Modbus (Client) DSLink User Guide

Kinetic - Edge & Fog Processing Module (EFM)

Revised: April 2, 2020

Version 1.0.0.2

Contents

Introduction

This document describes the Modbus DSLink for the Cisco Kinetic Edge & Fog Processing Module. The ModbusDSLink allows for client connectivity to an external Modbus Slave, discovery the tags available and the ability to subscribe to specific tags for telemetry.

The Modbus DSLink supports more than one Modbus Slave to be connected simultaneously.

Networking

The Cisco Modbus DSLink will connect to its DSA message broker via tcp/http(s) communication. If the link was started from the DSA broker, the link will use a tcp connection to a local dynamic port of the DSA broker. If the link was started manually, the link has to use the public http or https DSA broker port.

Filesystem

The installed Cisco Modbus DSLink consists of:

Artifact	Type	Description
`bin`	Folder	Contains the start scripts
`lib`	Folder	Contains all depending libraries
`.key`	file	Contains the DSA broker token
`dslink.json`	file	Configuration for the initial link configuration and start
`nodes.json`	file	Configuration for the current link configuration and current metric information

Installation, Life cycle management and Logging

The Cisco Serial DSLink uses the standard EFM mechanisms for install, upgrade and logging. Please look into the Cisco Kinetic EFM System Administrator User Guide.

Node hierarchy

Each Modbus Slave that is defined has a unique name to identify the slave. For each connection, a top-level node with the given name will be created in the Cisco Modbus DSLink node hierarchy. In addition, the following sub-nodes are created:

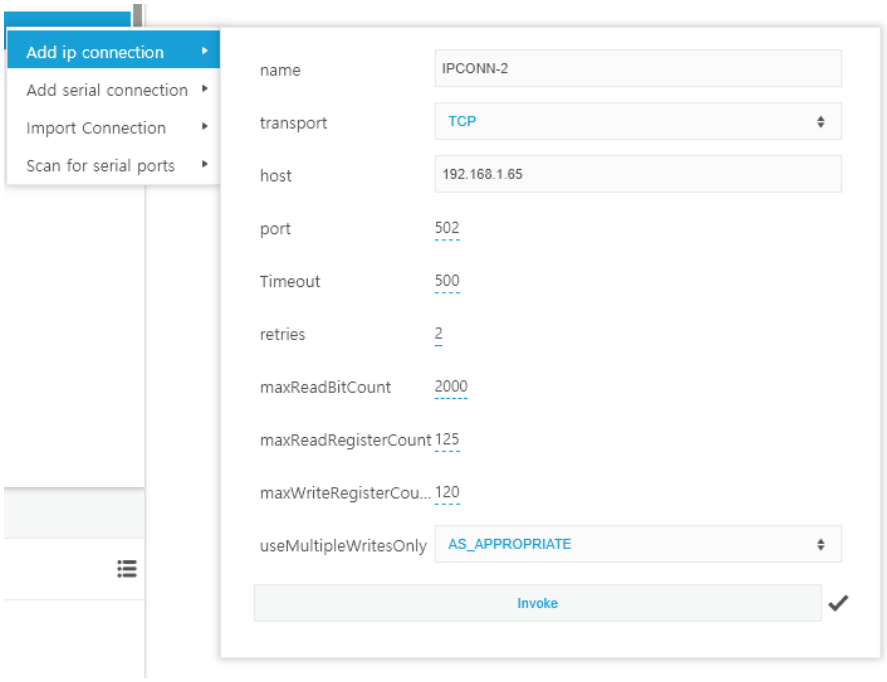
- Points - the list of points that are being read
- Folders - Folders are optional and are user created so to allow for placement of points below for organizational purposes

Link Actions and subactions

Add IP connection

Add Endpoint creates a new connection to an Modbus SlaveModbus Slave IP address or host name. Use the the Add ip connection action. To add a new connection to an Modbus IP Connection, the following information is required:

Parameter	Type	Default	Description
name	String		Name of the connection
transport	Selector	TCP	transport protocol to use ,TCP or UDP
host	String		IP address or host name of the Modbus endpoint
port	Integer	502	Connection port number, 502 is protocol standard
Timeout		500	timeout for Modbus requests, in milliseconds
Retries		2	Number of attempts to make to open the connection
Max read bit count		2000	maximum number of bits that can be read (from coils or discrete inputs) in one request
Max read register count		125	maximum number of (holding or input) registers that can be read in one request
Max write register count		120	maximum number of holding registers that can be written in one request
Use multipleWritesonly	Selector	As_Appropiate	**As_Appropiate, Always, Never** If target Modbus master supports multiple write commands, set this to true. If not certain, set it to be "As_Appropritae" and the DSLink would intelligently figure it out and work accordingly



If the connection is successful, a new node under the Modbus node will be added and the Connection status node will have a value of Connected. The next step is to use the connection's Add ip device action to connect to a device.

Add IP Device

Once the IP connection is established, an IP Device can be added. By selecting the ip connection node name, the action becomes available. To add a new ip device to an Modbus IP Connection, the following information is required:

Parameter	Type	Default	Description
name	String		Name for the device
Slave id	Integer	1	the device's ID number
polling interval (ms)	Integer		in milliseconds, how often the DSLink should poll the device for the values of points that you're subscribed to

zero on failed poll	Selector	false	whether the DSLink should display a value of 0 (or false for booleans) when it fails to get a point's value
use batch polling	Selector	true	timeout for Modbus requests, in milliseconds
contiguous batch requests only	Selector	false	affects how read requests are batched, if use batch polling is set to true. Some Modbus devices have non-contiguous sets of values within a single register range, potentially causing error responses to batch read requests Setting this parameter to true will ensure that this doesn't happen, by partitioning requests into only contiguous sets. This is generally not very efficient, so only set this to true if you are seeing errors when trying to read points

Add Folder

You can use the add folder action of the device node (or a folder node) to add a folder child node. This is purely for organizational purposes. Added points can be placed into the folder via the add point action. By selecting the folder additional folders and points may be added under the node.

Add Point

After adding the IP Device adding its points can be accomplished in the Dataflow editor. You can use the add point action of the device node or a folder node to add a point whose value you want to track. Consulting the device's register map is needed in order to determine the parameters of the device's points. Register maps vary significantly in format, so some trial and error may be necessary to figure it out.

Parameter	Type	Default	Description
name	String		Name for the data point
type	Integer	1	the type of Modbus entity (coil, discrete input, input register, or holding register) represented by this point
offset	Integer	0	the address of this point
Number of registers	Integer	1	the number of registers that this point takes up only applies to string data types (CHARSTRING and VARCHARSTRING) for non-string data types, number of registers is inferred from the type

datatype	Selector	Boolean	data type - the data type of this point (See Data Types section below for more information).
bit	Integer	null	which bit of the register contains the value only applies to packed booleans (BOOLEAN data type and either INPUT or HOLDING type), where 16 boolean values are stored in a single 16-bit register
scaling	Integer	1	Allow for scaling the number down by the factor defined. For example, if the scaling value is 1000 then the output value will be divided by 1000. Scaling/Scaling offset is applied only to those values that could be represented with Int64 range, i.e. int64 (-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807) Once Scaling/Scaling offset is applied, up to 2 decimal points are displayed
Scaling offset	Integer	0	number to add to the value before displaying it Once Scaling/Scaling offset is applied, up to 2 decimal points are displayed
writeable	Selector	false	whether or not the point is writable controls whether or not the DSLink will allow you to try to write to this point

For more information on point mapping, consult the device user manual.

Import

Use Import action to read in the exported IP Device node configurations and properties.

Note that to import the exported info, one must select the parent node for the action. The import must correspond to the node type that was exported. For example, nodes exported from an IP Connection node could be imported by selecting the DSLink instance node and for nodes exported from a Device node, they can be imported on a specific IP Connection node.

Export

Export displays a JSON table in the textbox with all the configuration and configuration info of the selected node and its subnode(s) of an IP Device.

Clone

This action allows cloning a selected IP Device with the new name provided and added to the node tree. Once the point is cloned, it is possible to select the newly cloned node and use the Edit Point to change any specific attribute. This is designed to provide convenience to support various use cases, i.e. point read and write needs to be grouped in separate nodes, multiple devices are deployed with almost identical points to for example installed same device to expand production lines.

Stop

Polling of metrics stops.

Start

Polling of metrics is started. If the polling is successful, the last successful polled value is updated.

Remove

This action removes an IP Device point.

@addAtribute

This action allows for adding an attribute to the IP Device.

Point node Actions

Edit Point

This action allows for modification of existing attributes that define a point. By selecting an existing point node, the Edit Point action allows for modifying any of the attributes that define the point. This is used to correct errors or modify cloned points.

Clone

This action allows cloning a selected Point with the new name provided. Once the point is cloned, it is possible to select the newly cloned node and use the Edit Point to change any specific attribute. This is designed to provide convenience to support various use cases, i.e. point read and write needs to be grouped in separate nodes, multiple devices are deployed with almost identical points to for example installed same device to expand production lines.

Remove

Write Point (value)
If the Point attribute writable is true, then the @set action is allowed. This action writes a value to the Modbus point.

@addAtribute

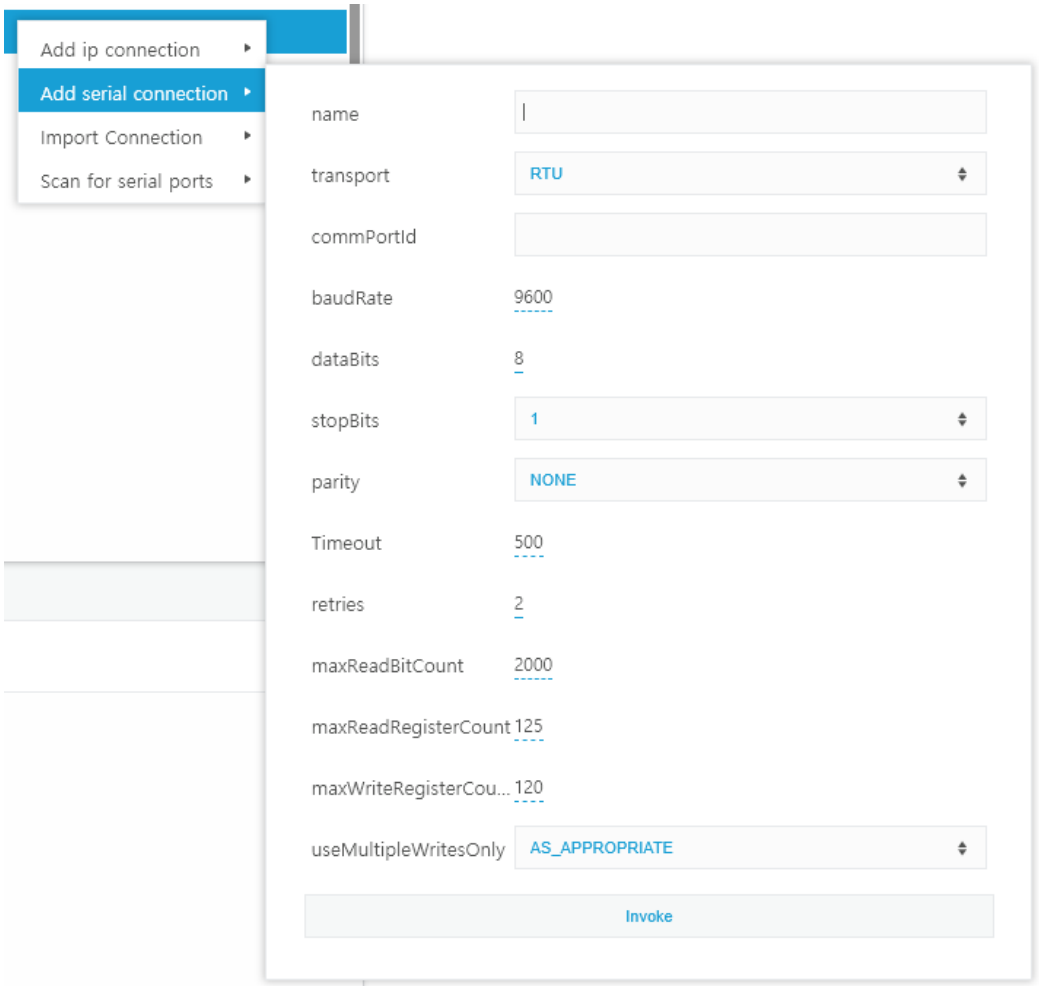
This action allows for adding an attribute to the Point.

Add serial connection

Add Serial Connection creates a new connection to an Modbus Slave using the local hosts serial interface.

Add a new connection to an Modbus serial connection, the following information will be required:

Parameter	Type	Default	Description
name	String		Name of the connection
transport	Selector	RTU	Modbus transmission mode RTU or ASCII
Comm port ID	String		The DSLink should automatically detect any available serial ports, allowing you to choose one from a drop-down menu If you don't see your serial port in the drop-down, try invoking the **scan for serial ports** action If no serial ports are found, you can enter the name of your serial port manually here
baudrate	Integer	9600	The baud rate of the serial connection
Data bits	Integer	8	the data bits of the serial connection
Stop bits	Selector	1	the stop bits of the serial connection. Stopbits that can be either 0 or 1
parity	Selector	None	The parity of the serial connection. Parity that can be selected from None, Odd, Even
Timeout	Integer	500	The timeout for Modbus requests, in milliseconds
Retries	Integer	2	Number of attempts to open the connection
Max read bit count		2000	maximum number of bits that can be read (from coils or discrete inputs) in one request
Max read register count		125	maximum number of (holding or input) registers that can be read in one request
Max write register count		120	maximum number of holding registers that can be written in one request
Use multiple writes only	Selector	As_Appropiate	**As_Appropiate, Always, Never** If target Modbus master supports multiple write commands, set this to true. If not certain, set it to be "As_Appropritae" and the DSLink would intelligently figure it out and work accordingly



Add serial device

Once the serial connection is established, a serial device can be added. By selecting the ip device node name, the action becomes available. To add a new serial device to an Modbus Serial Connection, the following information is required:

Parameter	Type	Default	Description
name	String		Name for the device

Slave id	Integer	1	the device's ID number
polling interval (ms)	Integer		in milliseconds, how often the DSLink should poll the device for the values of points that you're subscribed to
zero on failed poll	Selector	false	whether the DSLink should display a value of 0 (or false for booleans) when it fails to get a point's value
use batch polling	Selector	true	timeout for Modbus requests, in milliseconds
contiguous batch requests only	Selector	false	affects how read requests are batched, if use batch polling is set to true. Some Modbus devices have non-contiguous sets of values within a single register range, potentially causing error responses to batch read requests Setting this parameter to true will ensure that this doesn't happen, by partitioning requests into only contiguous sets. This is generally not very efficient, so only set this to true if you are seeing errors when trying to read points

Add Folder

You can use the add folder action of the device node (or a folder node) to add a folder child node. This is purely for organizational purposes. Added points can be placed into the folder via the add point action. By selecting the folder additional folders and points may be added under the node.

Add Point

After adding the IP Device adding its points can be accomplished in the Dataflow editor. You can use the add point action of the device node or a folder node to add a point whose value you want to track. Consulting the device's register map is needed in order to determine the parameters of the device's points. Register maps vary significantly in format, so some trial and error may be necessary to figure it out.

Parameter	Type	Default	Description
name	String		Name for the data point
type	Integer	1	the type of Modbus entity (coil, discrete input, input register, or holding register) represented by this point
offset	Integer	0	the address of this point

Number of registers	Integer	1	the number of registers that this point takes up only applies to string data types (CHARSTRING and VARCHARSTRING) for non-string data types, number of registers is inferred from the type
datatype	Selector	Boolean	data type - the data type of this point (See Data Types section below for more information). The options are BOOLEAN, INPUT and HOLDING type.
bit	Integer	null	which bit of the register contains the value only applies to packed booleans (BOOLEAN data type and either INPUT or HOLDING type), where 16 boolean values are stored in a single 16-bit register
scaling	Integer	1	Allow for scaling the number down by the factor defined. For example, if the scaling value is 1000 then the output value will be divided by 1000. Scaling/Scaling offset is applied only to those values that could be represented with Int64 range, i.e. int64 (-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807) Once Scaling/Scaling offset is applied, up to 2 decimal points are displayed
Scaling offset	Integer	0	number to add to the value before displaying it Once Scaling/Scaling offset is applied, up to 2 decimal points are displayed
writeable	Selector	false	whether or not the point is writable controls whether or not the DSLink will allow you to try to write to this point

For more information on point mapping, consult the device user manual.

Import

Use Import action to read in the exported IP Device node configurations and properties.

Note that to import the exported info, one must select the parent node for the action. The import must correspond to the node type that was exported. For example, nodes exported from an IP Connection node could be imported by selecting the DSLink instance node and for nodes exported from a Device node, they can be imported on a specific IP Connection node.

Export

Export displays a JSON table in the textbox with all the configuration and configuration info of the selected node and its subnode(s) of an IP Device.

Clone

This action allows cloning a selected IP Device with the new name provided and added to the node tree. This is designed to provide convenience to support various use cases, i.e. point read and write needs to be grouped in separate nodes, multiple devices are deployed with almost identical points to for example installed same device to expand production lines.

Stop

Polling of metrics stops.

Start

Polling of metrics is started. If the polling is successful, the last successful polled value is updated.

Remove

This action removes an IP Device point.

@addAtribute

This action allows for adding an attribute to the serial Device.

Point node Actions

Edit Point

This action allows for modification of existing attributes that define a point. By selecting an existing point node, the Edit Point action allows for modifying any of the attributes that define the point. This is used to correct errors or modify cloned points.

Clone

This action allows cloning a selected Point with the new name provided. Once the point is cloned, it is possible to select the newly cloned node and use the Edit Point to change any specific attribute. This is designed to provide convenience to support various use cases, i.e. point read and write needs to be grouped in separate nodes, multiple devices are deployed with almost identical points to for example installed same device to expand production lines.

Remove

Write Point (value)
If the Point attribute writable is true, then the @set action is allowed. This action writes a value to the Modbus point.

@addAtribute

This action allows for adding an attribute to the Point.

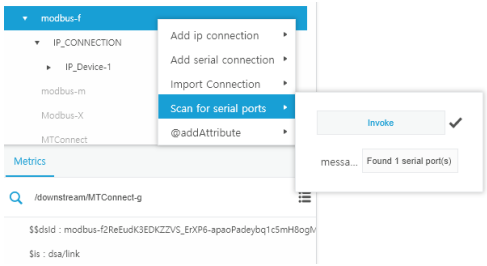
Import Connection

Use Import action to read in the exported IP Connection node configurations and properties.

Note that to import the exported info, one must select the parent node for the action. The import must correspond to the node type that was exported. For example, nodes exported from an IP Connection node could be imported by selecting the DSLink instance node and for nodes exported from a Device node, they can be imported on a specific IP Connection node.

Scan for Serial Ports

Serial ports can be scanned on the host for verification. The message indicates the number of serial ports found and if serial ports are found then the list is stored. This list is used when adding a serial connection action.



Modbus Point Addressing

The manner in which register maps display the “address” of a Modbus entity can vary. There is a distinction between an entity’s “number” and its “address”, and many register maps display the “number”. The number uses the first digit to indicate the type of entity, and the rest of the digits to indicate the specific address (or “offset”). In the traditional standard, the numbering is as follows:
In the traditional standard, numbers for those entities start with a digit, followed by a number of 4 digits in the range 1–9,999:

- coils numbers start with 0 and span from 00001 to 09999,
- discrete input numbers start with 1 and span from 10001 to 19999,
- input register numbers start with 3 and span from 30001 to 39999,
- holding register numbers start with 4 and span from 40001 to 49999.

The address (offset) of a point is the offset of its number from the lowest number of its type. For example, if a point has number 41000, then the type is HOLDING and the offset is 99 (HOLDING starts at 40001).

This allows the address to be we anywhere between 0 and 9998. Since the offset field of a Modbus request frame is 16 bits long, some devices forego the traditional numbering in fravor of an extended referencing, in which each number has 6 digits by adding an additional digit. The numbers are as follow:

- coil numbers span from 000001 to 065536,
- discrete input numbers span from 100001 to 165536,
- input register numbers span from 300001 to 365536,
- holding register numbers span from 400001 to 465536.

Modbus Point Data Types

Information about the available data types that are implemented when defined a device point.

- Modbus registers are 16 bit, 16-bit values take up one register, 32-bit values take up two, and so on.
- INTX types represent signed X-bit integers
- UINTX types represent unsigned X-bit integers
- FLOATX types represent X-bit floating-point numbers
- BCDX type represent X-bit binary-coded decimal numbers
- INT32M10K and UINT32M10K types represent “modulo 10000” 32-bit integers, where the more significant register contains value/10000 and the less significant register contains value%10000
- There is some ambiguity in numeric types as to which bytes are more or less significant. Normally, the first register is most significant, and within each register, the first byte is most significant. When this is not the case, you should use on of the data types that end in SWAP.
    - The ordering of bytes from most to least significant is as follows (rX[Y] represents the Yth byte of the Xth register):
        - INT16, UINT16 and BCD16: r0[0], r0[1]
        - INT16SWAP and UINT16SWAP: r0[1], r0[0]
        - INT32, UINT32, FLOAT32, BCD32 and INT32M10K: r0[0], r0[1], r1[0], r1[1]
        - INT32SWAP, UINT32SWAP, FLOAT32SWAP, BCD32SWAP and INT32M10KSWAP: r1[0], r1[1], r0[0], r0[1]
        - INT32SWAPSWAP and UINT32SWAPSWAP: r1[1], r1[0], r0[1], r0[0]
        - INT64, UINT64 and FLOAT64: r0[0], r0[1], r1[0], r1[1], r2[0], r2[1], r3[0], r3[1]
        - INT64SWAP, UINT64SWAP and FLOAT64SWAP: r3[0], r3[1], r2[0], r2[1], r1[0], r1[1], r0[0], r0[1]

Persisting Modbus Configuration parameters

Modbus Client DSLink automatically persist configurations information in the nodes.json file.

References for securing the EFM operating platform

The EFM Installation guides describe the product configuration options, but the administrators may desire hardening the operating platform that the EFM and its components function.

RedHat: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/pdf/security_guide/Red_Hat_Enterprise_Linux-7-Security_Guide-en-US.pdf

Microsoft Windows: Server Hardening: Windows Server 2012 https://technet.microsoft.com/en-us/security/jj720323.aspx

Cisco Guide to Hardening Cisco IOS devices - https://www.cisco.com/c/en/us/support/docs/ip/access-lists/13608-21.html

Obtaining documentation and submitting a service request

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.