MANAGER TOOLKIT ETHERNET/IP

KEY WORDS: EIP, CONTROL LOGIX, ALLEN BRADLEY, CIP, ADD-ON

EVOLUTIONS

Revision	Writer	Action	Date
1	CDT	Initial version	02/07/2016
2	ED	Rewrite after task modification	12/12/2016
2.1	ED	Add server redundancy support	27/03/2017

content

Scope of this document

The scope of this document is to describe and explain how to set-up the add-on svmgrEIP.dll.

Introduction

The add-on usrmgrEIP.dll enables to communicate with Ethernet IP via CIP Symbolic element**.**

The add-on DLL must present along with stack DLL EIP_Driver.dll

To get information regarding the svmgrEIP task while running the application, please check the event viewer for lines starting with "EtherNet/IP..."

Project structure

Add-on activation

In order to activate the add-on, it is necessary to create the file usrmgr.dat file. You can simply rename the svmgrEIP_UsrMgr.dat into usrmgr.dat

If the file is missing you can simply create a text file named usrmgr.dat with the following content

```
[USRMGR\usrmgrEIP]
DLL=svmgrEIP.dll
```

Configuring the task

The task is configured via a single file located in the project PER directory.

YourProjectName\PER

EtherNetIP.ini: Used to configure the add-on.

File description

EtherNetIP.ini

Please note that if an entry is omitted from the file or if the file is not present in the project then its default value will be applied by the task.

The file format is:

```
[TaskParameters]
AttributeNumber   = 16
LocalIpAddress    = 192.168.56.1
nbEquipments      = 1
nbPollingGroups   = 1

[PollingGroup1]
Name   = Fast
RefreshRateInSec = 1

[Equipment1]
Name          = EqtTest
IpAddress     = 192.168.56.11
Port          = 0
Slot          = 0
TimeOutInSec  = 10
```

Where:

[TaskParameters]

AttributeNumber: Specify the extended attribute used for configuring an EIP/ CIP variable.
Format: Number from 3 to 16. If omitted, **default is 16**

LocalIpAddress: Specify the IP Address of the Pc on which the task runs.
Format: String

AssociationName: Specify the association name in a redundant architecture. The task is only active on the active server.
Format: String. If omitted, **default is an empty string**

NbEquipments: Specify the number of equipment the task connect to.
Format: Integer. If omitted, **default is 0 equipment**

NbPollingGroups: Specify the number of polling group defined for the apllication.
Format: Integer. If omitted, **default is 0 polling group**

[PollingGroupX] Specify the definition of the polling group X.
Format: number from 1 to nbPollingGroups specified above. The value must be contiguous

> ***Name***: Specify a name of the polling group.
> *Format*: String. If omitted, **default is PollingGroupX as defined in [PollingGroupX]**
>
> ***RefreshRateInSec***: Specify the polling rate of the group in seconde.
> If entry omitted from the file the **default is Normal**

[EquipmentX] Specify the definition of the equipment X.
Format: number from 1 to nbEquipments specified above. The value must be contiguous

> ***Name***: Specify a name of the equipment.
> *Format*: String. If omitted, **default is EquipmentX as defined in [EquipmentX]**
>
> ***IPAddress***: Specify the IP Address of the equipment you want to reach.
> *Format*: String formatted x.x.x.x. If omitted **default is empty string**
>
> ***Port***: Specify the equipment port
> *Format:* Number. If omitted, **default is 0**
>
> ***Slot***: Specify the equipment slot
> *Format:* Number. If omitted, **default is 0**
>
> ***TimeOutInSec***: Specify the timeout in second on read or write request before considering the request has time-out.
> *Format:* Number. If omitted, **default is 10 s**
>
> ***DeviceType***: Specify the IP Address of the equipment you want to reach.
> *Format*: String formatted LogixAccess or CIPAccess. If omitted **default is LogixAccess string**
>
> ***MultipleServiceSupport***: Specify if the devise support the Service 0x0a meaning that several request can be concatainated within on.
> *Format*: String formatted Yes or No. If omitted **default is Yes string**

CompatcBitArray: Specify if the devise support the compact bit array. If yes the array are codede within DWORD value otherwise each bit is coded within one byte.
Format: String formatted Yes or No. If omitted **default is Yes string**

Variable definition

To define an EtherNet/IP variable in the supervisor you simply need to specify in the extended attribute (number specify in the EtherNetIP.ini file) of the variable:

EIP#EquipmentName#TagName#Type#PollingGroup#Access#Scaled

Where:

***EIP*:** Key to consider the variable to be an EIP variable

***EquipmentName*:** Equipment name from which the variable gets its value as defined in the EtherNetIP.ini

***Tag Name*:**
In case of Control Logix access:
Tag name refer to the tag name in the equipment.

If the tag is part of a program named MyProg:
syntax should be **Program:MyProg.MyTag**

If the tag is part of a UDT structure TemplateStructA called Mystruct which contains the element MyTag. To access MyTag of the instance,
syntax should be: **MyStruct.MyTag**
If MyStruct is instantiated with the program MyProg:
syntax should be **Program:MyProg**.M**yStruct.MyTag**

If the tag is an array:
syntax should be **MyArray[x]** where x is the array index of the specific element you want to access.

In case of CIP generic device:

Tag name refer to a specific explicit message within the CIP device. It has a specific syntax: **CIP:S:C:n:i:[x]** where

S: Service express in hexadecimal for instance

01: Get Attributes all

0e: Get Attribute single

C: Class in hexadecimal

I: Instance number in decimal

N: Attribute number

[x]: The offset in byte of the response buffer from which to extract the variable value.

***Type*:** Define the variable data type. Expect Following value:

Format	Scaling available	Data type Min Max
BOOL	No	
SINT	Yes	min = 0 max = 255
INT	Yes	min = 0 max = 65535
DINT	Yes	min = 0 max = 4294967295
USINT	Yes	min = 0 max = 255
UINT	Yes	min = 0 max = 65535
UDINT	Yes	min = 0 max = 4294967295
REAL	No	

***PollingGroup*:** Polling group name at which you want the value to be refreshed.

***Access*:** Enter **W** if you want the supervisor to write to the PLC the given tag.

***Scaled*:** Enter **S** if you want the supervisor to apply a scaling between the min max the data type and the min max defined in the PcVue variable. If omitted the variable will not be scaled.

Trace

Additional trace can be triggered using the SCADA BASIC verb

TRACE(1/0,11,FlagBit);

Where:

1/0	1: Turn ON the additional trace	
	0: Turn OFF the additional trace	
11	Svmgr order	11: 1st svmgr launched
		…
		18: 8th svmgr launched
Flagbit	hexadecimal combination of flag	
	001 To view cyclic errors	

Example

TRACE(1,11,"001"); 'Turn on all additional trace

TRACE(0,11,"001"); 'Turn off all traces.

Error code

Dongle protection

EtherNet/IP - Unable to read dongle, error 12

> No dongle plug in. The driver will run for 1 hour

EtherNet/IP – No license found on the protection key

> The Ethernet/Ip license is not coded in the dongle. The driver will run for 1 hour

Device connection

EtherNet/IP – DeviceName (ip Slot Port) error Value on request

> Value = 33: Time out on the request

> Value = 74: Response too long. The response is bigger than 511 bytes.

> Value = 58: TCP network error. Cannot connect to DeviceName please check connection using ping

TagName error

EtherNet/Ip - DeviceName ArrayTagName[index_min..Index_max] Error Value

EtherNet/Ip - DeviceName TagName Response Error Value [VarName]

> Value = 0x4: Invalid TagName. A syntax error was detected decoding the Request Path.

> Value = 0x5: Invalid TagName. Request Path destination unknown: Probably instance number is not present.

> Value = 0x6: Insufficient Packet Space: Not enough room in the response buffer for all the data.

> Value = 0x13: Insufficient Request Data: Data too short for expected parameters.

> Value = 0x26: The Request Path Size received was shorter or longer than expected.

> Value = 0xFF: General Error: Access beyond end of the object.