la configuración por defecto de la comunicación TCP y así cuando se inicie el programa BMS Server aparecerá dicha configuración.



La parte que nos interesa del archivo es la comunicación TCP y es la siguiente:

```
--============================================================================

-- TCP Subsystem
```

```
--============================================================================

function Create__SysTCP()

 local CfgCngFunc = "__SysTCPConfigChanged()"

 local SndFunc    = "__SysTCPSend()"

 local EnbFunc    = "__SysTCPEnabled()"

 sysTCP = {}

 sysTCP[IsCON] = false

 sysTCP[NM]    = "__SysTCP"

 sysTCP[DP]    = "TCP"


 sysTCP[OUT] = nxa.AddSysCustomItem(OUT, EptDESC, nxa.access.All,      nxa.type.String,  "", 0, 0, IdSep, bPath, sysTCP[DP])

 sysTCP[IN]  = nxa.AddSysCustomItem(IN,  EptDESC, nxa.access.Readable, nxa.type.String,  "", 0, 0, IdSep, bPath, sysTCP[DP])

 sysTCP[ENB] = nxa.AddSysCustomItem(ENB, EptDESC, nxa.access.All,      nxa.type.Boolean, "", 0, 0, IdSep, bPath, sysTCP[DP])

 sysTCP[CON] = nxa.AddSysCustomItem(CON, EptDESC, nxa.access.Readable, nxa.type.Boolean, "", 0, 0, IdSep, bPath, sysTCP[DP])

 sysTCP[LE]  = nxa.AddSysCustomItem(LE,  EptDESC, nxa.access.All,      nxa.type.String,  "", 0, 0, IdSep, bPath, sysTCP[DP])


 sysTCP[LH]  = nxa.AddExtSysCustomItem(LH, EptDESC, nxa.access.All, nxa.type.String,  true, false, false, "", 0, 0, IdSep, bPath, sysTCP[DP],
cfgPath)

 sysTCP[LP]  = nxa.AddExtSysCustomItem(LP, EptDESC, nxa.access.All, nxa.type.Integer, true, false, false, "", 0, 0, IdSep, bPath, sysTCP[DP],
cfgPath)

 sysTCP[RH]  = nxa.AddExtSysCustomItem(RH, EptDESC, nxa.access.All, nxa.type.String,  true, false, false, "", 0, 0, IdSep, bPath, sysTCP[DP],
cfgPath)

 sysTCP[RP]  = nxa.AddExtSysCustomItem(RP, EptDESC, nxa.access.All, nxa.type.Integer, true, false, false, "", 0, 0, IdSep, bPath, sysTCP[DP],
cfgPath)


 nxa.SetValue(sysTCP[ENB], false)

 nxa.SetValue(sysTCP[CON], false)

 nxa.SetValue(sysTCP[LE], "")

 nxa.SetValue(sysTCP[LH], "")

 nxa.SetValue(sysTCP[LP], 0)

 nxa.SetValue(sysTCP[RH], "")

 nxa.SetValue(sysTCP[RP], 0)


 nxa.AddScriptTask(sysTCP[LH],  "", true, true, true, 100, CfgCngFunc)

 nxa.AddScriptTask(sysTCP[LP],  "", true, true, true, 100, CfgCngFunc)

 nxa.AddScriptTask(sysTCP[RH],  "", true, true, true, 100, CfgCngFunc)

 nxa.AddScriptTask(sysTCP[RP],  "", true, true, true, 100, CfgCngFunc)

 nxa.AddScriptTask(sysTCP[OUT], "", true, true, true, 100, SndFunc)

 nxa.AddScriptTask(sysTCP[ENB], "", true, true, true,   0, EnbFunc)

end
```

```
function __SysTCPEnabled()
 xcon.Close(sysTCP[NM])
 if (nxa.InputValue() == true) then
   -- nxa.LogInfo("Sys TCP: Enabled")
   xcon.CreateTCP(
     sysTCP[NM],
     nxa.GetValue(sysTCP[LH], ""),
     nxa.GetValue(sysTCP[LP], 0),
     nxa.GetValue(sysTCP[RH], ""),
     nxa.GetValue(sysTCP[RP], 0))
 end
end

function __SysTCPConfigChanged()
 -- nxa.LogInfo("Sys UDP: configuration changed")
 nxa.SetValue(sysTCP[ENB], false)
end

function __SysTCPSend()
 if (sysTCP[IsCON] == false) then
   nxa.SetValue(sysTCP[LE], "Not connected")
 else
   xcon.SendText(sysTCP[NM], nxa.InputValue())
 end
end

function __SysTCP_OnErrorEvent(vData)
 -- nxa.LogWarning("Sys TCP: " .. vData)
 nxa.SetValue(sysTCP[LE], vData)
end

function __SysTCP_OnConnectEvent()
 -- nxa.LogInfo("Sys TCP: connected")
 sysTCP[IsCON] = true
 nxa.SetValue(sysTCP[CON], true)
end

function __SysTCP_OnDisconnectEvent()
 -- nxa.LogInfo("Sys TCP: disconnected")
```

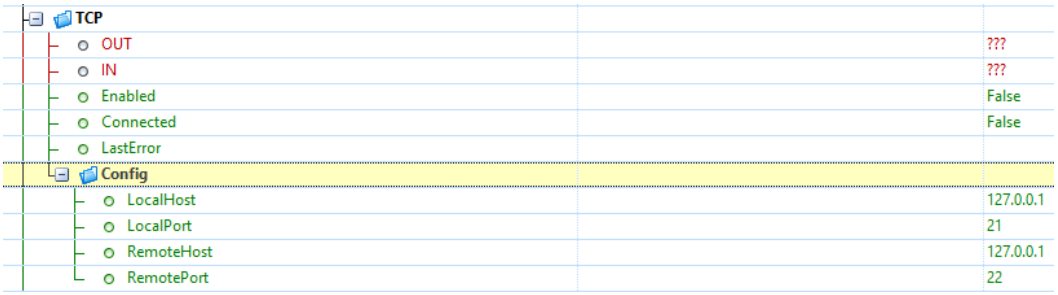
```
 sysTCP[IsCON] = false
 nxa.SetValue(sysTCP[CON], false)
end

function __SysTCP_OnReceiveEvent(vData, vSize, vIP, vPort)
 -- nxa.LogInfo("Sys TCP: received " .. vSize .. " Bytes from " .. vIP .. ":" .. vPort .. " = " .. vData)
 nxa.SetValue(sysTCP[IN], vData)
end
```

Dentro de esta parte nos centraremos en las líneas donde modificar la configuración por defecto, estas líneas son las que se muestran a continuación:

```
nxa.SetValue(sysTCP[ENB], false)
nxa.SetValue(sysTCP[CON], false)
nxa.SetValue(sysTCP[LE], "")
nxa.SetValue(sysTCP[LH], "127.0.0.1")
nxa.SetValue(sysTCP[LP], 21)
nxa.SetValue(sysTCP[RH], "127.0.0.1")
nxa.SetValue(sysTCP[RP], 22)
```

Se ha puesto por defecto la configuración que hemos utilizado para los ejemplos anteriores.

Al iniciar el BMS Server obtenemos:



Como podemos apreciar, la configuración se ha realizado de forma correcta e idéntica a la configuración establecida manualmente. De esta forma, incorporando los parámetros de configuración en el script de lua nos evitamos realizar la configuración de forma manual cada vez que arrancamos la simulación del BMS Server.

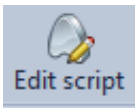
5.2. Configuración a través de un Scripts propio

Para crear un Script nos dirigimos a la ruta:

C:\Program Files\NETxAutomation\NETx.BMS.Server.2.0\Workspaces\MyFirstWorkspace\ScriptFiles

Creamos un documento de texto con extensión ".lua" donde escribiremos nuestro código.

Una vez creado el archivo que aloja el código lua, podremos modificarlo desde el propio BMS Server o desde el block de notas o programas similiares. En el caso de que deseemos modificar el script desde el propio programa, clicaremos en el botón Edit Script que se aprecia en la siguiente imagen:



El código empleado en nuestro Script es el siguiente:

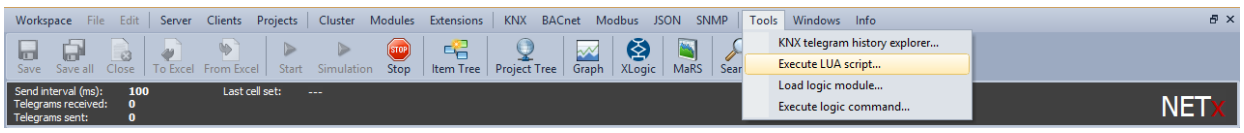
```
function TCP()

-- Configuración de la comunicación TCP entre NETx BMS Studio y un simulado TCP, en este caso se

-- ha utulizado SocketTest v 3.0.0


        nxa.SetValue("NETx.XCON.TCP.Config.LocalHost","127.0.0.1")

        nxa.SetValue("NETx.XCON.TCP.Config.LocalPort","21")

        nxa.SetValue("NETx.XCON.TCP.Config.RemoteHost","127.0.0.1")

        nxa.SetValue("NETx.XCON.TCP.Config.RemotePort","22")


-- Mensaje informativo


        nxa.LogInfo("Configuración establecida")


-- Ejecución de la Tarea. Escribe en el parámetro OUT del BMS Server el valor del Holding Registers 1.

-- Escribe en el Holding Register 0 el valor del parámetro IN.

        nxa.SetValue("NETx.XCON.TCP.OUT",nxa.GetValue("NETx\\XIO\\Modbus\\MODBUS1\\Holding Registers\\1"))

        nxa.WriteValue("NETx\\XIO\\Modbus\\MODBUS1\\Holding Registers\\0",nxa.GetValue("NETx.XCON.TCP.IN"))

end
```

Una vez ya creado el script con nuestro código, será necesario dar un paso más para poder ejecutarlo. Al igual que hemos hecho con nuestro script, abriremos el script “nxaDefinitions.lua” alojado en la ruta especificada anteriormente e incluiremos la siguiente línea:

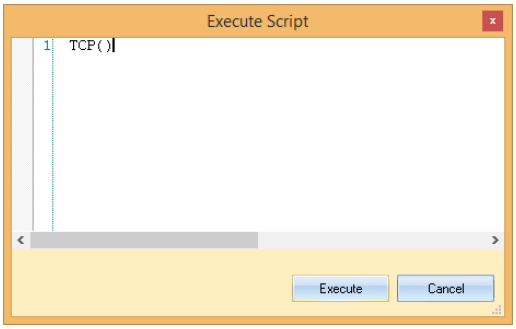
```
require "Nombredelarchivo"
```

Esta línea se incluirá a continuación de las líneas ya presentes en el archivo que presentan un “require” y nos permitirá el empleo de nuestro propio Script.

Una vez creado y modificado el Script hasta tener el código deseado, desde el BMS Server se ejecuta el Script desde la pestaña Execute LUA script, que se encuentra en Tools.



Tras realizar este paso nos aparecerá la siguiente ventana:



Para ejecutar el código deseado escribiremos el nombre de la función que deseamos ejecutar tal y como puede apreciarse en la imagen superior, y presionamos execute. En el caso de que todo se desarrolle de forma correcta se ejecutará nuestro código, en el caso de que se produzca un error en la ejecución o un mensaje, éste aparecerá en la ventana System messages del programa.

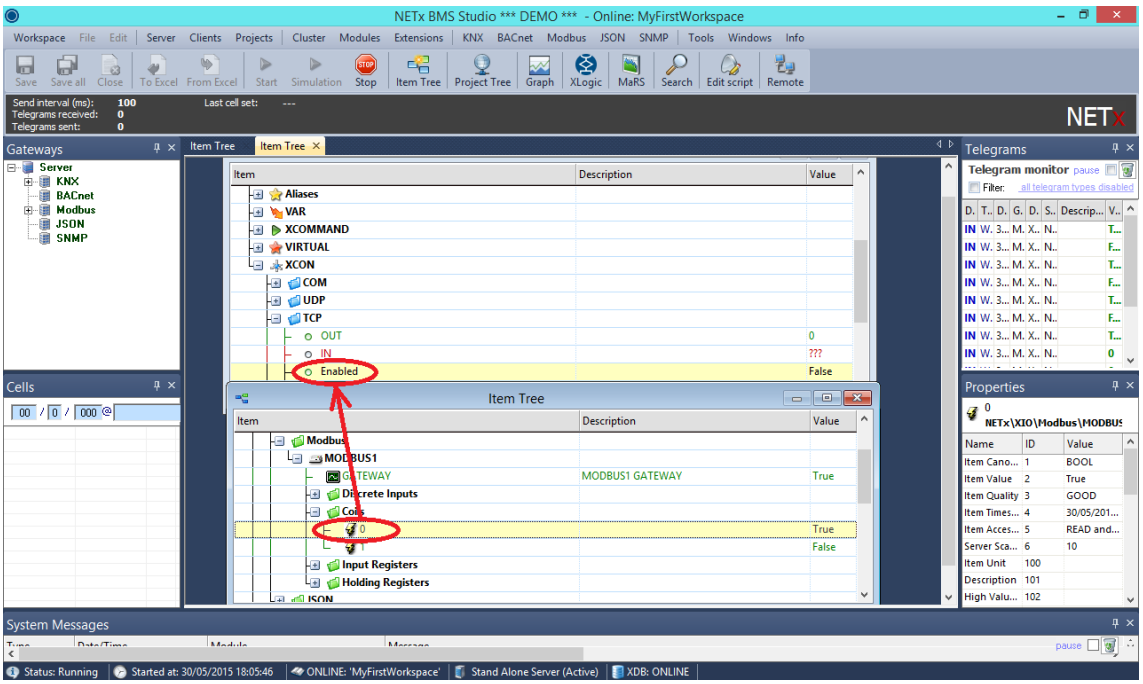
Al ejecutar este script se modifican los parámetros ya mencionados en la configuración manual:

- LocalHost: 127.0.0.1
- LocalPort: 21
- RemoteHost: 127.0.0.1
- RemotePort: 22

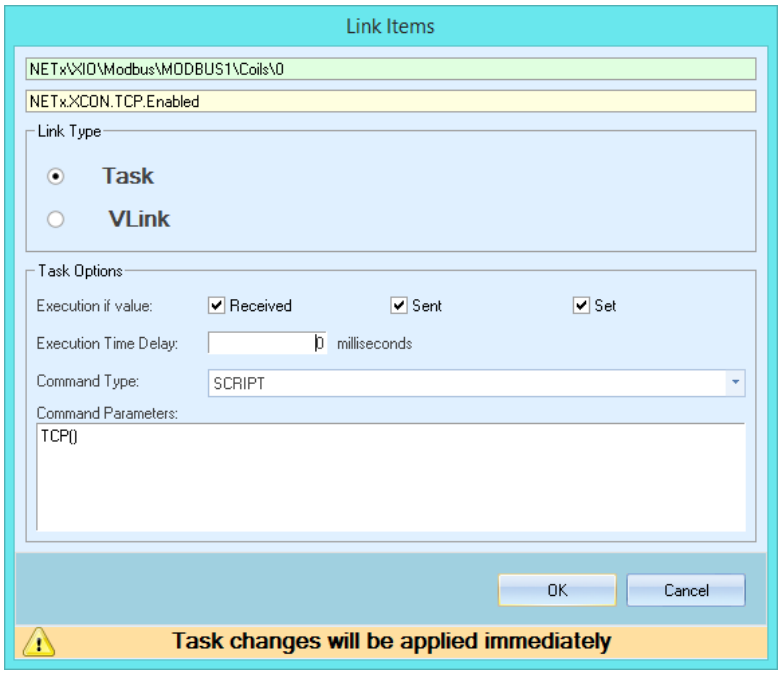
Como podemos apreciar el parámetro Enable no se modifica con nuestro script, esto es debido a que dicho parámetro debe ser modificado de forma manual, de manera que será posible realizar y ejecutar una función dentro de un script que establezca la configuración pero para establecer la conexión será preciso que se modifique el parámetro Enable manualmente. Tras esto, ya queda configurada la conexión y será posible emplear la conexión TCP a nuestro antojo.

5.3. *Ejecución de un script asociado a un parámetro o punto virtual*

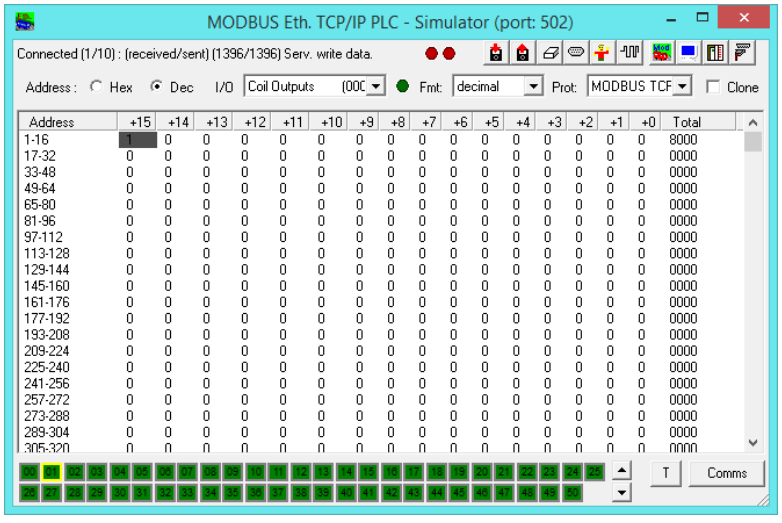
Es posible configurar el BMS Server vinculando la ejecución de un script de lua a un parámetro o punto virtual, de manera que modificando éste se ejecute el parámetro.



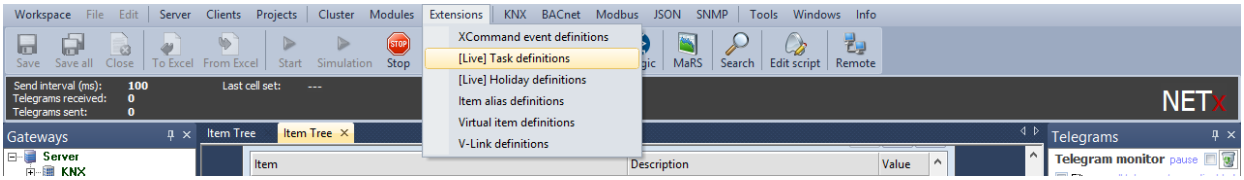
Para configurar la ejecución de ésta forma el procedimiento es muy sencillo, abriremos dos pestañas Item Tree, de forma que podamos visualizar el parámetro al que deseamos asignar la tarea que ejecutará el script y el punto con el que deseamos vincularlo. A continuación procedemos arrastrando uno sobre otro como se aprecia en la imagen superior, de forma que se nos mostrará la siguiente ventana:



Como podemos apreciar en la parte superior de la ventana se muestran los dos puntos vinculados, en nuestro caso se ha vinculado el "Coil 0" con el parámetro "Enabled" dentro de "XCON/TCP". Para asignar el script a ejecutar seleccionaremos la opción "Task" dentro de "Link Type" y dentro de "Task Options" seleccionaremos "SCRIPT" como "Comand Type" y escribiremos el nombre de la función a ejecutar dentro del cuadro de diálogo "Command Parameters". Finalmente cliclaremos en "Ok" y ambos puntos quedarán vinculados entre sí y a la tarea seleccionada. Una vez realizado todos los pasos descritos, en nuestro ejemplo, cuando se modifique el Coil asignado desde el terminal (tal como se aprecia en la imagen inferior), se ejecutará la tarea,



También se pueden crear tareas desde la pestaña "Extensions/Task definitions". Este procedimiento ya se encuentra explicado en la documentación proporcionada con el programa, por lo que para evitar la duplicidad de información, consultaremos dicha documentación en el caso de que sea necesario.



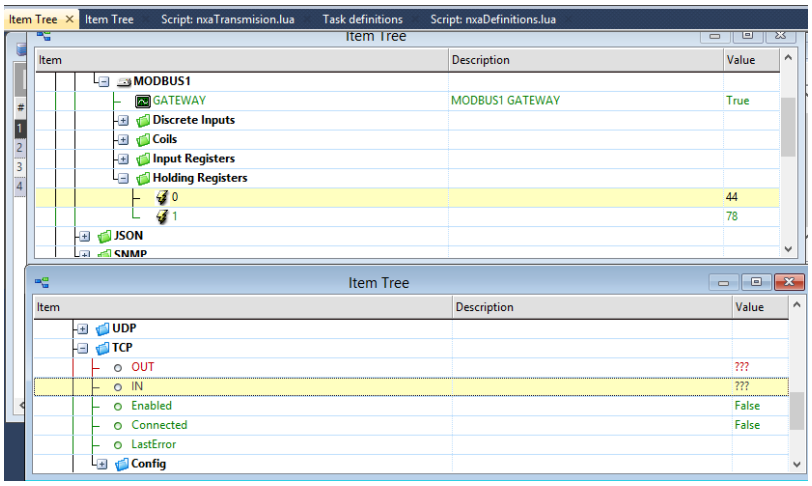
5.4. Transmisión de un mensaje mediante TCP

Para transmitir un mensaje empleando la comunicación TCP se ha generado un Script llamado "nxaTransmision.lua". El contenido de este Script puede apreciarse a continuación:

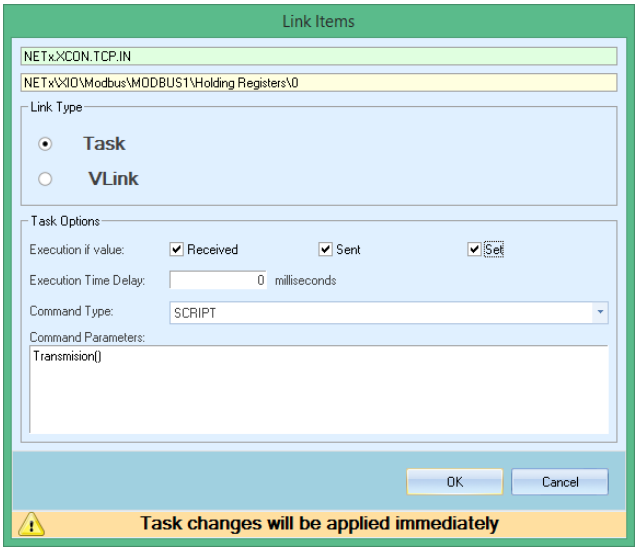
```
function Transmision()

-- Comunicación TCP entre NETx BMS Studio y un simulador TCP, en este caso se

-- ha utulizado SocketTest v 3.0.0

nxa.SetValue("NETx.XCON.TCP.OUT",nxa.GetValue("NETx\\XIO\\Modbus\\MODBUS1\\Holding Registers\\1"))

-- Mensaje informativo

        nxa.LogInfo("Comunicacion establecida")

end
```

Este Script transmitirá mediante TCP el valor del "Holding Register 1", y dicho valor aparecerá en el terminal, en nuestro caso, SocketTest. Para ello emplearemos la función "nxa.SetValue()", esta función se encargará de tomar el valor del Holding Register 1 y enviarlo mediante el parámetro "OUT" de TCP, tal y como puede apreciarse en el código. El procedimiento es el siguiente:

1. Se vincula la entrada de datos con el Holding Register 0



2. Se asigna el script de lua a la tarea en concreto y se configura para que al modificar el Holding Register se realice un envío. De esta forma realizamos un envío al BMS y programamos una recepción de datos.



3. Modificamos el Holding Register introduciendo un valor y éste aparecerá en el terminal tal y como se aprecia en la captura de pantalla: