TUTORIAL XCON INTERFACES HTTP Y RSS

TABLA DE CONTENIDO

Contenido

HTTP Y RSS

Definiciones según el manual

INTERFAZ HTTP:

Esta interfaz se encuentra incluida dentro de la interfaz XCON que a su vez se encuentra dentro de NETx.

Según el manual oficial de NETx BMS Studio:

La interfaz HTTP puede ser usada para obtener información de un host remoto desde el servidor de NETx BMS. Una simple petición HTTP es enviada al host remoto y su resultado es almacenado en otro ítem. Éste puede ser leído, analizado, y más información puede ser sacada de él. Necesita ser habilitada antes de usarla.

Parámetros:

- GET:
    - El tipo de datos es String, su valor por defecto es None.
    - Las reglas de acceso establecidas permiten la Lectura y la Escritura.
    - Su ruta estándar es NETx.XCON.<HTTP>.GET.
    - Es necesario que el parámetro Enabled esté activado para que al modificar el parámetro se envié la petición HTTP get de forma inmediata.
    - Nota: No es necesario incluir http:// ya que es incluido al llamar al comando.
    - Ejemplo: www.dte.us.es

- PUT:
    - El tipo de datos es String, su valor por defecto es None.
    - Las reglas de acceso establecidas permiten la Lectura y la Escritura.
    - Su ruta estándar es NETx.XCON.<HTTP>.PUT
    - Es necesario que el parámetro Enabled esté activado para que al modificar el parámetro se envié la petición HTTP put de forma inmediata.
    - Nota: No es necesario incluir http:// ya que es incluido al llamar al comando.

- RESULT:

  - El tipo de datos es String, su valor por defecto es None.
  - Las reglas de acceso establecidas permiten la Lectura.
  - Su ruta estándar es NETx.XCON.<HTTP>.RESULT.
  - El resultado de la última petición HTTP enviada será almacenada en este ítem.
  - Ejemplo: <HTML><HEAD>My Page</HEAD><BODY>Hello World!</BODY ></HTML>

- Enabled:

  - El tipo de datos es Boolean y su valor por defecto es 0 (False).
  - Las reglas de acceso establecidas permiten la Lectura y la Escritura.
  - Su ruta estándar es NETx.XCON.<HTTP>.Enabled.
  - Según el manual, si este ítem del servidor es puesto a "True" cualquier comando Get Put será inmediatamente enviado fuera del servidor a su destino determinado, pero al menos en esta versión de NETx BMS Studio (2.0.7210) y en la anterior (2.0.7000), es necesario ponerlo éste parámetro a True antes de hacer el comando Get o Put, ya que si se hace como dice el manual tendremos que volver a escribir el comando Get/Put de nuevo una vez establecido Enabled como True.

- LastError:

  - El tipo de datos es String y su valor por defecto es empty string.
  - Las reglas de acceso establecidas permiten la Lectura y la Escritura.
  - Su ruta estándar es NETx.XCON.<HTTP>.LastError
  - Si un error ocurre durante la transmisión, éste será almacenado aquí. El ítem del servidor mantendrá este valor. Este cambiará en el siguiente error, también puede ser sobrescrito mediante un script y de otras maneras.

HTTP Y RSS

INTERFAZ RSS:

Esta interfaz se encuentra incluida dentro de la interfaz XCON que a su vez se encuentra dentro de NETx.

Según el manual oficial de NETx BMS Studio:

La interfaz RSS permite al usuario obtener información desde un RSS provider. Ésta necesita ser habilitada antes para poder usarse. Escribiendo una RSS URL inicializaremos la comunicación.

El RSS facilita la gestión y publicación de información y noticia webs. RSS es una forma estandarizada de distribución de la información de las páginas web a los lectores de las páginas. Esta información se distribuye a través de las fuentes RSS oCanales RSS. Gracias al RSS, los lectores pasan a tener una herramienta útil para mantenerse informado sobre las noticias y webs que le resultan de interés, conservando y almacenando toda la información en un solo lugar que se actualiza de manera automática.

En el archivo RSS simplemente están los datos de las novedades del sitio, como el título, fecha de publicación o la descripción. El programa que lea el RSS será encargado de darle estilo o apariencia a los datos que se incluyan en el archivo y presentarlos de una manera atractiva al usuario y de fácil lectura.

Que RSS sea un formato basado en XML significa que el archivo RSS se compone por una serie de etiquetas definidas que tendrán un formato dado, que respetará las reglas generales de XML.

Parámetros:

- GET:
    - El tipo de datos es String, su valor por defecto es None.
    - Las reglas de acceso establecidas permiten la Lectura y la Escritura.
    - Su ruta estándar es NETx.XCON.<RSS>.GET.
    - Es necesario que el parámetro Enabled esté activado para que al modificar el parámetro se envié la petición RSS get de forma inmediata.
    - La URL de la fuente RSS se colocará aquí.
    - Ejemplo: http://www.dte.us.es (dado que funciona igual que http para url normales)
    - En el caso de usar una URL RSS sería necesario poner “http:// “ dirección “.xml” Ejemplo: http://estaticos.elmundo.es/elmundo/rss/portada.xml

- CHANNEL:

    - El tipo de datos es INT4, su valor por defecto es -1.
    - Las reglas de acceso establecidas permiten la Lectura y la Escritura.
    - Su ruta estándar es NETx.XCON.<RSS>.CHANNEL
    - El canal es índice basado en cero del canal que se quiera obtener del documento RSS. Con "-1" se devuelven todos los canales.

- RESULT:

    - El tipo de datos es String, su valor por defecto es 0 (False).
    - Las reglas de acceso establecidas permiten la Lectura.
    - Su ruta estándar es NETx.XCON.<RSS>.RESULT.
    - El resultado de la última petición RSS enviada será almacenada en este ítem.
    - Ejemplo: Código del .xml

- Enabled:

    - El tipo de datos es Boolean y su valor por defecto es 0 (False).
    - Las reglas de acceso establecidas permiten la Lectura y la Escritura.
    - Su ruta estándar es NETx.XCON.<RSS>.Enabled.
    - Según el manual, si este ítem del servidor es puesto a "True" cualquier comando Get será inmediatamente enviado fuera del servidor a su destino determinado, pero al menos en esta versión de NETx BMS Studio (2.0.7210) y en la anterior (2.0.7000), es necesario ponerlo éste parámetro a True antes de hacer el comando Get, ya que si se hace como dice el manual tendremos que volver a escribir el comando Get de nuevo una vez establecido Enabled como True.

- LastError:

    - El tipo de datos es String y su valor por defecto es empty string.
    - Las reglas de acceso establecidas permiten la Lectura y la Escritura.
    - Su ruta estándar es NETx.XCON.<RSS>.LastError
    - Si un error ocurre durante la transmisión, éste será almacenado aquí.

HTTP Y RSS

NOTA: Es necesario también comentar las dos funciones NXA para enviar emails existentes para HTTP y RSS:

- **xcon.CreateHTTP:**
  Esta función puede ser usada para crear una conexión HTTP en LUA.
  Parámetros:
  - string – Handle que identifica la conexión dentro del script LUA.

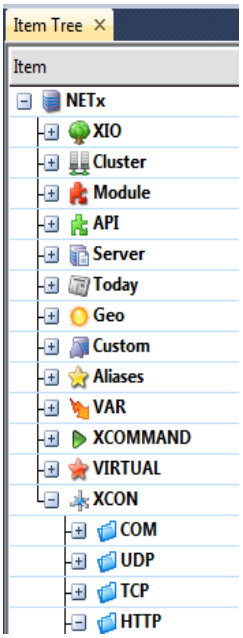
- **xcon.CreateRSS:**
  Esta función puede ser usada para crear una conexión RSS en LUA.
  Parámetros:
  - string – Handle que identifica la conexión dentro del script LUA.

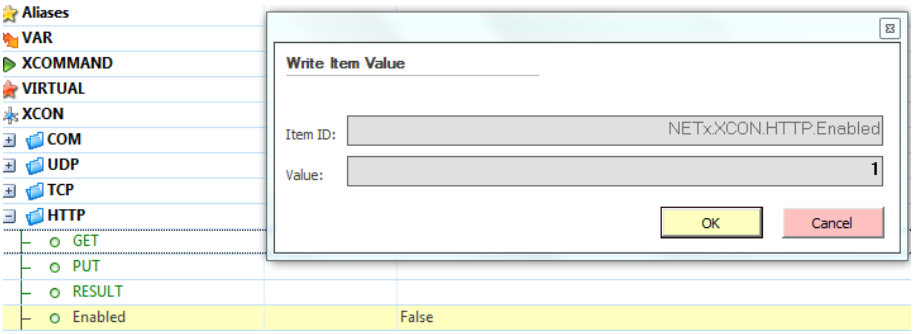
Estas serán utilizadas en los ejemplos.

Ejecución y ejemplos de la interfaz HTTP

Primero vamos a ver cómo funciona el comando Get. Podemos hacerlo de dos formas:

- DE FORMA MANUAL USANDO LA INTERFAZ GRÁFICA:
  De esta manera es muy sencillo obtener el resultado y visualizar el resultado. Primero, debemos desglosar la subpestaña NETx del Item tree, después desglosar la subpestaña XCON y finalmente desglosar también el apartado de la interfaz HTTP.
  Una vez hecho esto, tan solo tenemos que clicar con el botón derecho del ratón en el apartado Enabled -> Write ítem value y ponemos 1.

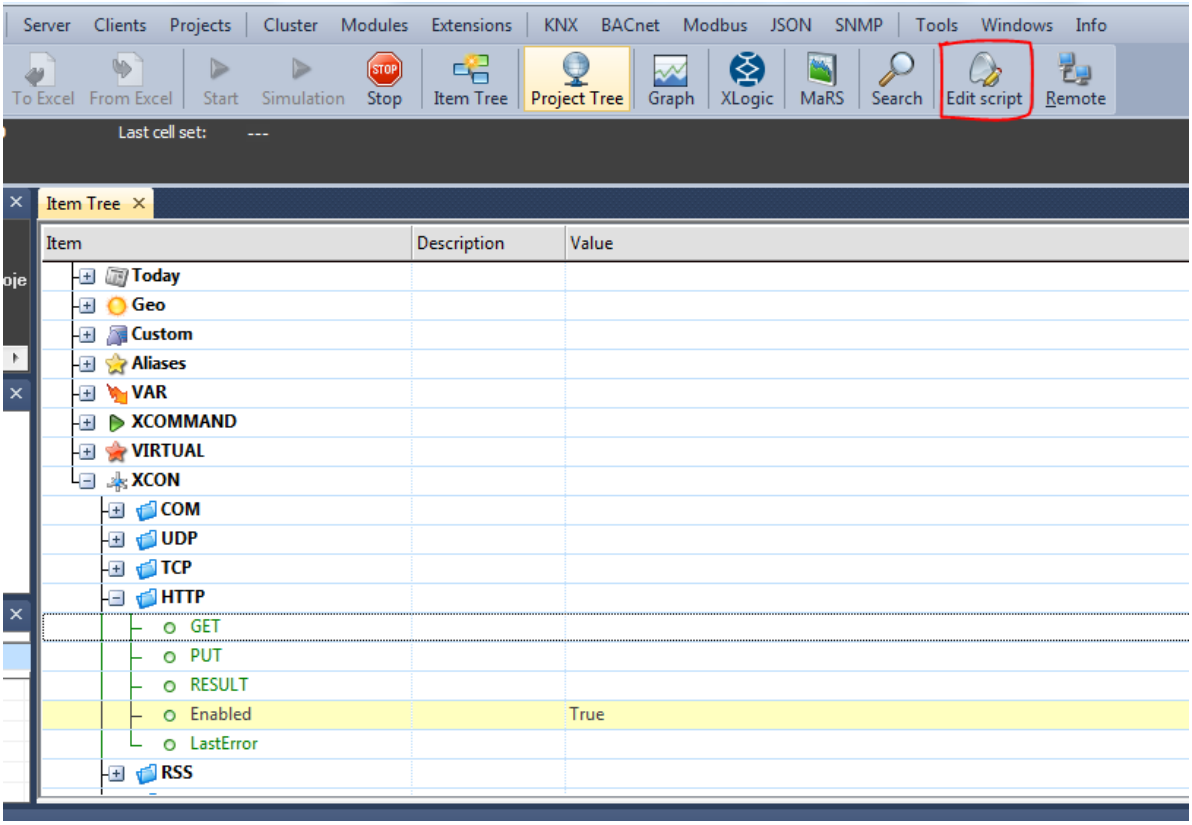




Así pondremos enabled a True y habilitaremos la conexión HTTP.
Ahora, procederemos a realizar el envío del comando Get, para ello haremos click derecho en GET -> Write ítem value y ponemos la URL deseada (recordar que no es necesario incluir http:// ya que se incluirá automáticamente en el envío).
Por ejemplo:
Si escribimos “www.google.es” obtendremos en RESULT el código HTML de la página web.

- MEDIANTE SCRIPTS: Esta forma es un poco más compleja, pero conociendo los comandos adecuados veremos que es muy útil, ya que podríamos establecer que cada vez que se iniciase el servidor NETx BMS se ejecutase dicho fragmento de código entre otras utilidades.

  Primero crearemos el script, para ello nos crearemos un archivo http.lua en la carpeta donde se almacenan los script cuya ruta es C:\Program Files (x86)\NETxAutomation\NETx.BMS.Server.2.0\Workspaces\MyFirstWorkspace\ScriptFiles. Una vez hecho esto clicaremos en la opción Edit script, y elegiremos el .lua que acabamos de crear.



  A continuación se mostrará el contenido de http.lua y se explicarán algunas de las funciones utilizadas.

HTTP Y RSS

Script: http.lua | Script: nxaDefinitions.lua | Script: nxaSystemXCON.lua | Item Tre

```
1  function getHTTP (url)
2      local reg = "a"
3      xcon.CreateHTTP(reg)
4      nxa.SetValue("NETx.XCON.HTTP.Enabled", true)
5      nxa.WriteValue("NETx.XCON.HTTP.GET", url)
6      res = nxa.ReadValue("NETx.XCON.HTTP.RESULT")
7      nxa.LogInfo(res)
8  end
9
10
```

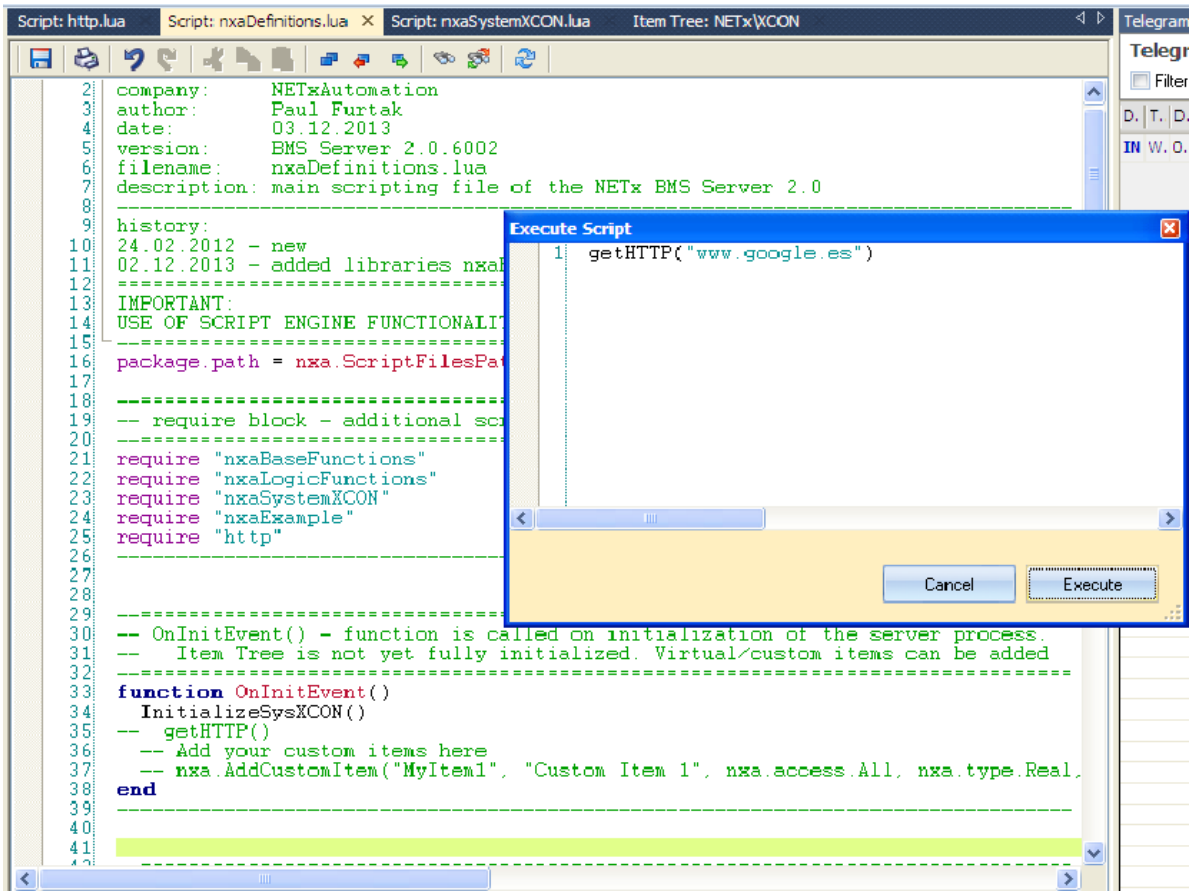
- xcon.CreateHTTP(Handle): Crea una conexión HTTP en LUA.
- nxa.SetValue ("rutaDelItem", valor): rutaDelItem obtenida haciendo click derecho en Enabled-> copy ítem path to clipboard

- nxa.WriteValue ("rutaDelItem", valor): Esta función escribe activamente el valor dado del item especificado. En comparación con SetValue, una escritura de valor también propagará el cambio de valor al dispositivo de campo. Un retraso opcional también podría especificarse.

- nxa.ReadValue ("ruta"): Lee el valor del campo RESULT

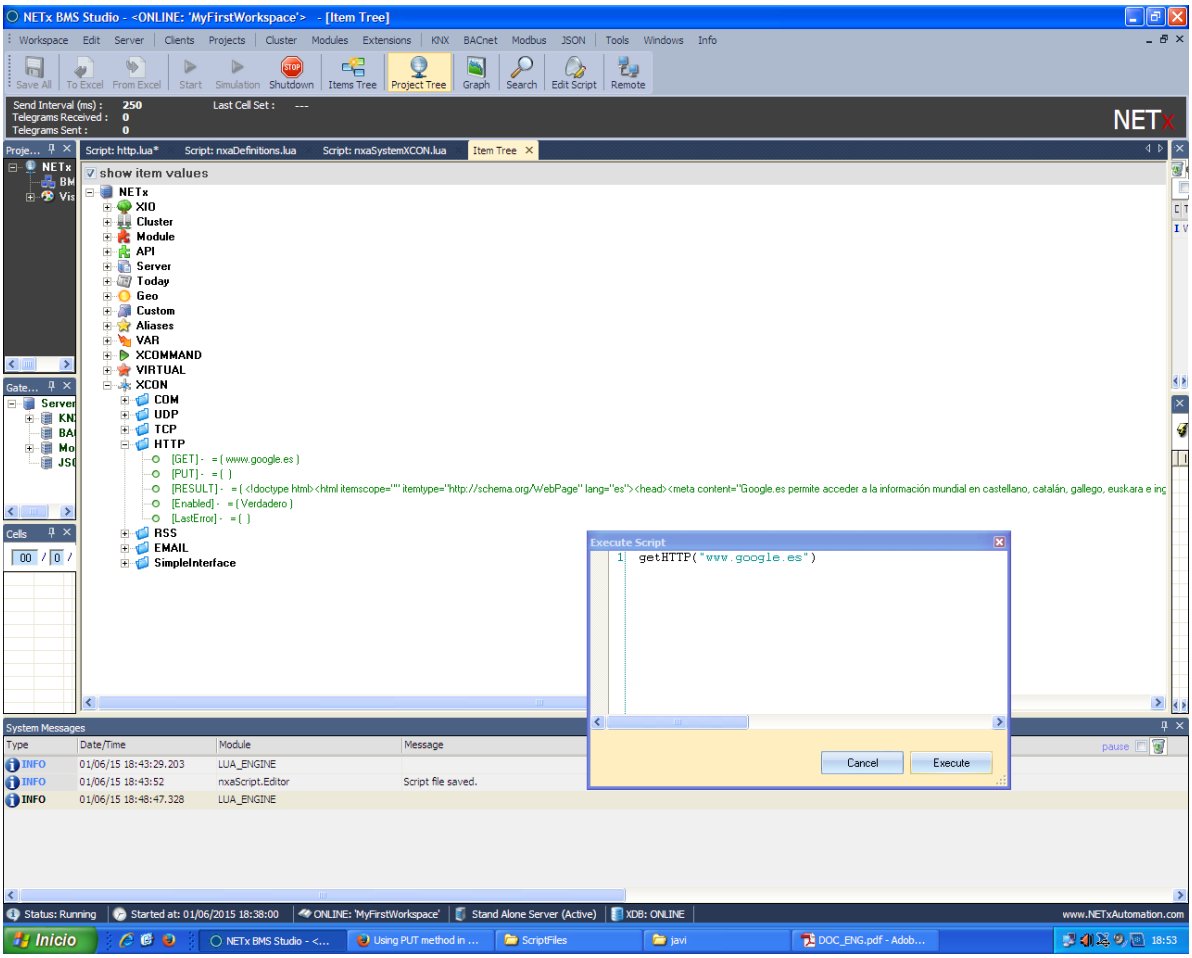
- nxa.LogInfo(lo que se desea mostrar por la consola System Messages)

También es necesario modificar el script nxaDefinitions.lua, si queremos que nuestro script pueda ser ejecutado mediante la herramienta Execute LUA script deberemos incluir el script creado en este mediante el comando require tal y como podemos observar en la siguiente captura:



Por último, en la siguiente página mostraremos el resultado de ejecutar dicho script viendo cómo se rellenan los campos de la interfaz gráfica por sí mismos.

HTTP Y RSS


NETx BMS Studio - <ONLINE: 'MyFirstWorkspace'> - [Item Tree]
Workspace Edit Server Clients Projects Cluster Modules Extensions KNX BACnet Modbus JSON Tools Windows Info
Save All To Excel From Excel Start Simulation Shutdown Items Tree Project Tree Graph Search Edit Script Remote
Send Interval (ms) : 250
Telegrams Received : 0
Telegrams Sent : 0
Last Cell Set : ---
NETx
Script: http.lua*
Script: nxaDefinitions.lua
Script: nxaSystemXCON.lua
Item Tree
show item values
NETx
XIO
Cluster
Module
API
Server
Today
Geo
Custom
Aliases
VAR
XCOMMAND
VIRTUAL
XCON
COM
UDP
TCP
HTTP
[GET] - = { www.google.es }
[PUT] - = { }
[RESULT] - = { <!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="es"><head><meta content="Google.es permite acceder a la información mundial en castellano, catalán, gallego, euskara e ing
[Enabled] - = { Verdadero }
[LastError] - = { }
RSS
EMAIL
SimpleInterface
Execute Script
1 getHTTP("www.google.es")
Cancel
Execute
System Messages
Type Date/Time Module Message
INFO 01/06/15 18:43:29.203 LUA_ENGINE
INFO 01/06/15 18:43:52 nxaScript.Editor Script file saved.
INFO 01/06/15 18:48:47.328 LUA_ENGINE
Status: Running
Started at: 01/06/2015 18:38:00
ONLINE: 'MyFirstWorkspace'
Stand Alone Server (Active)
XDB: ONLINE
www.NETxAutomation.com
Inicio
NETx BMS Studio - <...
Using PUT method in ...
ScriptFiles
javi
DOC_ENG.pdf - Adob...
18:53

A la hora de probar el comando Put el procedimiento sería el mismo, tan solo sería necesario cambiar la ruta y la url en el comando nxa.WriteValue, y poner en su lugar la sintaxsis adecuada para una HTTP Put request, sería algo similar a esto:

```
*** REQUEST
GET /users/123 HTTP/1.1
Host: www.example.org
...

*** RESPONSE
200 OK HTTP/1.1
...

<html>
...
<form action="http://www.example.org/users/123" method="put" if-match="q1w2e3r4t5">
  <input name="user-name" value="" />
  <input name="hat-size" value="" />
  <input type="submit" />
</form>
...
</html>

*** REQUEST
PUT /users/123 HTTP/1.1
Host: www.example.org
If-Match: "q1w2e3r4t5"
Content-Type: application/x-www-form-urlencoded
Content-Length: nnn

user-name=mike&hat-size=medium

*** RESPONSE
200 OK HTTP/1.1
...
<html>
...
<ul>
  <li>user-name: mike</li>
  <li>hat-size: medium</li>
</ul>
...
<html>
```

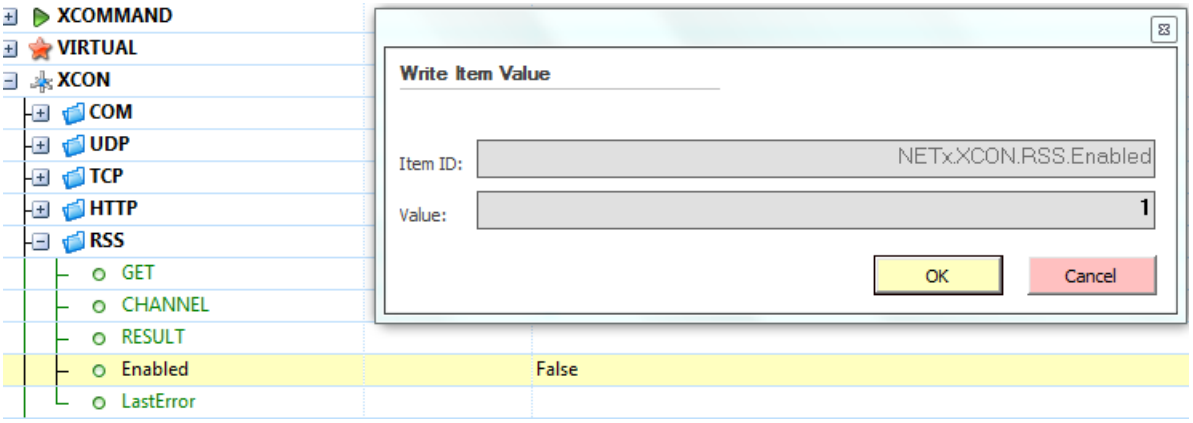
PROBLEMAS ENCONTRADOS:

En la última versión (2.0.7210) no es posible, mediante la interfaz gráfica ver los resultados en RESULT tras introducir la url en GET. Tan solo muestra a intervalos irregulares una página amarilla en resolución desmedida (lo veremos en el ejemplo de RSS), para la realización del ejemplo hecho del comando Get ha sido necesario utilizar la versión 2.0.7000 del software.

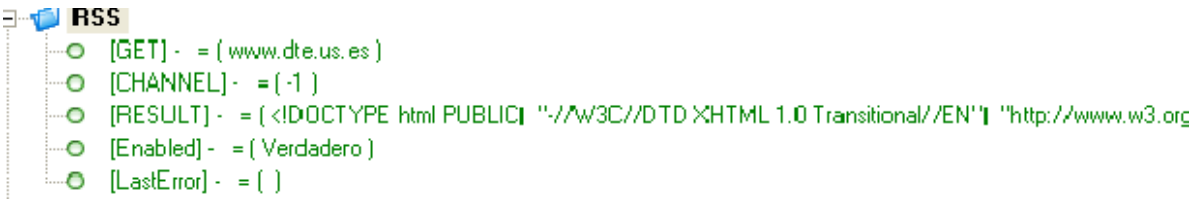
Ejecución y ejemplos de la interfaz RSS

Ahora, vamos a comprobar el funcionamiento de del comando Get de la interfaz RSS. Podemos hacerlo de dos formas:

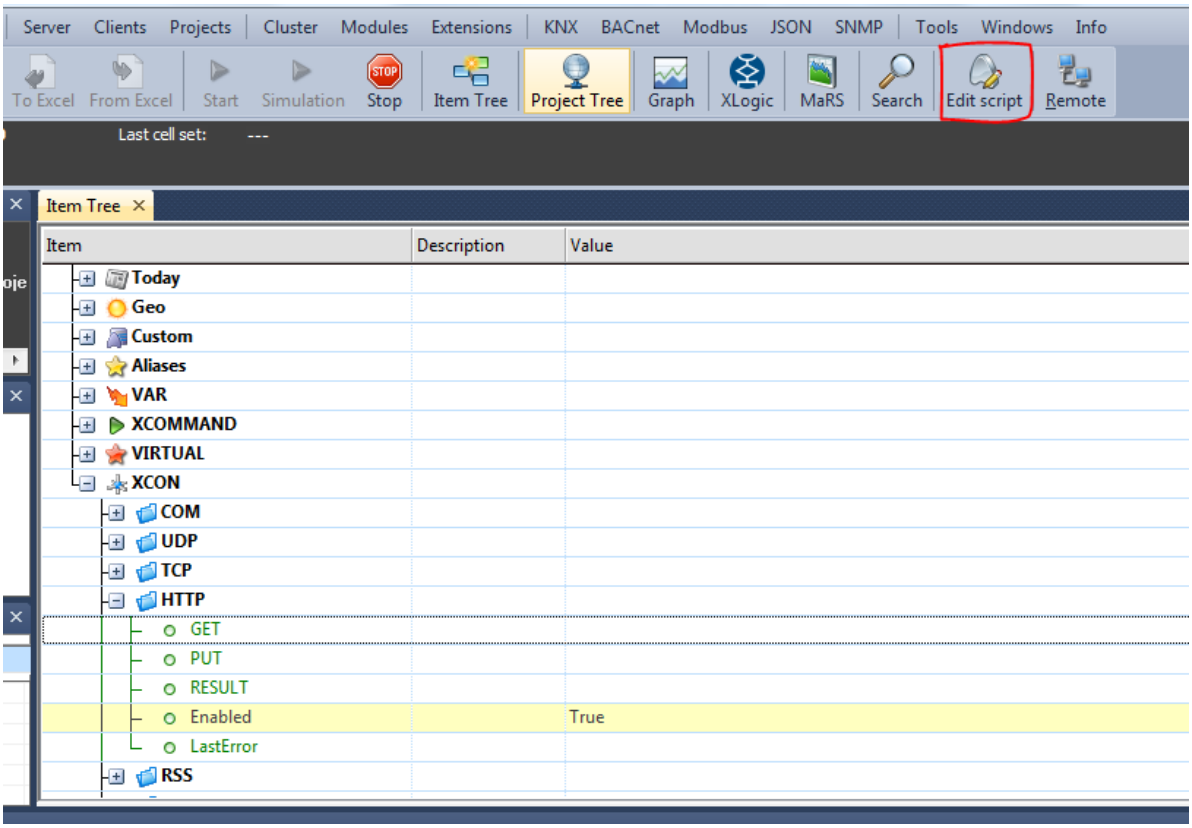
- DE FORMA MANUAL USANDO LA INTERFAZ GRÁFICA:
  De esta manera es muy sencillo obtener el resultado y visualizar el resultado. Primero, debemos desglosar la subpestaña NETx del Item tree, después desglosar la subpestaña XCON y finalmente desglosar también el apartado de la interfaz RSS.
  Una vez hecho esto, tan solo tenemos que clicar con el botón derecho del ratón en el apartado Enabled -> Write ítem value y ponemos 1.



Así pondremos enabled a True y habilitaremos la conexión RSS.
Ahora, procederemos a realizar el envío del comando Get, para ello haremos click derecho en GET -> Write ítem value y ponemos la URL deseada (recordar que no es necesario incluir http:// ya que se incluirá automáticamente en el envío en caso de ser una url normal).
Por ejemplo:
Si escribimos "www.dte.us.es" obtendremos en RESULT el código HTML de la página web.



- MEDIANTE SCRIPTS: Primero crearemos el script, para ello nos crearemos un archivo rss.lua en la carpeta donde se almacenan los script cuya ruta es C:\Program Files (x86)\NETxAutomation\NETx.BMS.Server.2.0\Workspaces\MyFirstWorkspace\ScriptFiles. Una vez hecho esto clicaremos en la opción Edit script, y elegiremos el .lua que acabamos de crear.



A continuación se mostrará el contenido de http.lua y se explicarán algunas de las funciones utilizadas.

HTTP Y RSS

Script: http.lua | Script: nxaDefinitions.lua | Script: nxaSystemXCON.lua | Item Tree | Script: rss.lua

```lua
function getRSS (url)
    local reg = "a"
    xcon.CreateRSS(reg)
    nxa.SetValue("NETx.XCON.RSS.Enabled", true)
    nxa.WriteValue("NETx.XCON.RSS.GET", url)
    res = nxa.ReadValue("NETx.XCON.RSS.RESULT")
    nxa.LogInfo(res)
    nxa.LogInfo("Terminado getRSS")
end
```

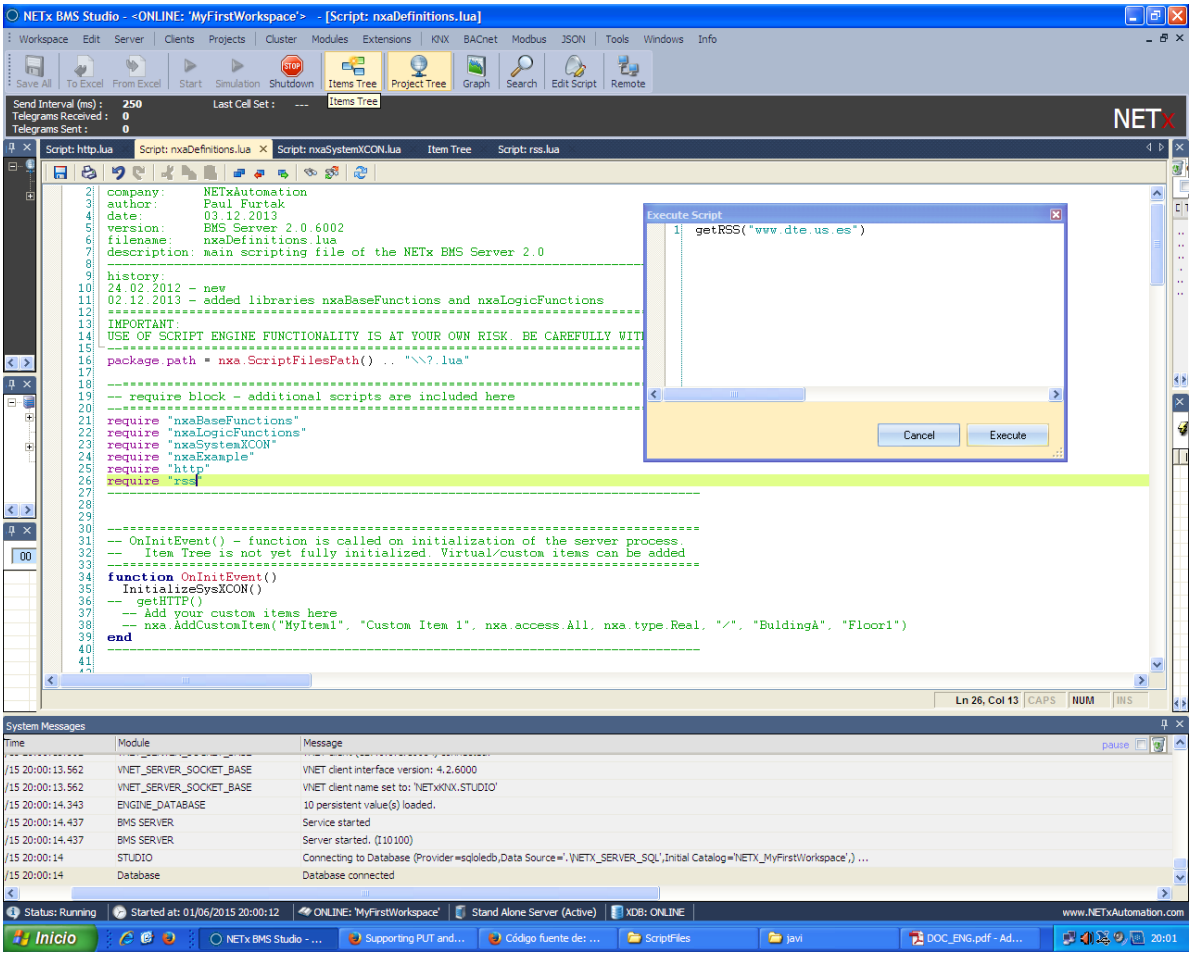
- xcon.CreateRSS(Handle): Crea una conexión RSSen LUA.
- nxa.SetValue ("rutaDelItem", valor): rutaDelItem obtenida haciendo click derecho en Enabled-> copy ítem path to clipboard

- nxa.WriteValue ("rutaDelItem", valor): Esta función escribe activamente el valor dado del item especificado. En comparación con SetValue, una escritura de valor también propagará el cambio de valor al dispositivo de campo. Un retraso opcional también podría especificarse.

- nxa.ReadValue ("ruta"): Lee el valor del campo RESULT

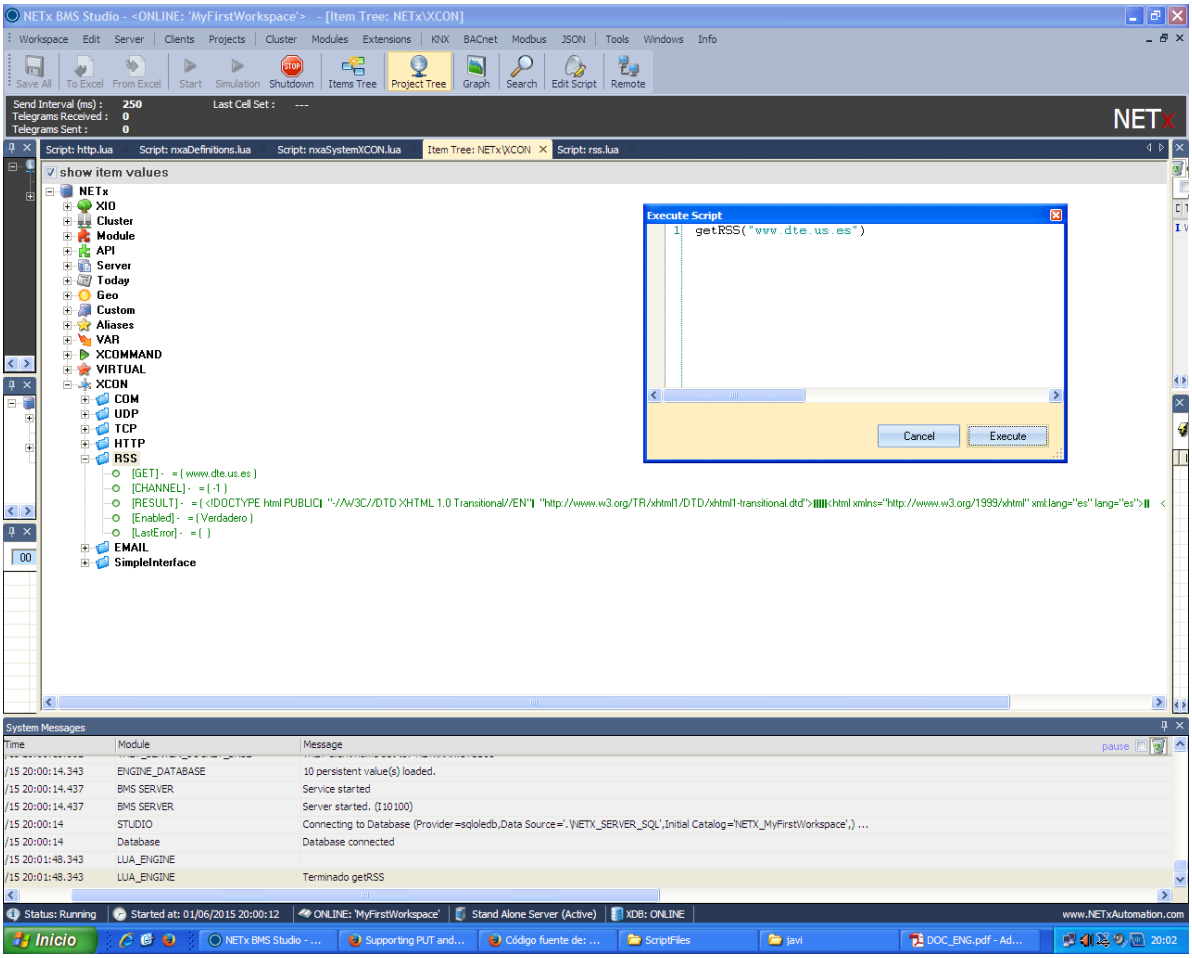
- nxa.LogInfo(lo que se desea mostrar por la consola System Messages)

También es necesario modificar el script nxaDefinitions.lua, si queremos que nuestro script pueda ser ejecutado mediante la herramienta Execute LUA script deberemos incluir el script creado en este mediante el comando require tal y como podemos observar en la siguiente captura:



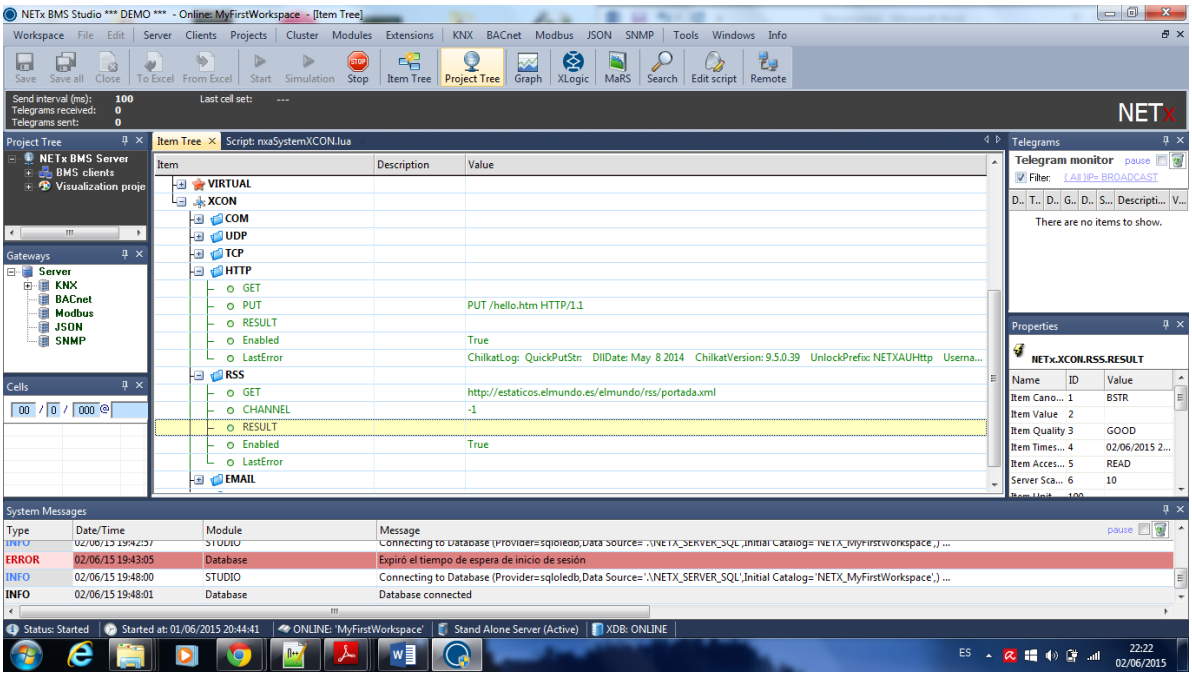
Por último, en la siguiente página mostraremos el resultado de ejecutar dicho script viendo cómo se rellenan los campos de la interfaz gráfica por sí mismos.

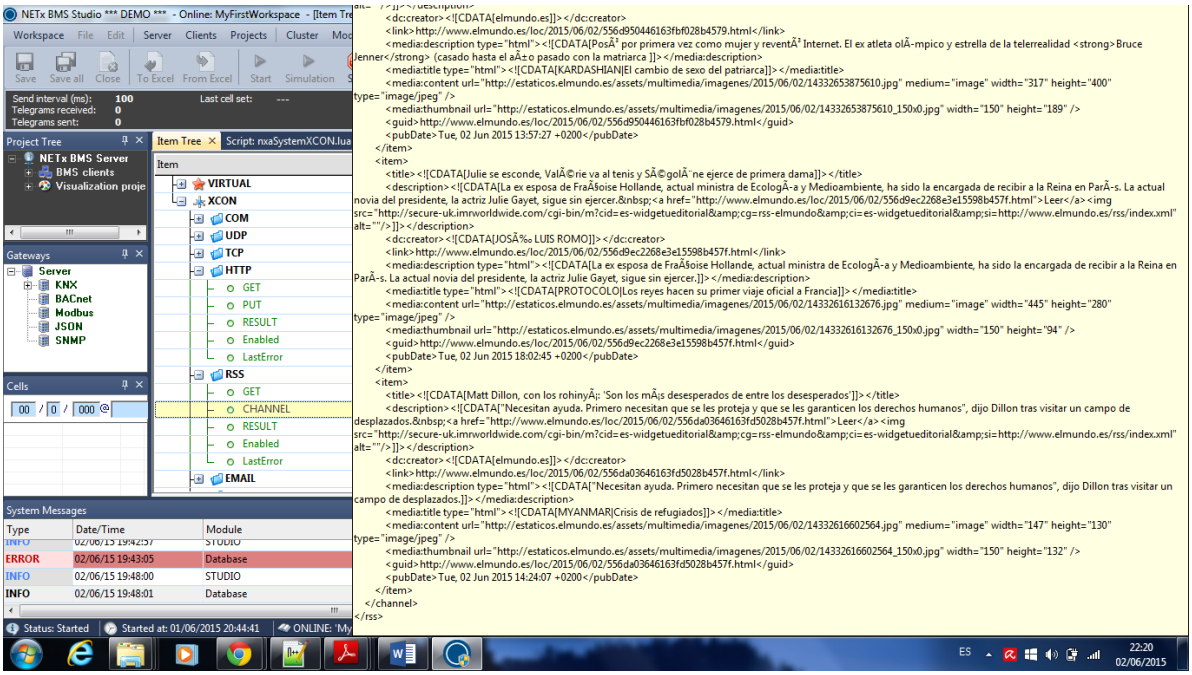
HTTP Y RSS



Por último, en la siguiente página veremos los resultados de utilizar una URL RSS y, de paso, comprobaremos el error de página amarilla en resolución desmedida (dado que ésta vez noi utilizamos la versión 2.0.7000, sino la 2.0.7210)

HTTP Y RSS





Como puede observarse en la captura, se muestra la página RSS deseada del periódico El Mundo, pero en lugar de mostrarse en el campo RESULT se muestra en esa ventana amarilla y sin permitirnos ver el contenido en su totalidad.