NXA

funciones

XCON

Contenido

En este apartado, vamos a usar los scripts de LUA, para crear comunicaciones de cara al exterior. Se podrán crear diferentes tipos, como por ejemplo crear sockets para conexiones TCP o UDP, COM, RSS, o poder crear un HTML o enviar emails a diferentes personas.

A lo largo de este documento se irá detallando los distintos métodos que hay, el código para poder ejecutarlo y si fuese posible, indicar un ejemplo de uso. Por lo que vamos a empezar a desarrollar los diferentes métodos.

Antes de nada, hay que destacar y esto es muy importante, que el código que vamos a indicar, a continuación es el TEORICO, siguiendo el tutorial, pero ese código no sirve, antes de poder ejecutar este comando, hay que hacer una estructura completa, que es el comando

–IntializeSysXCON()

Sin la estructura que hace ese comando, primeramente no podemos ver si hemos abierto los sockets, y no lo podremos ver en el itemTree.

Por lo que cuando se terminen de crear los distintos sockets, explicaré la estructura que se sigue verdaderamente para que el sistema funcione correctamente.

xcon.Create

Este método de creación de Sockets mediante LUA, es el más genérico, ya que permite, mediante un parámetro poder crear, cualquier tipo de sockets indicado anteriormente.

Este método requiere de varios parámetros de entrada para poder crearse correctamente y posteriormente, realizar acciones sobre el socket correspondiente. Los parámetros de entrada son:

- String -> Manipulador que identifica la conexión mediante LUA
- Number -> Mediante un entero, se indica el tipo de socket que se va a crear, pudiendo ser los siguientes valores:
    - 1 para crear nxa.xcon.UDP
    - 2 para crear nxa.xcon.TCPClient
    - 4 para crear nxa.xcon.HTTP
    - 5 para crear nxa.xcon.RSS
    - 6 para crear nxa.xcon.COM

- String -> La IP local, si se dejase vacío lo detecta automáticamente
- Number -> El puerto local para esta conexión, si se dejase el valor a 0, el sistema lo genera automáticamente
- String -> La IP destino a la que conectarse
- Number -> Puerto destino a la que conectarse

El código de uso sería el siguiente:

```
function OnInitEvent()
  --InitializeSysXCON()
  xcon.Create("__SysUDP",1,"",9091,"",9092)

end
```

Se ha incluido en OnInitEvent para que se ejecute cuando arranque el sistema. De esta forma tendremos un socket UDP abierto con esos datos.

xcon.CreateUDP

Con este método vamos a crear un socket UDP. Los parámetros de entrada son similares al anterior caso, lo único que cambia, es que no hay que indicarle el número, quedando por tanto así:

- String -> Manipulador que identifica la conexión mediante LUA
- String -> La IP local, si se dejase vacío lo detecta automáticamente
- Number -> El puerto local para esta conexión, si se dejase el valor a 0, el sistema lo genera automáticamente
- String -> La IP destino a la que conectarse
- Number -> Puerto destino a la que conectarse

```
function OnInitEvent()
  --InitializeSysXCON()
    xcon.CreateUDP("__Sys_UDP","",9091,"127.0.0.1",9092)
end
```

Xcon.CreateTCP

Con este método vamos a crear un socket TCP. Los parámetros de entrada son similares al anterior caso:

- String -> Manipulador que identifica la conexión mediante LUA
- String -> La IP local, si se dejase vacío lo detecta automáticamente
- Number -> El puerto local para esta conexión, si se dejase el valor a 0, el sistema lo genera automáticamente
- String -> La IP destino a la que conectarse
- Number -> Puerto destino a la que conectarse

```
function OnInitEvent()
  --InitializeSysXCON()
    xcon.CreateTCP("__Sys_TCP","",9091,"127.0.0.1",9092)
end
```

Xcon.CreateCOM

Con este método vamos a crear una conexión Serie con LUA. Los parámetros a utilizar son los siguientes:

- String -> Manipulador que identifica al puerto Serie
- String -> Nombre del COM del sistema
- Number -> Indica la velocidad de transmisión (Baudrate)
- Number -> La cantidad de bits de datos del puerto serie
- Number -> La paridad usada
- Number -> La cantidad de bits de stop
- Number -> El protocolo de handshake utilizado
- Number -> El evento de carácter usado

```
function OnInitEvent()
  InitializeSysXCON()
  xcon.CreateCOM("__SysCOM","COM1",9600,8,0,0,0,0)
end
```

xcon.CreateHTTP

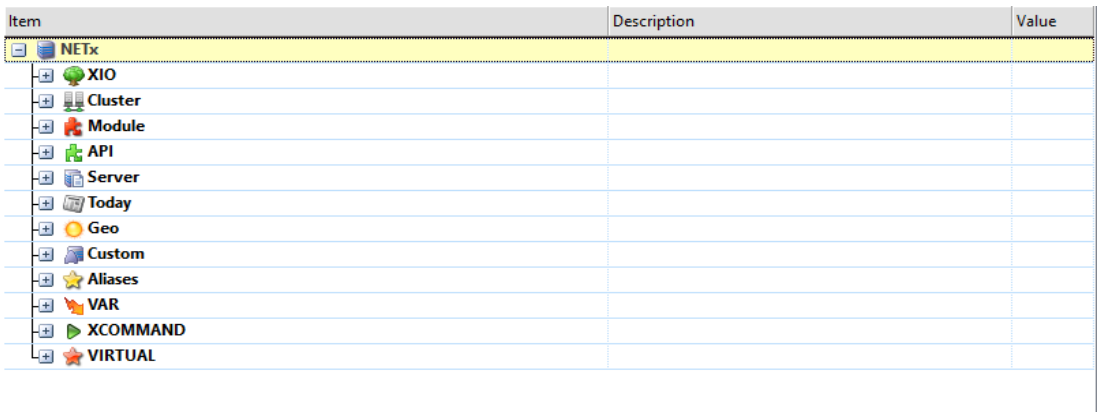
Con este método, vamos a poder crear una conexión HTTP con LUA. Para ello, solo es necesario el String correspondiente al manipulador de la conexión HTTP

xcon.CreateRSS

Con este método, vamos a poder crear un RSS con Lua, siguiente con el mismo caso que antes, solo es necesario el String correspondiente al manipulador del RSS

Aclaración sobre la creación de los diferentes sockets

Como hemos indicado anteriormente, esto por sí solo no funciona. Además si lo intentamos crear así podemos ver lo siguiente en el árbol de Ítems:



Como vemos, en este árbol de elementos, no aparece la pestaña para los XCON. Esto es debido a que no se ha creado la estructura para las comunicaciones que son las que se cargan con el

elemento que indicábamos anteriormente que estaba conectado. Un ejemplo de una estructura creada para UDP sería la siguiente:

```
71  -- UDP Subsystem
72  --------------------------------------------------------------------------------
73  function Create_SysUDP()
74    local CfgCngFunc = "__SysUDPConfigChanged()"
75    local SndFunc    = "__SysUDPSend()"
76    local EnbFunc    = "__SysUDPEnabled()"
77    sysUDP           = {}
78    sysUDP[IsCON]    = false
79    sysUDP[NM]       = "__SysUDP"
80    sysUDP[DP]       = "UDP"
81
82    sysUDP[OUT] = nxa.AddSysCustomItem(OUT, EptDESC, nxa.access.All,      nxa.type.String,
83    sysUDP[IN]  = nxa.AddSysCustomItem(IN,  EptDESC, nxa.access.Readable, nxa.type.String,
84    sysUDP[ENB] = nxa.AddSysCustomItem(ENB, EptDESC, nxa.access.All,      nxa.type.Boolean,
85    sysUDP[CON] = nxa.AddSysCustomItem(CON, EptDESC, nxa.access.Readable, nxa.type.Boolean,
86    sysUDP[LE]  = nxa.AddSysCustomItem(LE,  EptDESC, nxa.access.All,      nxa.type.String,
87
88    sysUDP[LH]  = nxa.AddExtSysCustomItem(LH, EptDESC, nxa.access.All, nxa.type.String,  tr
89    sysUDP[LP]  = nxa.AddExtSysCustomItem(LP, EptDESC, nxa.access.All, nxa.type.Integer, tr
90    sysUDP[RH]  = nxa.AddExtSysCustomItem(RH, EptDESC, nxa.access.All, nxa.type.String,  tr
91    sysUDP[RP]  = nxa.AddExtSysCustomItem(RP, EptDESC, nxa.access.All, nxa.type.Integer, tr
92
93    nxa.SetValue(sysUDP[ENB], false)
94    nxa.SetValue(sysUDP[CON], false)
95    nxa.SetValue(sysUDP[LE], "")
96    nxa.SetValue(sysUDP[LH], "")
97    nxa.SetValue(sysUDP[LP], 9091)
98    nxa.SetValue(sysUDP[RH], "")
99    nxa.SetValue(sysUDP[RP], 9092)
100
101   nxa.AddScriptTask(sysUDP[LH],  "", true, true, true, 100, CfgCngFunc)
102   nxa.AddScriptTask(sysUDP[LP],  "", true, true, true, 100, CfgCngFunc)
103   nxa.AddScriptTask(sysUDP[RH],  "", true, true, true, 100, CfgCngFunc)
104   nxa.AddScriptTask(sysUDP[RP],  "", true, true, true, 100, CfgCngFunc)
105   nxa.AddScriptTask(sysUDP[OUT], "", true, true, true, 100, SndFunc)
106   nxa.AddScriptTask(sysUDP[ENB], "", true, true, true,   0, EnbFunc)
107 end
```

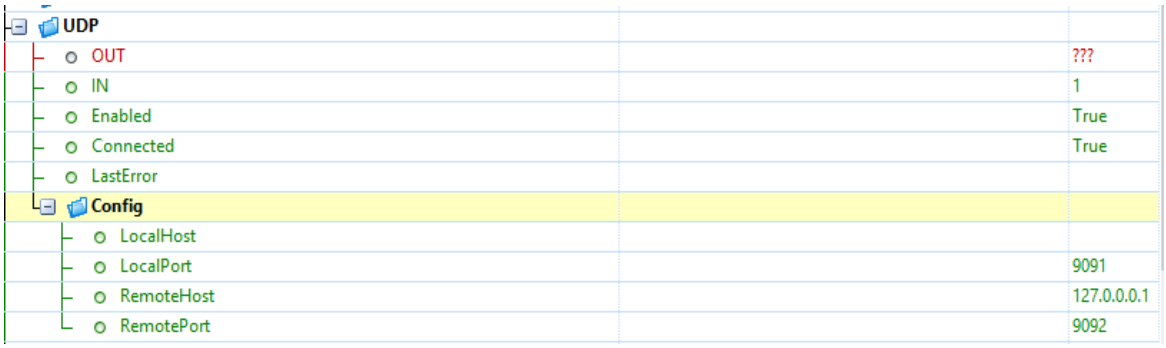
Como se ve, crea una estructura inicializada con unos valores por defectos, y sobre los campos de esta estructura es donde podemos trabajar. Es decir, en este caso por ejemplo, sysUDP. Un ejemplo puede ser el siguiente:

```
function OnInitEvent()

  InitializeSysXCON()
  nxa.SetValue(sysUDP[ENB], true)
  nxa.SetValue(sysUDP[CON], true)
  nxa.SetValue(sysUDP[RH], "127.0.0.0.1")

end
```

Partiendo de la imagen de la estructura, hemos hecho, que cuando se cree, modifique la IP destino, y active el enable y el connected para poder realizar comunicaciones, y para probarlo hemos enviado un mensaje mediante UDP, resumiéndose en lo siguiente:



Como vemos lo hemos activado correctamente y hemos recibido un mensaje, en este caso el número 1 mediante una aplicación UDP conectada al puerto 9091.

Por lo que partiendo de este ejemplo, vamos a seguir continuando con el resto de la documentación.

xcon.Close

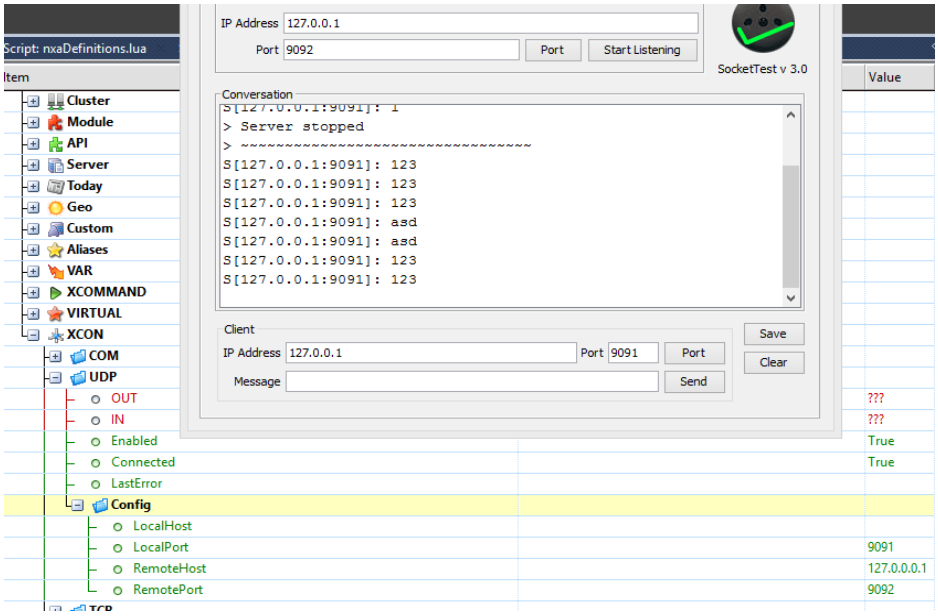
Con este método, lo que vamos a hacer es cerrar la conexión, por lo que de parámetro solo necesita un String que se corresponda con el manipulador. En este caso, vamos a mantener los valores de antes, pero, aun manteniéndose en enable y connected, es imposible recibir ningún mensaje.

```
-----------------------------------------------
function OnInitEvent()

   InitializeSysXCON()
   nxa.SetValue(sysUDP[RH], "127.0.0.0.1")
   nxa.SetValue(sysUDP[ENB], true)
   nxa.SetValue(sysUDP[CON], true)
   xcon.Close(sysUDP[NM])


end
-----------------------------------------------
```

Obteniendo lo siguiente:



Vemos, que nunca se actualiza el valor de entrada, por muchos intentos que le hagamos.

xcon.Send

Con este método lo que pretendemos es enviar un mensaje a través de una conexión establecida, siguiendo la misma pauta anterior, lo vamos a comprobar con UDP. Para ello, son necesarios los siguientes parámetros:

- String -> Manipulador del socket por el que enviaremos el mensaje
- String -> Mensaje a enviar
- Number -> Cantidad de caracteres a enviar

Además, este método devuelve un parámetro:

- Boolean -> True si el servidor está inicializado o False si no lo está

Se podría ejecutar de la siguiente forma:

```
function OnInitEvent()

  InitializeSysXCON()
  xcon.Send(sysUDP[NM],"Holaa mundooo",6)

end
```

xcon.SentText

Esta función sirve para enviar un texto a través de una conexión establecida. Lo vamos a comprobar con UDP. Para ello, son necesarios los siguientes parámetros:

- String -> Manipulador del socket por el que enviaremos el mensaje
- String -> Mensaje a enviar

Además, este método devuelve un parámetro:

- Boolean -> True si el servidor está inicializado o False si no lo está

Se podría ejecutar de la siguiente forma:

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

xcon.SendTo

Esta función, permite enviar un mensaje a un socket destino. Se necesitan los siguientes parámetros:

- String -> Manipulador de la conexión a usar
- String -> Mensaje a enviar
- Number -> Cantidad de caracteres a enviar
- String -> IP destino
- Number -> Puerto destino

Además, este método devuelve un parámetro:

- Boolean -> True si el servidor está inicializado o False si no lo está

xcon.sendTextTo

Esta función, permite enviar un mensaje a un socket destino. Se necesitan los siguientes parámetros:

- String -> Manipulador de la conexión a usar
- String -> Mensaje a enviar
- String -> IP destino

- Number -> Puerto destino

Además, este método devuelve un parámetro:

- Boolean -> True si el servidor está inicializado o False si no lo está

```
function OnInitEvent()
  InitializeSysXCON()
  xcon.SendTextTo(sysUDP[NM],"Hola","127.0.0.1",9092)
end
```

xcon.isConnected

Esta función, nos permite saber el estado de la conexión, por lo que le pasamos el manejador de la conexión y nos devuelve un booleano con el estado de la conexión.

xcon.SendEmailTo

Este método se usa para enviar un email a través de LUA. Para los 4 tipos de email, es necesario haber configurado previamente, correctamente el servicio de correo en la configuración del sistema.

Se necesitan los siguientes parámetros:

- String -> correo destino
- String -> Asunto del correo
- String -> Cuerpo del correo

Con el siguiente código, se puede enviar un correo a un destino

```
function OnInitEvent()
  InitializeSysXCON()
  xcon.SendEmailTo("luzxor@gmail.com","Prueba 1", "email")
end
```

xcon.SendAdminEmail

Este método se usa para enviar un email al administrador del sistema a través de LUA. Se necesitan los siguientes parámetros:

- String -> Asunto del correo
- String -> Cuerpo del correo

Como resultado se puede ver lo siguiente:

```
function OnInitEvent()

  InitializeSysXCON()

  xcon.SendAdminEmail("Asunto 1","Asunto 1")
  xcon.SendSystemEmail("Asunto 2","Asunto 2")
  xcon.SendUserEmail("Asunto 3","Asunto 3")

end
```

xcon.SendSystemEmail

Este método se usa para enviar un email al correo del sistema a través de LUA. Se necesitan los siguientes parámetros:

- String -> Asunto del correo
- String -> Cuerpo del correo

Como resultado se puede ver lo siguiente:

```
function OnInitEvent()

  InitializeSysXCON()

  xcon.SendAdminEmail("Asunto 1","Asunto 1")
  xcon.SendSystemEmail("Asunto 2","Asunto 2")
  xcon.SendUserEmail("Asunto 3","Asunto 3")

end
```

xcon.SendUserEmail

Este método se usa para enviar un email a un usuario previamente configurado en el sistema a través de LUA. Se necesitan los siguientes parámetros:

- String -> Asunto del correo
- String -> Cuerpo del correo

Como resultado se puede ver lo siguiente:

```
function OnInitEvent()

  InitializeSysXCON()

  xcon.SendAdminEmail("Asunto 1","Asunto 1")
  xcon.SendSystemEmail("Asunto 2","Asunto 2")
  xcon.SendUserEmail("Asunto 3","Asunto 3")

end
```