NXA
EVENT
CALLBACKS

4.6.10. NXA EVENT CALLBACKS

Como complemento a las múltiples funciones de LUA, la librería XNA LUA también provee de múltiples eventos de callback o auto-cargado en español, los cuales son invocados por el servidor en específicos momentos de tiempo. Con estos callbacks, el usuario puede integrar Scripts de LUA que son ejecutados cuando el servidor invoca al callback correspondiente. Usando este modo, se puede implementar una funcionalidad de control que deberá estar disponible en momentos concretos de tiempo.

- **OnInitEvent**

Este callback es invocado o llamado durante la inicialización del servidor. En este callback podemos ejecutar tareas de inicialización, es importante destacar que a la hora de ejecutar esta función, los ítems aun no están creados y por tanto no están disponibles todavía, por ejemplo:

```
function OnInitEvent()
  -- Aquí inicializamos una serie de items

-- Cambiamos el estado de una variable a true (podría ser un led
que indica el estado del servidor)

     nxa.SetValue("NETx\\VAR\\Boolean\\Item1", true)

     nxa.WriteValue("NETx\\XIO\\KNX\\192.168.1.2\\03/0/001",
true)


  -- Añadimos funciones lógicas del módulo nxaLogicFunctions
aquí

     ADD_OR("NETx\\VAR\\Boolean\\Item15",
"NETx\\VAR\\Boolean\\Item11", "NETx\\VAR\\Boolean\\Item12",
"NETx\\VAR\\Boolean\\Item13", "NETx\\VAR\\Boolean\\Item14")

End
```

- **OnStartEvent**

Este callback es llamado durante la puesta en marcha del servidor. Una vez se haya ejecutado “OnStartEvent” y posteriormente también se haya ejecutado el evento “OnInitEvent”, todos los elementos ya estarán disponibles y se pueden acceder a través de scripts LUA , por ejemplo:

```
function OnStartEvent()

  -- Cambiamos el estado de una variable a true (podría ser un
led que indica el estado del servidor)

     nxa.SetValue("NETx\\VAR\\Boolean\\Item1", true)

end
```

- **OnStopEvent**

Este callback es llamado durante el proceso de shutdown del servidor, por ejemplo:

```
function OnStopEvent()

  -- Cambiamos el estado de una variable a false (podría ser un
led que indica el estado del servidor)

     nxa.SetValue("NETx\\VAR\\Boolean\\Item1", false)

end
```

- **OnSecondTimerEvent**

Este callback es llamado cada segundo. Una de sus posibles utilidades es que puede ser usado para implementar una funcionalidad que tenga que ser ejecutada cada segundo, por ejemplo:

```
function OnSecondTimerEvent()

  -- Comprobamos cada segundo que la bomba (Item2) está
funcionando, ya que de apagarse podría tener consecuencias
nefastas

     if nxa.GetValue("NETx\\VAR\\Boolean\\Item2")
     then
       print (“Error la bomba esta apagada”)
     end
end
```

- **OnMinuteTimerEvent**

Este callback es llamado cada minuto. Una de sus posibles utilidades es que puede ser usado para implementar una funcionalidad que tenga que ser ejecutada cada minuto, por ejemplo:

```
function OnMinuteTimerEvent()

  -- Comprobamos cada minuto que la bomba (Item2) está
funcionando, ya que de apagarse podría tener consecuencias
nefastas

     if nxa.GetValue("NETx\\VAR\\Boolean\\Item2")
     then
       print (“Error la bomba esta apagada”)
     end

end
```

- **OnHourTimerEvent**

Este callback es llamado cada hora. Una de sus posibles utilidades es que puede ser usado para implementar una funcionalidad que tenga que ser ejecutada cada hora, por ejemplo:

```
function OnMinuteTimerEvent()

  -- Comprobamos cada hora que la bomba (Item2) está
funcionando, ya que de apagarse podría tener consecuencias
nefastas

      if nxa.GetValue("NETx\\VAR\\Boolean\\Item2")
      then
        print ("Error la bomba esta apagada")
      end

end
```

- **OnKNXGatewayConnectedEvent**

Este callback es llamado cada vez que una gateway de KNX ha sido conectada. El parámetro que se le pasa a la función puede ser usado para determinar que la acción se ejecute para una gateway determinada, por ejemplo:

Parámetros:
- string – Dirección IP de la entrada KNX.

```
function OnKNXGatewayConnectedEvent(ipaddress)

  -- Indicamos que ha sido conectado un dispositivo con IP =
ipaddress

      print ("Ha sido conectado un dispositivo en el sistema con IP =
".. ipaddress)

  -- Si conocemos el dispositivo, decimos cual es, por ejemplo
climatización aula 2.11

      if ipaddress == variableLocalConIP
      then
        print ("El dispositivo X ha sido conectado")
      end

end
```

- **OnKNXGatewayDisconnectedEvent**

Este callback es llamado cada vez que una gateway de KNX ha sido desconectada. El parámetro que se le pasa a la función puede ser usado para determinar que la acción se ejecute para una gateway determinada, por ejemplo:

Parámetros:
- string – Dirección IP de la entrada KNX.

```
function OnKNXGatewayDisconnectedEvent(ipaddress)

  -- Indicamos que ha sido desconectado un dispositivo con IP =
ipaddress

      print ("Ha sido desconectado un dispositivo en el sistema con IP
= ".. ipaddress)

  -- Si conocemos el dispositivo, decimos cual es, por ejemplo
climatización aula 2.11

      if ipaddress == variableLocalConIP
      then
        print ("El dispositivo X ha sido desconectado")
      end

end
```

- **nxa.OnClientConnectedEvent**

Esta función es invocada cuando un cliente es conectado al Servidor BMS de NETx, por ejemplo:

Parámetros:
- clientType – Tipo de cliente (WEB, VNET, or OPC)
- clientName – Nombre de cliente
- source – De donde viene el cliente (e.g. Voyager.5.0, OPC Client name or IP address of BMS Client)
- IPAddress – Dirección IP del cliente
- user – Usuario que está actualmente online.

```
function OnClientConnectedEvent(clientType, clientName, source,
IPAddress, user)

  -- Le damos la bienvenida al usuario que se ha conectado

      print ("Bienvenido "..clientName)

end
```

- **nxa.OnClientDisconnectedEvent**

Esta función es invocada cuando un cliente es desconectado del Servidor BMS de NETx, por ejemplo:

Parámetros:

- clientType – Tipo de cliente (WEB, VNET, or OPC)
- clientName – Nombre de cliente
- source – De donde viene el cliente (e.g. Voyager.5.0, OPC Client name or IP address of BMS Client)
- IPAddress – Dirección IP del cliente
- user – Usuario que está actualmente online.

```
function OnClientDisconnectedEvent(clientType, clientName,
source, IPAddress, user)

  -- Nos despedimos del usuario que se va a desconectar

     print ("Hasta pronto "..clientName)

end
```