4.6.7 FUNCIONES NXA PARA CREAR ENLACES VIRTUALES E ITEMS DEL SERVIDOR

TABLA DE CONTENIDO

Contenido

4.6.7 Funciones NXA para crear enlaces virtuales e items del servidor

Estas funciones pueden ser usadas para crear items definidos por el usuario, también llamados Custom items. Estos items pueden ser organizados dentro de la jerarquía del servidor, permitiendo tener acceso completo a su funcionalidad. La diferencia es que estos Custom Items sólo están disponibles en el modelo de datos virtual del servidor, es decir que no hay un punto de datos físico donde estén conectados. Por lo tanto estos items pueden ser vistos como puntos de dato virtual.

Hay que tener en cuenta que los Custom items pueden ser solo creados y borrados cuando el servidor este iniciado o en marcha. (Ejemplo: dentro de la llamada a la función "OnInitEvent")

A continuación veremos las funciones para crear items definidos por el propio usuario, el uso y funcionalidad de cada una.

FUNCIONES:

1. **nxa.AddCustomItem (string, string, nxa.access, nxa.type, string, string…):**

   Esta función nos permite añadir un nuevo Custom items durante la fase de inicialización del servidor, y devuelve un ID, que identifica al item.
   Esta función sólo puede ser usada dentro de la llamada a la función "OnInitEvent"

   Parámetros que recibe la función son los siguientes:

   - Recibe como primer parámetro una cadena que indica el nombre que se le desea poner al elemento que estamos creando.
   - Recibe como segundo parámetro una cadena que describe el elemento personalizado.
   - Recibe como tercer parámetro del tipo nxa.access para indicar el tipo de acceso al item. Los accesos pueden ser de los siguientes tipos:
     - nxa.access.Readable : tipo de acceso de sólo lectura
     - nxa.access.Writeable: tipo de acceso de sólo escritura
     - nxa.access.All: tipo de acceso de lectura y escritura.

   - Recibe como cuarto parámetro de tipo nxa.type para indicarle el tipo de dato del elemento. Los tipos a elegir son los siguientes:
     - nxa.type.Integer: tipo integer
     - nxa.type.Real: tipo números reales
     - nxa.type.Date: tipo fecha
     - nxa.type.String: tipo cadena
     - nxa.type.Boolean: tipo booleano, verdadero o falso.

   Como parámetros opcionales tenemos:

- Tipo String "delimitador": que indica el delimitador que se utiliza para construir el identificador del item.
- Tipo String "path": para indicar la ruta que almacena el elemento personalizado. Se pueden añadir más de una ruta.

Esta función devuelve el identificador "ID" del elemento que hemos creado.

Implementación:

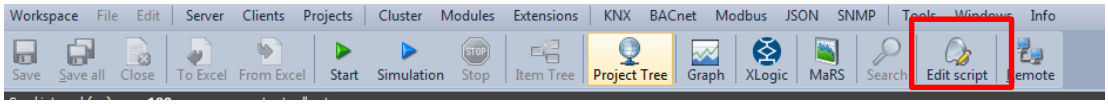
Nxa.AddCustomItem *("Elemento1", "descripción del elemento", nxa.access.All, nxa.type.Real, "/", "Ruta1", "Ruta2")*

Para la ejecución de la función, el BMS Server nos proporciona unos scripts, que contiene las funciones a ejecutarse. El fichero principal que se va a utilizar es el "nxaDefinitions.lua", el cual contiene la función "OnInitEvent" donde vamos a llamar a las funciones de creación de items.

El fichero se encuentra en la siguiente ruta:

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

También desde el BMSServer tenemos uno opción ***"Editor Script"*** -> ***ScriptFiles***

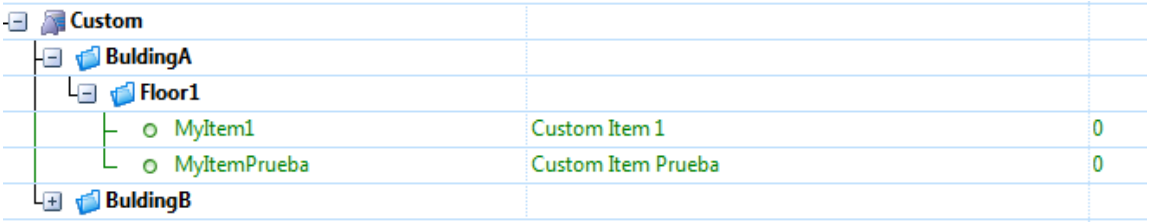


Una vez abierto el archivo "*nxaDefinitions*" dentro de la función "*OnInitEvent*" creamos nuestro item de la siguiente forma:

```
function OnInitEvent()
  InitializeSysXCON()
  Example()

  -- Add your custom items here
  nxa.AddCustomItem("MyItem1",
   "Custom Item 1",
    nxa.access.All,
    nxa.type.Real,"/",
    "BuldingA",
    "Floor1")

  end
```

En este ejemplo hemos creado un nuevo Custom Item llamado "MyItem1" que se encuentra dentro del directorio: "Custom/BuldingA/Floor1/MyItem1"

Para comprobar que se ha creado el Custom Item en el árbol de items podemos verlo dentro de la opción Custom, de la siguiente forma:



2. nxa.AddExtCustomItem (string, string, nxa.access, nxa.type, bool, bool, bool string, string…):

Esta función añade un nuevo elemento personalizado durante la fase de inicialización del servidor y devuelve el identificador del elemento creado, de la misma forma como lo hace la anterior función. Pero a diferencia de la anterior *"Nxa.AddCustomItem"* se puede añadir una bandera, la cual se puede configurar como "Persistente", "Histórico" y "Sincronizado". El parametro "**Persistent**" nos indicara si el valor del item es persistente, es decir que es restaurado desde la base de datos después de la puesta en marcha del servidor; el parametro "**Historical**" especifica si los valores históricos del item se almacenan en la base de datos o no; y el parametro "**Synchronize**" indica que el valor se sincroniza entre el servidor principal y el de respaldo si se establece como "TRUE".

Parámetros de entrada:

- Recibe como primer parámetro una cadena que indica el nombre que se le desea poner al item que estamos creando.
- Recibe como segundo parámetro una cadena que describe el Custom item.
- Recibe como tercer parámetro del tipo nxa.access para indicar el tipo de acceso al item.
- Recibe como cuarto parámetro de tipo nxa.type para indicarle el tipo de dato del elemento.
- Recibe un quinto parámetro un tipo booleano (true o false), que indica si es "Persistent" o no.
- Un sexto parámetro de tipo booleano (true o false), para establecer si es de tipo "Historical"
- Un séptimo parámetro de tipo booleano (true o false), para establecer si es de tipo "Synchronize"

Como parámetros opcionales tenemos:

- Tipo String "delimitador": que indica el delimitador que se utiliza para construir el Identificador del elemento.
- Tipo String "path": para indicar la ruta que almacena el elemento personalizado. Se pueden añadir más de una ruta.

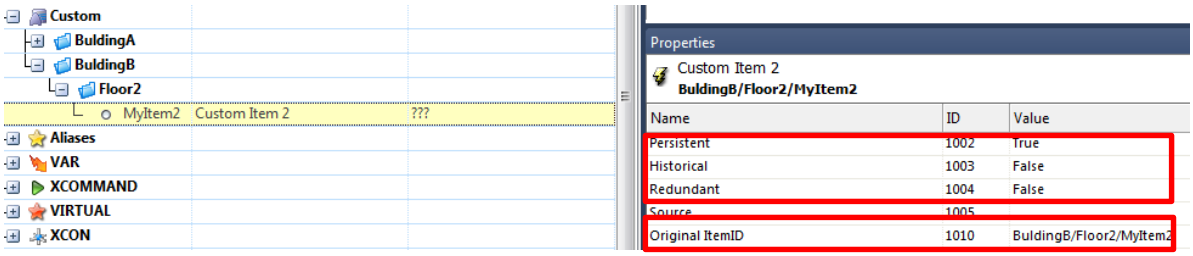
Implementación:

Para implementar esta función, igual que la anterior, dentro de la función "*OnInitEvent*" que se encuentra dentro del script *nxaDefinitions* la añadimos de la siguiente forma:

```
nxa.AddExtCustomItem("MyItem2",
 "Custom Item 2",
 nxa.access.All,
 nxa.type.Real,
 true, false,
 false,"/",
 "BuldingB", "Floor2")
```

En este ejemplo se ha creado un nuevo item llamada MyItem2 el cual solo tiene la bandera activada a "Persistent" y las demás desactivas.

Para comprobarlo podemos verlo en el árbol de items



También es posible ver las propiedades de los items, haciendo click derecho sobre el item creado y en la opción "*show properties*", y ver el itemID que nos devuelve, que en este caso es:

BuldingB/Floor2/MyItem2

Otra forma de ver o coger el itemID se hace haciendo click derecho sobre el item y "*copy item path to clipboard*"

3. nxa.AddSysCustomItem (string, string, nxa.access, nxa.type, string, number, number, string, string…):

Esta función permite añadir un nuevo item del sistema durante la fase de inicialización del servidor y devuelve un identificador del elemento (ID). En comparación con los anteriores, con esta función un Custom items del sistema nuevo se puede colocar en cualquier lugar del árbol de items (Item Tree).
En general esta función debe ser usada en la llamada a la función "*OnInitEvent*"

Parámetros:

- Recibe como primer parámetro una cadena que indica el nombre que se le desea poner al item que estamos creando.
- Recibe como segundo parámetro una cadena que describe el Custom item.
- Recibe como tercer parámetro del tipo nxa.access para indicar el tipo de acceso al item.

4.6.7

- Recibe como cuarto parámetro de tipo nxa.type para indicarle el tipo de dato del item.
- Recibe un parámetro de tipo cadena que describe las unidades del Custom item.
- Recibe un parámetro de tipo number donde se añade el valor mínimo permitido para el Custom item. (opcional)
- Recibe un parámetro tipo number que describe el máximo valor permitido para el Custom item (opcional)

Como parámetros opcionales tenemos:

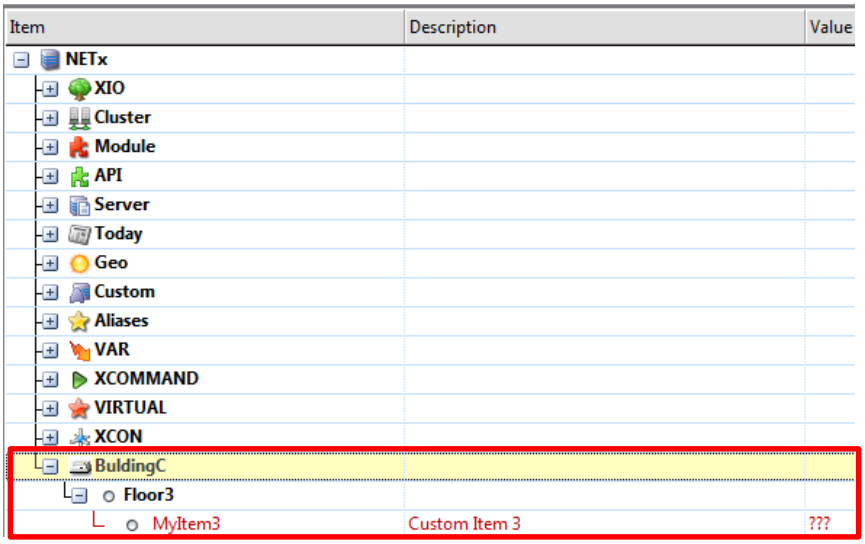
- Tipo String "delimitador": que indica el delimitador que se utiliza para construir el Identificador del item.
- Tipo String "path": para indicar la ruta que almacena el Custom item. Se pueden añadir más de una ruta.

Implementación:

Dentro de la función OnInitEvent añadimos lo siguiente:

```
nxa.AddSysCustomItem("MyItem3",
 "Custom Item 3",
  nxa.access.All,
  nxa.type.Real,
  "uno",1,3,"/",
  "BuldingC", "Floor3")
```

Y podemos comprobar dentro de la jerarquía de nuestro árbol de items que se ha añadido un nuevo item:



4. nxa.AddExtSysCustomItem (string, string, nxa.access, nxa.type, bool, bool, bool, string, number, number, string, string…):

Esta función permite añadir un nuevo elemento del sistema durante la fase de inicialización del servidor y devuelve el identificador correspondiente a ese elemento (ID). A diferencia del anterior, en esta función podemos configurar las banderas como "Persistente", "Histórica", y "Sincronizada."

Parámetros de entrada:

- Recibe como primer parámetro una cadena que indica el nombre que se le desea poner al item que estamos creando.
- Recibe como segundo parámetro una cadena que describe el Custom item.
- Recibe como tercer parámetro del tipo nxa.access para indicar el tipo de acceso al item.
- Recibe como cuarto parámetro de tipo nxa.type para indicarle el tipo de dato del item.
- Recibe un quinto parámetro un tipo booleano (true o false), que indica si es "Persistent" o no.
- Un sexto parámetro de tipo booleano (true o false), para establecer si es de tipo "Historical"
- Un séptimo parámetro de tipo booleano (true o false), para establecer si es de tipo "Synchronize

Como parámetros opcionales tenemos:

- Recibe un parámetro de tipo cadena que describe las unidades del Custom item.
- Recibe un parámetro de tipo number donde se añade el valor mínimo permitido para el Custom item. (opcional)
- Recibe un parámetro tipo number que describe el máximo valor permitido para el Custom item (opcional)
- Tipo String "delimitador": que indica el delimitador que se utiliza para construir el Identificador del elemento.
- Tipo String "path": para indicar la ruta que almacena el elemento personalizado. Se pueden añadir más de una ruta.

Implementación:

Se implementa de la misma forma que el anterior pero con las siguientes modificaciones:

```
nxa.AddExtSysCustomItem ("MyItemSys",
 "Custom Item Sys",
  nxa.access.All,
  nxa.type.Real,
  true, false, false,
  "uno",1,3,"/",
  "BuldingD", "Floor2")
```

5. nxa.AddItemLink (string. string, number)

Esta función enlaza un item origen a un item destino. De tal forma que el valor del elemento origen es enviado al item destino con un retraso dado.

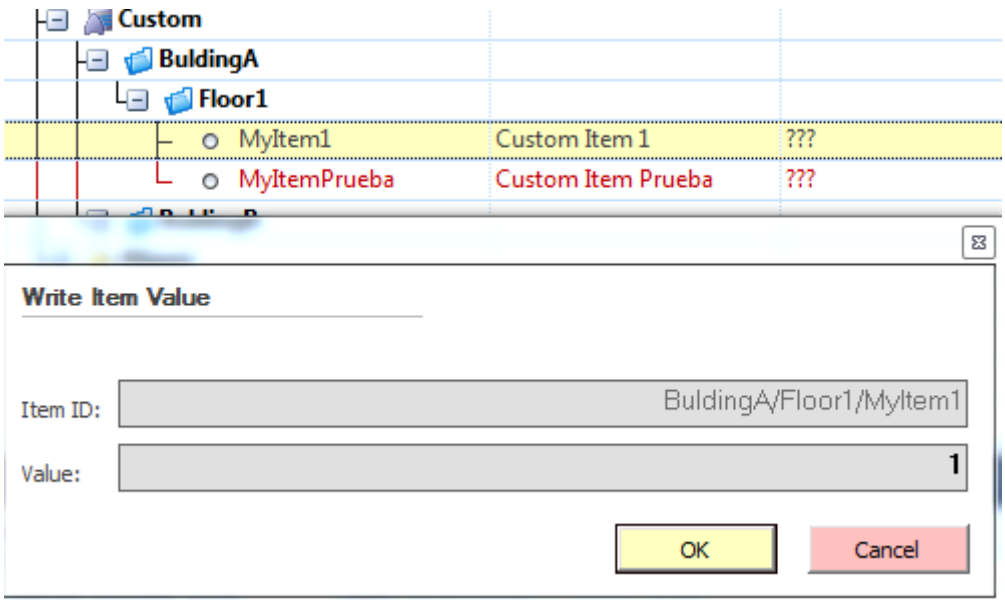
Parámetros de entrada:

- El primer parámetro indica el idenficador del item origen.
- El segundo parámetro indica el identificador del item destino.
- El tercer parámetro indica el retraso en milisegundos de enviar desde origen a destino.

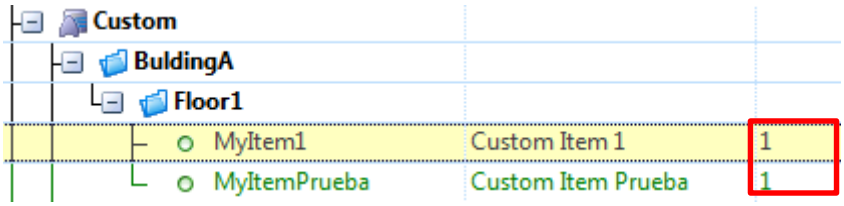
Implementación:

```
function OnInitEvent()
  InitializeSysXCON()

 nxa.AddItemLink("BuldingA/Floor1/MyItem1","BuldingA/Floor1/MyItem2",100)

end
```

Para comprobar que funciona, dentro del árbol de items, escribimos un valor en el item origen y se escribe ese valor al item destino de la siguiente forma



Y ahora se muestra el valor en los dos items:



6. nxa.AddDirectLink (string, string)

Esta función enlaza de forma directa un item origen con uno destino, es decir que el valor del item destino cambia en el mismo instante el que el del ítem origen es cambiado.

Parámetros de entrada:

- El primer parámetro indica el idenficador del elemento origen.
- El segundo parámetro indica el identificador del elemento destino.

Implementación:

```
function OnInitEvent()
  InitializeSysXCON()

 nxa.AddDirectLink("BuldingA/Floor1/MyItem1","BuldingA/Floor1/MyItem2")

end
```

7. nxa.RemoveDirectLink (string, string)

Esta función nos permite borrar un enlace directo entre un elemento origen y uno destino.

Parámetros de entrada:

- El primer parámetro indica el idenficador del elemento origen.
- El segundo parámetro indica el identificador del elemento destino.

Implementación:

```
function OnInitEvent()
  InitializeSysXCON()
 nxa.RemoveDirectLink("BuldingA/Floor1/MyItem1","BuldingA/Floor1/MyItem2")


end
```

8. nxa.AddDirectBLink (string, string)

Esta función nos permite hacer un enlace bidireccional, es decir que el valor se puede transmitir de origen a destino y viceversa.

Parámetros de entrada:

- El primer parámetro indica el idenficador del elemento origen.
- El segundo parámetro indica el identificador del elemento destino.

Implementación:

```
function OnInitEvent()
  InitializeSysXCON()
nxa.AddDirectBiLink("BuldingA/Floor1/MyItem1","BuldingA/Floor1/MyItem2")

end
```

9. nxa.RemoveDirectBiLink (string, string)

Esta función nos permite borrar los enlaces bidireccionales entre ítems origen y destino.

Parámetros de entrada:

- El primer parámetro indica el idenficador del elemento origen.
- El segundo parámetro indica el identificador del elemento destino.

Implementación:

```
function OnInitEvent()
  InitializeSysXCON()
nxa.RemoveDirectBiLink("BuldingA/Floor1/MyItem1","BuldingA/Floor1/MyItem2")

end
```

10. nxa.AddItemEvent (string, bool, bool, bool, number, string)

Esta función permite añadir un evento, el cual se activara cada vez que el elemento cambie.

Parámetros de entrada:

- El primer parámetro indica el identificador del elemento origen, el cual activara el evento.
- El segundo parámetro dependiendo del valor que le demos-false, true-el ítem será activado o no si ha recibido una petición de lectura.
- El tercer parámetro dependiendo del valor que le demos-false, true-el ítem será activado o no si escribimos en él.
- El cuarto parámetro dependiendo del valor que le demos-false, true-el ítem será activado o no si se ha modificado el valor.
- El quinto parámetro nos indica el tiempo que tarda en ejecutarse el evento una vez se realice la acción sobre el ítem. Está en milisegundos.
- En el último parámetro ponemos la función que ejecuta el evento. Debe tener en cuenta que como este parámetro es de tipo string debe ir entre comillas.

Implementación:

```
function OnInitEvent()
  InitializeSysXCON()
nxa.AddItemEvent("BuldingA/Floor1/MyItem1",false,true,false,10000,"HelloWorld()")

end
```

Una vez que cambiemos el valor en el item se ejecuta el script del evento que hemos creado, en este caso nuestro script es HelloWorld y va a escribir por consola HelloWorld.

11. nxa.AddItemTemplate ()

Esta función nos permite crear una nueva plantilla para los items personalizados que tenemos creados, y los añade a la lista de plantillas dentro del servidor.

Implementación:

```
1  function template()
2
3      return  nxa.LogInfo(nxa.AddItemTemplate())
4
5  end
```

Devuelve el identificador de la plantilla, es decir el ID que identifica de forma exclusiva la plantilla.

12. nxa.AddPropertyTemplate (number, number, string, nxa.type, variant)

Una vez tengamos creada la plantilla con la función vista anteriormente, con esta función podemos añadir nuevas propiedades a la plantilla.

Parámetros:

- El primer parámetro que recibe esta función es el identificador de la plantilla de tipo numérico, creado anteriormente con la función anterior.
- El segundo parametro le pasamos el Id que identifica la propiedad dentro de la plantilla. El rango se comprende entre 7000-7999
- El tercer parámetro indica el nombre de la propiedad, por tanto es de tipo cadena.
- El cuarto parámetro indica el tipo de la propiedad. De tipo nxa.type.
- El último parámetro indica el valor por defecto de la propiedad.

Implementación:

```
function OnInitEvent()
  InitializeSysXCON()

 nxa.AddPropertyTemplate(nxa.AddItemTemplate(),7001,"template1",nxa.type.Real,11)

 end
```

13. nxa.AttachTemplate (string, number)

Con esta función asigna una plantilla a un custom item.

Parámetros:

- En el primer parámetro indica el identificador del item a enlazar.
- En el segundo parámetro ponemos el identificador del template que queremos enlazar.

Implementación:

```
function OnInitEvent()
  InitializeSysXCON()

 nxa.AttachTemplate("BuldingA/Floor1/MyItem1",39)
 end
```

14. nxa.DetachTemplate (string)

Esta función separa una plantilla unida a un item.

Parámetros:

Recibe un único parametro de tipo string que indica el identificador del item en la plantilla que se le ha sido asignada.

Implementación:

```
function OnInitEvent()
  InitializeSysXCON()

 nxa.DetachTemplate("BuldingA/Floor1/MyItem1")
 end
```