Elvis Custom Port

Whitepaper zu Elvis Custom Port
Version 1.0 vom 17.02.2006 zum Custom Port vom 14.02.2006 (elviscpd.dll)

Allgemeines

Der Custom Port ist ein Anschluss für die Kommunikation mit Geräten oder Programmen (Fremdsystem) über IP Protokolle oder V24.
Die Implementierung des fremden Protokolls erfolgt in einem Script. Dieses Script enthält alle für die Kommunikation notwendigen Einstellungen für den Aufbau der Kommunikation sowie die Anweisungen für die Verarbeitung eingehender und ausgehender Sendungen (Telegramme). Damit liegen alle wesentlichen Operationen in unserer Hand, dennoch ist die Erstellung eines Scripts nicht schwierig. Spezielle Funktionen (elviscpd.dll) für den Datenaustausch mit dem Prozessserver und die Kommunikation mit dem Fremdsystem machen uns die Arbeit leicht! Der Custom Port besteht also aus der elviscpd.dll und unserem Skript.

Zwei Schnittstellen

Im Skript werden zwei Systeme, die sich nicht kennen, verbunden. Dazu werden zwei Schnittstellen benutzt. Die Schnittstelle zum Prozessserver und die Schnittstelle zum Fremdsystem. Alle Funktionen die zur Kommunikation mit dem Prozessserver und dem Fremdsystem erforderlich sind, werden von der elviscpd.dll bereitgestellt. Der Prozessserver kommuniziert nur mit der elviscpd.dll. Dennoch kann von zwei unterschiedlichen Schnittstellen gesprochen werden, denn die zur Kommunikation verwendeten Funktionen unterscheiden sich. Nachfolgend wird deshalb auch von der Schnittstelle zum Prozessserver und von der Schnittstelle zum Fremdsystem gesprochen.

Die Schnittstelle zum Prozessserver

Wenn der Prozessserver startet wird auch der Custom Port gestartet. Damit Initialisierungen erfolgen können oder eine Verbindung aufgebaut werden kann wird OnOpen aufgerufen. Beim Beenden des Servers wird OnClose aufgerufen.

OnOpen(sParam As String)

Aufruf beim Start des Prozessservers, dient zur Initialisierung des Ports. Wenn ein Verbindungsaufbau zum Fremdsystem beim Starten des Prozessservers gewünscht ist oder ein Socket Server eingerichtet werden soll ist dies hier möglich. Siehe: Schnittstelle zum Fremdsystem.

Parameter:

sParam enthält die Parameter die beim Datenpunkt-Anschluss mit OpenParam="<Parameter>" angegeben sind.

OnClose()

Aufruf beim Beenden des Prozessservers, dient zum Aufräumen des Ports.

Wenn die elvissvr.dll vom Prozessserver zum Senden oder Abfragen von Daten aufgefordert wird, gibt sie diese Aufforderung an das Script weiter. Dazu sind spezielle Funktionen definiert:

SendValue(address As String, value As Variant, svrinfo As String)

Aufruf wenn sich der NominalValue des Datenpunkts mit der angegebenen Adresse geändert hat.

Parameter:

address ist die Adresse

value der Wert

svrinfo die Zusatzinfo (aus der Liste der Datenpunkt-Typen)

RequestValue(address As String, svrinfo As String)

Aufruf wenn der ActualValue des Datenpunkts mit der angegebenen Adresse benötigt wird.

Parameter:

address ist die Adresse

svrinfo die Zusatzinfo (aus der Liste der Datenpunkt-Typen)

Wenn ein Wert zu einer Adresse vorliegt, und dieser an den Prozessserver weitegeleitet werden soll damit der ActualValue des Datenpunkts mit dieser Adresse gesetzt wird ist FireReceiveValue aufzurufen.

FireReceiveValue address, value

Aufruf im Skript wenn der ActualValue eines Datenpunkts gesetzt werden soll.
Parameter:

address ist die Adresse (String)

value der Wert (Variant)

Für Protokolle die ein bestimmtes Timing-Verhalten benötigen, z.B. ein Lebenszeichen in einem festen Intervall zu senden, sind die Funktionen OnTimer und SetTimer zu verwenden.

SetTimer intervall

Zum setzen des Timer-Intervalls. Im Angegebene Intervall wird die Funktion OnTimer aufgerufen.
Parameter:

intervall ist das Intervall in Millisekunden (Long)

OnTimer()

Aufruf wenn der Timer abgelaufen ist. Start des Timers mit SetTimer.

Log text

Der Text wird in die Elvis-Logdatei (elvissvr.log) geschrieben
Parameter:

text (string)

Die Schnittstelle zum Fremdsystem über V24

OpenCommPort port, mode

Zum Aufbau der Kommunikation und Einstellen der Kommunikation.
Parameter:

port der Com-Port z.B. COM1 (String)
mode die Kommunikationseinstellungen für den Port (String)

baud=<Baudrate>
parity=n|e|o|m|s
data=5|6|7|8
stop=1|1,5|2
to=on|off
xon=on|off
odsr=on|off
octs=on|off
dtr=on|off|hs
rts=on|off|hs|tg
idsr=on|off

Beispiel: OpenCommPort "COM1", "Baud=2400 Parity=E Data=7 Stop=1"

CloseCommPort

Schließt die Verbindung.

CommSend sSend

Zum Senden einer Zeichenkette (sSend) an das Fremdsystem.

Parameter:

sSend die Zeichenkette die gesendet werden soll (String)

OnReceive(ByteData As Byte)

Aufruf wenn Daten empfangen wurden. Der Aufruf erfolgt für jedes einzelne Byte. Das Byte wird als Zahl (0-255) übergeben. Mit diesem Verfahren können alle Binärwerte empfangen werden.

Parameter:

ByteData enthält ein Daten-Byte (8 Bit -> Wertebereich: 0-255)

Hinweise zur Benutzung:

Damit die Telegramme ausgewertet werden können müssen die Daten zuerst gesammelt werden. Das geschieht am besten in einem Array of Byte. Immer wenn neue Daten empfangen wurden wird geprüft ob ein vollständiges telegramm im Array vorliegt. Wenn das Telegramm vollständig ist wird es Ausgewertet und aus dem Array gelöscht.

svrinfo = QueryServerInfo address

Zum Abfragen der Zusatzinfo (aus der Liste der Datenpunkt-Typen).

Parameter:

address die Adresse des Datenpunkts zu dem die Zusatzinfo benötigt wird (String)

Rückgabewert:

svrinfo die Zusatzinfo zum Datenpunkt (String)

Die Schnittstelle zum Fremdsystem über IP (TCP, UDP)

CreateSocketServer params

Zum Erzeugen eines Socket Servers.
Parameter:

params enthält die Kommunikationseinstellungen

Port=<PortID>
Protocol=<Protokoll> () mögliche Protokolle sind: TCP|UDP|ICMP|RAW default ist TCP
Timer=<Intervall> das Intervall (in ms) in dem nach eingehenden Paketen für den Port gesehen wird, default ist 1000

DestroySocketServer

Zum Abbau des Socket-Servers.

Die Schnittstelle zum Fremdsystem über TCP

ConnectSocketClient addr, port, options

Parameter

addr	die IP-Adresse des Servers (string)
port	die Port ID der Socketverbindung (integer)
options	Verbindungsoptionen, das Trennzeichen zwischen den Optionen ist der Strichpunkt (string)

DisconnectSocketClient

Zum Abbau des Verbindung zum Socket-Server.

OnReceive(data As String)

Aufruf wenn Daten empfangen wurden.
Parameter

data	Die empfangenen Daten (string)

SocketSend(sSend)

Zum Senden einer Zeichenkette (sSend) an das Fremdsystem.

Parameter:

sSend die Zeichenkette die gesendet werden soll (String)

Rückgabe:

bRes Rückgabewert ob das Sender erfolgreich war (boolean)

OnConnect()

Wird beim Verbindungsaufbau aufgerufen (sowohl auf der Server wie auf der Client-Seite), dient zur Initialisierung der Socket Connection (z.B. Quality of Service etc.).

OnDisconnect

Wird beim Verbindungsabbau aufgerufen. Dient zum Aufräumen des Ports.

Die Schnittstelle zum Fremdsystem über UDP

OnReceiveFrom(addr As String, port As Long, data As String)

Aufruf wenn Daten empfangen wurden.
Parameter

addr	die IP-Adresse des Senders (string)
port	die port-ID (Long)
data	Die empfangenen Daten (string), statt String ist auch ein Byte-Array möglich

SocketSendTo(addr As String, port As Long, data As String

Aufruf wenn Daten gesendet werden sollen.
Parameter

addr	die IP-Adresse des Senders (string)
port	die port-ID (Long)
data	Die zu sendenden Daten (string), statt String ist auch ein Byte-Array möglich

Anmerkung: Connect und Disconnect wird für UDP nicht aufgerufen! Die Socket-Verbindung kann vom Server zurückgesetzt werden durch Aufruf der Methode DestroySocketServer() gefolgt von einem erneuten CreateSocketServer(...)