CIM10 Documentation

Introduction:

The CIM 10 represents a cutting-edge industrial product that epitomizes technological sophistication. At its core lies a formidable high-power industrial 32-bit CPU, designed to meet the demands of the most intricate industrial processes. This device boasts a harmonious integration of multi-layered software support and hardware mechanisms, forming a robust foundation that facilitates seamless connectivity with a myriad of peripheral devices.

Overview:

The CIM 10 is engineered to excel in industrial environments, where reliability, efficiency, and adaptability are paramount. Its prowess is evident in its compatibility with multiple worldwide carrier 2G/4G cellular networks, specifically catering to LTE-CAT1 standards. This not only ensures global connectivity but also establishes the CIM 10 as a versatile solution for diverse industrial applications.

One of the standout features of the CIM 10 is its extensive protocol support. It embraces a wide range of industrial communication protocols, including OPC UA/DA, Modbus RTU, Modbus TCP/IP, BACnet, and BLE (Bluetooth Low Energy). This versatility enables the CIM 10 to seamlessly integrate with various devices and systems, fostering interoperability in industrial settings.

The CIM 10 is equipped with a rich array of interfaces, further enhancing its adaptability to different industrial scenarios. The inclusion of the X1 LAN port, X2 RS485 port, X1 USB Device port, digital input ports (X2), and analog input ports (X2) showcases the device's commitment to providing comprehensive connectivity options. These interfaces serve as gateways for data exchange, allowing the CIM 10 to interact with a diverse range of equipment and sensors.

The convergence of Operational Technology (OT) and Information Technology (IT) in industrial settings has become increasingly crucial for driving digital transformation. However, bridging the gap between these traditionally distinct domains poses challenges, and this is where CIMCON steps in. CIMCON offers a comprehensive platform, leveraging cloud-based solutions, support for multiple protocols, and the CIM 10 device for peripheral connections. This innovative approach addresses the shortcomings of OT/IT convergence and plays a pivotal role in streamlining data processing, enhancing operational efficiency, and ultimately improving safety in industrial operations.

Key Features and Solutions:

Cloud Platform Integration:

CIMCON's platform revolves around cloud integration, providing a centralized hub for data management and analytics. By aggregating data from multiple devices onto the cloud, CIMCON facilitates real-time monitoring, analysis, and decision-making. This not only breaks down silos between OT and IT systems but also ensures that data becomes a valuable asset for driving actionable insights.

Support for Multiple Protocols:

CIMCON recognizes the diverse landscape of industrial communication protocols. With support for protocols such as OPC UA/DA, Modbus RTU, Modbus TCP/IP, BACnet, and BLE, CIMCON ensures compatibility with a wide range of devices and systems. This interoperability minimizes integration challenges, allowing for a seamless flow of information between different components of an industrial setup.

CIM 10 Peripheral Connection:

The CIM 10 device serves as a linchpin in CIMCON's strategy. Designed with a high-power industrial 32-bit CPU, it acts as a versatile bridge, connecting various peripheral devices. Its support for worldwide 2G/4G cellular networks (LTE-CAT1) enhances connectivity options, enabling data exchange over multiple mediums. The inclusion of diverse interfaces, such as LAN, RS485, USB, digital input, and analog input ports, ensures that the CIM 10 can adapt to various industrial scenarios, promoting seamless data flow.

Computing and Processing Power:

CIMCON recognizes the importance of efficient computing and processing capabilities at the edge. By embedding powerful hardware mechanisms within the CIM 10, the platform ensures that data is not only collected but also processed locally. This decentralized approach minimizes latency, improves response times, and enhances the overall efficiency of industrial operations.

Improving Operational Efficiency and Safety:

The holistic approach of CIMCON addresses the core issues hindering OT/IT convergence. By transforming raw data into actionable insights, industries can optimize their operational processes. This not only enhances efficiency but also contributes to improved safety standards. Real-time monitoring and predictive analytics enable proactive decision-making, preventing potential risks and minimizing downtime.

End-to-End IIoT Solution:

CIM 10 offers a comprehensive end-to-end Industrial Internet of Things (IIoT) solution, addressing remote service, monitoring, and maintenance needs. From hardware to the Cloud, it is engineered to facilitate the creation of IoT solutions for a wide range of industrial use cases. Its versatility enables support for various protocols, making it suitable for applications such as monitoring, asset tracking, equipment condition monitoring, and predictive maintenance.

Connectivity Options:

CIM 10 boasts multiple wired and wireless connectivity options, ensuring seamless integration with a diverse array of sensors and devices. Equipped with analog and digital inputs, RS485, USB-Host, and Bluetooth, it enables data ingestion from both local and remote field devices. Furthermore, it supports RS232, Gigabit Ethernet, Wi-Fi, and 2G/4G cellular networks for host server/cloud connectivity, incorporating industry-standard protocols like MODBUS and MQTT.

High-Speed IO Expansion:

Designed to cater to applications with a substantial IO count, CIM 10 features a dedicated high-speed RS-485 expansion bus. This ensures efficient handling of large amounts of input and output data, facilitating scalability and adaptability to varying industrial requirements.

Edge Computing Capabilities:

Powered by a robust 1GHz single-core processor, CIM 10 excels in edge computing. This processing power empowers the execution of compute-intensive AI/ML algorithms, enabling real-time data processing for immediate alerts and notifications. By performing computations at the edge, CIM 10 reduces latency and minimizes the overhead associated with transmitting data to the cloud.

CIM CLOUD:

CIM 10 is seamlessly integrated with CIM Cloud, providing a centralized management solution for devices at scale and enterprise-wide. This ensures secure device management and facilitates efficient deployment and monitoring of the entire CIM 10 ecosystem.

Data Lake (iData):

CIMCON Digital introduces the iData platform, emphasizing the importance of data in asset management. The iData Data Lake securely stores structured, semi-structured, or fully unstructured data. Authorized personnel, including developers, data scientists, and analysts, can leverage this data for reporting, visualization, advanced analytics, and machine learning, contributing to informed decision-making.

Ease of Configuration:

CIM 10 prioritizes user-friendly configuration with multiple interface options. Users can leverage USB debug and Bluetooth to connect to smart devices, providing flexibility for on-site analysis and configuration during runtime data capture.

Security:

Cybersecurity is a paramount concern, and CIM 10 addresses this with secured provisioning, communication, and data integrity. The enhanced cybersecurity architecture operates at both IT and OT levels, ensuring the confidentiality, integrity, and availability of data throughout the IIoT ecosystem.

Device Technical Specifications:

Core:

ARM Cortex-A8

Single core, 1GHz

Memory:

RAM: 512 MB

Flash: 4 GB

microSD card slot: x1, for additional data storage

RTC (Real-Time Clock) with battery backup: Yes

Peripheral IO:

Digital Input: x2, 12/24V DC operated

Analog Input: x2, 12-bit resolution, 0-10V / 4-20mA input

IO Expansion: Yes, Modbus IO (External on RS-485)

Wired Connectivity:

RS-485 port: x1

Options:

Ethernet port: x1, 10/100 Mbps

USB Device: x1 (Console)

Wireless Connectivity:

Cellular modem interface: 2G/4G LTE-CAT1

Options:

GPS (Optional): GPS, GLONASS, BeiDou

Wi-Fi: Not specified

Bluetooth: x1 (BT5.0 protocol)

Power Supply:

DC operated: 12~24V DC, 15W max

Enclosure:

Plastic: ABS/PC

Mounting: Plate mount (with additional clamp) or DIN rail

Operating Conditions:

Temperature: -20°C to +55°C

Humidity*: 20%-90% RH, non-condensing

IP*: IP20 (*Panel installation)

These technical specifications outline the robust architecture and capabilities of the CIM 10 device. With its ARM Cortex-A8 core, ample memory, and versatile IO options, the CIM 10 is designed to meet the demands of industrial applications. The combination of wired and wireless connectivity, along with options for expansion and additional features like GPS and Bluetooth, positions the CIM 10 as a flexible and powerful solution for diverse industrial IoT scenario

Application Areas of CIM10 in Industrial IoT:

1. Smart Water:

Water Generation, Treatment, and Filtration:

CIM10 plays a crucial role in water treatment processes, aiding in the efficient generation, treatment, and filtration of water in plants.

Leak Detection and Safety:

The device excels in water leak detection and ensures safety through the monitoring of chlorine leaks and the detection of heat or smoke in starter panels.

Preventive Maintenance for Motors:

CIM10 facilitates predictive maintenance for pump and tube well motors, utilizing data analytics to detect issues such as excessive motor vibration, pump dry running, and locked rotor conditions by monitoring various parameters like voltage, current, power, flow, vibration, and temperature.

2. Smart Building:

Energy Efficiency and Optimization:

CIM10 contributes to energy efficiency analysis and optimization opportunities in smart buildings.

Access Control and Safety:

The device enhances security through access control features and aids in maintaining a safe environment.

Space Utilization and Parking Management:

CIM10 supports space utilization and efficient parking management in smart buildings.

3. Smart Manufacturing:

Machine Efficiency and Power Consumption:

In manufacturing environments, CIM10 provides analysis tools for machine efficiency and power consumption, contributing to overall operational optimization.

Fault Detection and Diagnosis:

The platform excels in detecting and diagnosing machine faults, ensuring minimal downtime and improved productivity.

Quality Analytics:

CIM10 supports quality analytics, helping manufacturers maintain and enhance product quality.

4. Oil & Gas Distribution Network:

Cathodic Protection Monitoring:

CIM10 offers real-time monitoring of critical parameters in Cathodic Protection applications, including TR current, TR voltage, and PSP (Pipe-to-Soil Potential).

Historical Data Logging and Trend Generation:

The device time-stamps data with geographical location information, enabling historical data logging and trend generation for comprehensive analysis.

5. Gas Compressor and Dispenser Monitoring:

Temperature and Pressure Monitoring:

CIM10 monitors temperature and pressure parameters in gas stations, generating alarms in case of anomalies.

Compressor Loading and Suction Pressure Control:

The platform supports remote control of gas compressors, ensuring efficient loading and monitoring suction pressure.

Quick Guide Manual: Dimensional Details of CIM 10

What's in the Box:

The CIM 10 package includes the following components:

CIM 10 device

GPS antenna

GSM antenna

SMPS (Switched-Mode Power Supply) - Optional

12/24V DC power supply for main power

Interface Layout:

The CIM 10 device has a well-organized interface layout to facilitate easy installation and connectivity.

Installation Considerations:

Heat Considerations:

Place CIM 10 devices away from heat-generating equipment and in cooler environments to ensure optimal performance.

Enclosure Installation:

These devices are designed to be installed in an enclosure or enclosed environment, accessible with the use of tools. They are suitable for various environments.

Proper Knowledge:

Do not use the device without proper training or adequate knowledge about CIM 10 and its capabilities.

Prohibition on Opening:

It is strictly prohibited to open the case of the CIM 10 device. Any damages resulting from such actions are considered "improper use" and are not covered by the warranty.

Power Supply:

The CIM 10 device operates on DC power, and the details are as follows:

Power Supply: 12V, 50W

This power supply is used to provide power to the CIM 10 device.

Connections:

Power Connection: 12/24V DC

Analog Input: X2

Ethernet Connection

Digital Input: X2

RS485 Connection: X1

GND Connection: X2

SIM Card Connection: CIM 10 supports GPRS connection using a SIM card for network connectivity.

Antenna (GPS and GSM):

External Antenna Support: The CIM 10 device supports connecting external GPS and GSM antennas for improved signal reception.

LED Status Indicators:

The CIM 10 device is equipped with multiple LED status indicators, providing users with information about the current status of the device. Below is a summary of the LED status indicators:

[Status Indicator 1]: [Description]

[Status Indicator 2]: [Description]

[Status Indicator 3]: [Description]

...

What is Intelligent EDGE?

Intelligent edge refers to the analysis of data and the development of solutions at the site where the data is generated. This approach reduces latency, costs, and security risks by processing data locally, making businesses more efficient. The three primary categories of intelligent edge are operational technology edges, IoT edges, and information technology edges, with IoT edges currently being the most extensive and popular.

Why Intelligent iEDGE?

Intelligent iEDGE technology maximizes business efficiency by performing data analysis at the location where the data is generated. Instead of sending data to a central data center or third-party service, intelligent iEDGE minimizes latency, enhances security, and reduces operational costs.

Application Interface:

WebUI:

The Application Interface WebUI serves as the user-friendly portal for configuring, monitoring application parameters, and conducting device health monitoring. It provides a centralized platform for users to interact with the CIM 10 device.

Login:

To access the WebUI, open the provided URL by CIMCON, typically set as 192.168.1.100. The default configuration is as follows:

IP: 192.168.1.100

Subnet: 255.255.255.0

Gateway: 192.168.1.1

Login with the provided credentials:

Username: [Enter Valid Username]

Password: [Enter Valid Password]

Upon successful login, users gain access to the system status page, offering insights into the current state of the CIM 10 device and its connected components.

(Image for illustration purposes only)

Configuration and Management:

System Status Page:

The system status page provides an overview of critical information, including device health, connectivity, and operational status. Users can quickly assess the CIM 10's performance and connectivity.

Configuration Options:

Within the WebUI, users can configure various settings, such as network parameters, data acquisition parameters, and security settings. This intuitive interface simplifies the process of tailoring the CIM 10 device to specific industrial requirements.

Monitoring Application Parameters:

The WebUI allows real-time monitoring of application parameters, ensuring users stay informed about the device's performance and the status of connected industrial equipment.

Device Health Monitoring:

Intelligent iEDGE includes robust device health monitoring features. The WebUI provides insights into the health of the CIM 10 device, enabling proactive maintenance and troubleshooting.

Device Status: Overview of CIM 10 System Status

The system status page provides a comprehensive overview of the current state of the CIM 10 device, offering real-time information critical for monitoring and managing its performance. This detailed status information is presented in five subparts, each shedding light on different aspects of the device's functionality.

a) CPU Consumption:

The CPU Consumption section reveals the total utilization of the device's CPU. This metric represents the ongoing CPU usage, monitored by the Edge monitoring services. Users can observe the real-time CPU consumption and make informed decisions based on the system's processing load. The values are updated dynamically, providing an accurate snapshot of the CPU's workload upon refreshing the page.

b) Memory Consumption:

The Memory Consumption section illustrates the total memory utilization of the CIM 10 device. It signifies the amount of memory being used by various CIM 10 services during execution, monitored by the Edge monitoring services. Like CPU consumption, this metric is updated in real-time, allowing users to assess the current memory demands and optimize device performance accordingly.

c) Network Interface:

In the Network Interface section, data usage is displayed for the CIM 10 device's connectivity modes, such as Eth0 (Ethernet). This segment exhibits the data consumption of the CIM 10 device connected to the network via Ethernet (eth0). Monitored by the Edge monitoring services, these metrics offer insights into the device's network activity, allowing users to gauge its connectivity performance. As with other sections, the values are dynamically updated upon refreshing the page.

d) Volume Backup:

Volume Backup showcases the current memory status, detailing used and free memory within the overall total capacity. Monitored by Edge monitoring services, this section provides a snapshot of the device's memory utilization, aiding in assessing its memory health. Real-time updates ensure users have accurate information at their disposal, assisting in proactive memory management.

e) Miscellaneous Section:

The Miscellaneous section encompasses crucial device details. It includes:

Device Uptime: The time elapsed since the last reboot, indicating the device's continuous operational duration.

Network Transport: The network communication protocol employed by CIM 10 to transmit data to CIMCON iCLOUD.

Active Network Interface: Displays the active network through which the CIM 10 device is currently connected.

Latitude & Longitude: Provides the GPS location of the CIM 10 device, enhancing geographical tracking capabilities.

Network Configuration and Management for CIM 10:

Network Interface Configuration Page:

The Network Interface Configuration page provides multiple options for connecting the CIM 10 device to a network through Ethernet and GPRS interfaces. This comprehensive support ensures flexibility and adaptability to various networking scenarios.

Ethernet (Eth0):

Using an Ethernet cable for network connectivity, CIM 10 can seamlessly send data from different devices to "CIMCON iCLOUD."

The default configuration of Eth0 service is displayed, showing the current status of the interface.

Users can choose between DHCP or Static IP configurations based on their requirements.

Connection is established by plugging in an Ethernet cable to the Eth0 port.

Ethernet (Eth0) Configuration Parameters:

Mode Selection:

a. Static IP address

b. DHCP Server

IP Address

Gateway Server

Subnet

Primary DNS Server (Optional)

Secondary DNS Server (Optional)

Users can configure and edit static IP parameters through the WebUI, providing flexibility in network setup.

GSM/LTE:

Utilizing a modem, CIM 10 connects to the network and sends data from various devices to "CIMCON iCLOUD."

Upon login, users access default configuration information.

GSM/LTE Configuration Parameters:

Serial Port

APN Name: Network Service Provider

Modem Name

Marked 4 Area Parameters: Displayed information once the modem is connected to the CIM 10 device.

Port Forwarding:

The WebUI's port forwarding feature enables users to access the device through different interfaces and network ports, using the CIM 10 device's assigned IP. This versatile feature supports diverse network configurations and ports, providing seamless access.

Port Forwarding Configuration Parameters:

Rule No.: Integer in increment order, reflecting the rule number added using the "+" sign.

Input Interface: Valid network interface name from the dropdown box (eth0, eth1, ppp0, wlan0).

Input Port: Valid port number for accessing the output interface asset.

Forward Interface: Valid network interface name from the dropdown box (eth0, eth1, ppp0, wlan0).

Forward IP: Valid IP address for the attached device to the output network interface.

Forward Port: Valid port number to be forwarded to the input interface.

Firewall Settings:

The WebUI's firewall setting feature empowers users to add multiple rules for whitelisting IPs and Ports using inbound and outbound rule types for TCP/UDP protocols, enhancing security and access control.

Firewall Configuration Parameters:

Rule No.: Integer in increment order, indicating the rule number added using the "+" sign.

Rule Type: Inbound or Outbound, determining whether the rule is for incoming or outgoing firewall.

Protocol Type: Selection of TCP or UDP protocol.

Whitelist IP: Ranges from "0.0.0.0" (enables the firewall for all IPs) to a specific IP for targeted access.

Whitelist Port: Ranges from 0 (enables the firewall for all ports) to a specific port for targeted access.

ChatGPT

Peripherals Integration with CIM 10 Device:

Digital Input:

CIM 10 supports up to 2 Digital Input (DI) pins, providing users with versatile options for monitoring various aspects such as power status, elevator status, counters, and more.

Configuration Parameters for Digital Input:

Pin Number:

Input Requirement: 1 to 2

Description: Specifies the DI pin number that needs to be read. Users can manually add DI pins up to 2.

Pin Name:

Input Requirement: String (32 characters)

Description: Sets the name for the Digital Input, enhancing user understanding.

Peripheral ID:

Input Requirement: Valid number

Description: Defines the valid peripheral ID, corresponding to the Client ID mentioned in the CIMCON iCLOUD portal.

Status:

Input Requirement: Enable/Disable

Description: Indicates the peripheral device's enable/disable status.

Remove:

Input Requirement: N/A

Description: Allows users to remove the device.

Sampling Rate:

Input Requirement: 0 to 3600 seconds

Description: Configures the duration for obtaining updated data, allowing users to set the sample rate.

Destination:

Input Requirement: Specifies the service where data should be sent, defined in the integration section.

Digital Input Specification:

Channel:

DI-1 (Input)

DI-2 (Input)

Scan Rate: 1000ms

Analog Input:

CIM 10 features 2 input pins for analog input, offering flexibility for connecting devices such as flow meters and discharge pressure transmitters.

Configuration Parameters for Analog Input:

Sampling Rate:

Input Requirement: 1 to 86400 seconds

Description: Determines the frequency for publishing data, defining the destination at the sampling rate.

Destination:

Input Requirement: Valid service name (multiple services)

Description: Specifies the services where analog data will be published, allowing a list of services to be defined.

Pin Number:

Input Requirement: 1 to 2

Description: Sets the Analog Input (AI) pin number that needs to be read. Users can add pins up to 2.

Name:

Input Requirement: String (32 characters)

Description: Configures the name for the analog channel.

Peripheral ID:

Input Requirement: Valid number

Description: Defines the valid peripheral ID, corresponding to the Client ID in the CIMCON iCLOUD portal.

Channel Type:

Input Requirement: Voltage (V) or Current (mA)

Description: Users can select based on the device's output, choosing between voltage or current.

Engg Scale Low/High:

Input Requirement: Voltage: -10 to +10, Current: 4 to 20 mA

Description: Converts raw values to engineering values within the specified range.

Scale Low/High:

Input Requirement: 0 to 100

Description: Converts engineering values to scale values based on device parameters.

Value:

Input Requirement: Real-time value of the device

Description: Displays the real-time value of the connected device.

Analog Input Specification:

Channel:

AI-1 (4 to 20 mA, -10V to 10V)

AI-2 (4 to 20 mA, -10V to 10V)

Scan Rate: 100ms

Modbus RTU Integration with CIM 10 Device:

Overview:

Modbus is a widely used open communication protocol in industrial manufacturing environments, facilitating master-slave or client-server communication between intelligent devices. CIM 10 devices are pre-configured to seamlessly work with CIMCON iCLOUD, allowing reliability managers to easily install and monitor equipment efficiently.

Modbus Register Map:

CIM 10 devices come pre-configured for use with CIMCON iCLOUD, ensuring quick and easy installation through a mobile application. The block diagram illustrates the communication flow between Modbus devices and the CIMCON iCLOUD platform.

Communication between Modbus Devices:

Modbus devices communicate using a master-slave technique, where the master initiates transactions, and slaves respond by supplying requested data. The master can address individual slaves or send broadcast messages to all slaves.

Basic Modbus Network:

The Modbus serial protocol is a master/slave protocol, with one master controlling data transactions and multiple slaves responding to read or write requests. Modbus is commonly used to control machines in factories, such as PLCs, ICS, DCS, and VFDs.

Applications of Modbus:

Used for master-slave/client-server communication between intelligent devices.

Commonly employed in industrial systems like PLCs, ICS, DCS, and VFDs.

Enables communication between approximately 247 devices on the same network.

Ideal for monitoring and programming devices, connecting sensors and instruments, and field device monitoring using PCs and HMIs.

Suitable for RTU applications where wireless communication is required.

CIM 10 Device Support for Modbus RTU Features:

Supports multiple types of equipment via RTU.

Adjustable address base (0 or 1) and full address range (0-65535).

Supports word and byte swapping (byte order), MSW, LSW, MSB, LSB.

Equipment slave id full range (1-254).

Supported functions: read coil status, read input status, read holding registers, read input registers, force single-coil, force multiple coils, preset multiple registers, connection status.

Supported data types: Boolean, Integer8, Integer16, Integer32, Unsigned8, Unsigned16, Unsigned32, Floating Point 32.

Supports reading/writing data spanning multiple contiguous registers with different sizes and byte orders.

Adjustable polling request time, minimum request interval per register, and polling request timeout.

Supports minimum channel silence on the serial bus.

Allows different communication options (baud rate, byte size, parity, and stop bits) on the same serial bus.

Configuration and Connection Setup:

Modbus RTU Interface: CIM 10 device and RS485 devices interface for Modbus RTU.

Connection Setup on WebUI:

Open WebUI with the provided URL (e.g., 192.168.3.100).

Login with valid credentials (Username: iEdgeAdmin, Password: iEA@12345).

Add new devices using the "+" sign in the WebUI.

Select the protocol (Modbus RTU) and provide a device name.

Save the configuration to see the device on the dashboard.

Example: Secure Meter Connection Setup:

Configure the device under Modbus RTU with details such as protocol, name, destination (e.g., "CIM1"), port settings (RS485E, RS485M, None, 1, 19200, 8, 5).

Add writable tags to update the device data through the WebUI.

Adding Writable Tags:

Configure writable tags by creating queries on the WebUI.

Assign a unique slave ID to each query.

Assign a peripheral name and obtain a Peripheral ID from the CIMCON Customer Success Team.

Add parameters to the query, configuring details such as the communication protocol (Modbus-RTU), slave ID, peripheral ID, and parameter details.

Note:

Users can add a maximum of 32 queries, each supporting up to 16 parameters.

Different Modbus RTU devices from various manufacturers can be added to the WebUI.

After adding queries and parameters, the device status becomes connected, and real-time data can be viewed in the Tag view.

This comprehensive integration guide provides users with the necessary steps to connect Modbus RTU devices to CIM 10, facilitating efficient monitoring and control in industrial applications.

Modbus TCP/IP Integration with CIM 10 Device

Overview

Modbus TCP/IP (Modbus-TCP) is an extension of the Modbus RTU protocol with a TCP interface, designed to run on Ethernet. Combining the Modbus application protocol with the Transmission Control Protocol and Internet Protocol (TCP/IP), it provides a standardized method for data representation, widely used in industrial applications. The CIM 10 device supports Modbus TCP/IP, facilitating efficient communication with various equipment via IP addressing.

CIM 10 Device Support for Modbus TCP/IP Features

Supports Multiple Equipment Types: Allows connectivity to various types of equipment via IP addressing.

Adjustable Address Base: Users can configure the address base as 0 or 1, with support for the full address range (0-65535).

Word and Byte Swapping: Enables flexibility in byte order with options for MSW (Most Significant Word First), LSW (Least Significant Word First), MSB (Most Significant Byte First), and LSB (Least Significant Byte First).

Equipment Slave ID: Supports a full range of slave IDs (1-254).

Supported Functions: Includes read coil status, read input status, read holding registers, read input registers, force single-coil, force multiple coils, preset multiple registers, and connection status.

Supported Data Types: Offers support for Boolean, Integer8, Integer16, Integer32, Unsigned8, Unsigned16, Unsigned32, and Floating Point 32.

Contiguous Register Handling: Supports reading/writing data spanning multiple contiguous registers with different sizes and byte orders.

Adjustable Polling Parameters: Allows users to set polling request time per equipment, minimum request interval per register, polling request timeout, and minimum channel silence.

Communication Flexibility: Permits different communication options such as baud rate, byte size, parity, and stop bits on the same serial bus.

Pre-requisites

Modbus Ethernet TCP/IP Simulator or Modbus TCP/IP device.

Whitelist IP address and Port.

Connection Setup

Open the iEdge 360 device web page with the provided URL (e.g., 192.168.3.100).

Login with valid credentials (Username: iEdgeAdmin, Password: iEA@12345).

Add a new device using the "+" sign.

Select the protocol (Modbus TCP/IP) and provide a proper name for the device.

Save the configuration to see the device on the dashboard.

Click on the "edit" option to configure the device with details like IP address, port, and port timeout.

Example: Modbus TCP/IP Simulator Connection Setup

Configure the device under Modbus TCP/IP with details such as protocol, name, IP address of the simulator or Modbus TCP/IP server, port, and port timeout.

Add writable tags to update device data through the WebUI.

Adding Writable Tags

Configure writable tags by creating queries on the WebUI.

Assign a unique slave ID to each query.

Configure parameters within the query, specifying details such as the start address, number of registers, interval, peripheral ID, peripheral name, and query interval.

OPC UA Integration with CIM 10 Device

Overview

OPC UA (Open Platform Communications United Architecture) is an open and secure data exchange standard for industrial communication. It provides a standardized method for transferring information between servers and clients in a secure and reliable manner. OPC UA is manufacturer-independent, works across various programming languages and operating systems, and supports a flexible mechanism for data exchange between enterprise-type systems and real-world devices.

The CIM 10 device seamlessly integrates OPC UA, supporting communication over TCP on VPNs, through firewalls, and across the internet, WAN, or LAN. This integration enables reading and writing OPC UA variable nodes by node ID, supporting multiple server connections, basic authentication, x509 certificate authentication, and data encryption via RSA standards. Key functionalities include Read, Write, and Subscription, with adjustable polling read time.

CIM 10 Device OPC UA Features

Protocol Support: OPC UA protocol over TCP on VPNs, through firewalls, and across the internet, WAN, or LAN.

Variable Node Operations: Read and write OPC UA variable nodes by node ID.

Multiple Server Connections: Supports connections to multiple OPC UA servers.

Authentication: Basic authentication and authentication through x509 certificates.

Data Encryption: Encryption via RSA Standards for secure data transmission.

Functionalities: Read, Write, and Subscription functionality.

Data Types: Supports various data types including BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, LINT, ULINT, LWORD, REAL, LREAL, CHAR, WCHAR, and STRING.

Pre-Requisites

OPC UA Server (Device) or OPC UA Simulation Server.

OPC UA Client (CIM 10 device).

Internet Connectivity.

Whitelist IP/Port.

OPC UA Client/Server Configuration

Server Side (Prosys OPC UA Simulation Server)

Install Prosys OPC UA Simulation Server on the PC. Download Link

After installation, verify server status through the Status View.

Client Side (CIM 10 Device)

Open the CIM 10 device WebUI using the provided URL (e.g., 192.168.3.100).

Login with valid credentials (Username: iEdgeAdmin, Password: iEA@12345).

Navigate to the "Device" section and add a new OPC UA device.

Configure the device by providing a name, OPC UA Server IP, and other relevant details.

Save the configuration and observe the device status changing to "Connected."

Parameter Configuration

Under the OPC UA device configuration page, set the sampling rate, destination, port details, and OPC UA server URL.

Add writable tags for configuring parameters by specifying Namespace Index, Identifier Type, Identifier, Name, Type, and more.

Configure security settings including Security Mode, Security Policy, and Authentication mode.

Configure variables or objects with specific details such as Namespace Index, Identifier Type, Identifier, Name, and Peripheral ID.

Configuring a Variable or Object

Utilize Prosys Simulation Server to obtain variable or object details.

Fill required information on the CIM 10 device OPC UA configuration page.

Save the configuration.

Tag View

Click on "Tag View" to see real-time values of configured parameters, including data type, timestamp, and status.

This OPC UA integration guide provides detailed steps for configuring OPC UA on the CIM 10 device, ensuring seamless communication with OPC UA servers and facilitating real-time monitoring and control in industrial applications.

BACnet RTU Integration with CIM 10 Device

Pre-Requisite

To verify the functionality of the BACnet RTU application, the following components are required:

BACnet RTU Simulator

BACnet Client (CIM 10 Device)

RS485 Connectivity

BACnet Simulation Device Configuration

BACnet Simulator Installation:

Download and install the BACnet Simulator software from https://polarsoft.com/distribute.asp?TRIAL-098BDC61-D134-4617-B171-22D928CE545D.

After successful installation, open the software.

CIM 10 Device Configuration:

Open the CIM 10 Device web page using the provided URL (e.g., 192.168.3.100).

Log in with the following credentials:

Username: iEdgeAdmin

Password: iEA@12345

Navigate to the "Device" section on the left side of the status page.

Click on the "BACnet RTU" protocol to add a BACnet-supported device.

Configure the following settings:

Name: Provide a relevant name for the BACnet RTU device.

Sampling Rate (Sec): Define the sampling rate in seconds.

Destination: Define or add the list of applications for receiving data.

Click "Save" to create the BACnet device.

The initially created BACnet device status will show as "Disconnected."

Click on "Edit" to configure the BACnet device settings.

BACnet Device Configuration:

Device Settings:

Protocol: BACnet RTU (already selected).

Name: Relevant name given during device creation.

Sampling Rate (Sec): User-defined sampling rate.

Destination: List of applications for data transfer.

Port Interface:

Port Select: Choose the appropriate RS485 port (Serial option for Modbus RTU protocol).

Configure the baud rate and other port details supported by the CIM 10 Device web portal.

Click "Save" to apply the configurations.

Adding Writable Tags:

CIM 10 Device web portal supports writing queries on the physical device. Check the peripheral permissions to verify if the writable option is allowed.

Add details for updating the writable tag.

Note: Refer to the BACnet device datasheet to check the read and write permissions.

Adding BACnet Device:

Provide device details such as Device ID, Device Name, and Peripheral ID.

Device ID: As mentioned in the simulator.

Device Name: Configurable based on simulator instructions.

Peripheral ID: As mentioned in CIMCON iCLOUD.

Add device ID and name, then configure object details by clicking on "Add Objects."

Configure device objects and their instance values based on the simulator.

Mention the object, instance, parameter name, and peripheral ID.

Click "Save" to complete the configuration.

Tag View:

To view the configured parameter values, go to "Tag View" on the configuration page. It displays parameter values along with data types, timestamps, and statuses.

BACnet Simulation Device Configuration

BACnet Simulator Installation:

Download and install the BACnet Simulator software from http://www.scadaengine.com/downloads.php.

After successful installation, the software will display relevant information.

CIM 10 Device Configuration:

Open the CIM 10 Device web page using the provided URL (e.g., 192.168.3.100).

Log in with the following credentials:

Username: iEdgeAdmin

Password: iEA@12345

Navigate to the "Device" section on the left side of the status page.

Click on the "BACnet IP" protocol to add a BACnet-supported device.

Configure the following settings:

Name: Provide a relevant name for the BACnet IP device.

Sampling Rate (Sec): Define the sampling rate in seconds.

Destination: Define or add the list of applications for receiving data.

Port Interface:

Port Select: TCP/IP (for BACnet IP)

Port: Mention the port as defined in the simulator.

Port Timeout: Configure the timeout in seconds as needed.

Listen Interface: Select from the available interfaces (e.g., Eth0, Wlan0, Eth1).

Click "Save" to create the BACnet device.

The initially created BACnet device status will show as "Disconnected."

Click on "Edit" to configure the BACnet device settings.

Adding Writable Tags:

The CIM 10 Device web portal supports writing queries on the physical device. Check the peripheral permissions to verify if the writable option is allowed.

Add details for updating the writable tag.

Note: Refer to the BACnet device datasheet to check the read and write permissions.

Adding BACnet Device:

Provide device details such as Device ID, Device Name, and Peripheral ID.

Device ID: As mentioned in the simulator.

Device Name: Configurable based on simulator instructions.

Peripheral ID: As mentioned in CIMCON iCLOUD.

Add device ID and name, then configure object details by clicking on "Add Objects."

Configure device objects and their instance values based on the simulator.

Mention the object, instance, parameter name, and peripheral ID.

Click "Save" to complete the configuration.

Tag View:

To view the configured parameter values, go to "Tag View" on the configuration page. It displays parameter values along with data types, timestamps, and statuses.

Viewing Data on CIMCON iCLOUD

Log in to CIMCON iCLOUD using the provided credentials.

Refer to the documentation on how to use CIMCON iCLOUD.

Bluetooth Low Energy (BLE) Integration with CIM 10 Device

Overview

Bluetooth Low Energy (BLE) is a low-power wireless technology designed for connecting devices with minimal energy consumption. Operating in the 2.4 GHz frequency, BLE is ideal for applications requiring low power and the transfer of small amounts of data. The benefits of BLE include lower power consumption, support for small data transfers, and cost-effectiveness.

Prerequisites

To integrate BLE with the CIM 10 device, the following prerequisites must be met:

CIM 10 device connected to a USB dongle (e.g., Laird Sensor) supporting Bluetooth 5.0.

CIM 10 device and peripheral device with Bluetooth support for data transmission.

Adding a BLE Device

Follow the steps below to add a device that supports the BLE protocol:

Open the CIM 10 device web page using the provided URL (e.g., 192.168.3.100).

Log in with the following credentials:

Username: iEdgeAdmin

Password: iEA@12345

On the status page, click on the "Device" option on the left.

Click the "+" sign to add a new device.

Select the BLE protocol and provide a name for the device.

Click "Save" to create the BLE device.

The device details will be displayed on the dashboard, showing the device name, status (Connected, Disconnected, Unknown), Tag View (live data), Edit (configuration), and Remove options.

Basic BLE Configuration

Under the BLE section, configure the following:

Device Settings

Name: Already configured during device creation.

Protocol: Already selected during device creation.

Sampling Rate (Min): Define the sampling rate in minutes.

Destination: Specify the destination where data will be sent (e.g., "CIM2" for CIMCON iCLOUD).

Port Settings

Port Select: Choose the "MAC" option.

MAC: Enter the MAC ID of the Bluetooth peripheral device.

UUID Configuration

Click "Add UUID" to add up to 16 UUIDs. Configure the following:

Characteristic UUID: Obtain from the peripheral device datasheet.

Base UUID: e.g., 0C4CXXXX-7700-46F4-AA96-D5E974E32A54.

UUID(0xXXXX): e.g., 0x3001.

Peripheral ID: As defined in CIMCON iCLOUD.

Operation: Choose between Polling and Notify.

Peripheral Name: Configure the peripheral name.

Parameter Configuration

Click "+ Add Parameter" to add up to 16 parameters. Configure the following:

Data Type: Select the data type from the drop-down.

Factor: As defined in the peripheral datasheet.

Output Type: Choose from the drop-down list.

Name: Configure the parameter name.

Viewing Data

To view the current status of the device, click on "Tag View." Data will be displayed based on the defined parameters.

Tag Services Overview and Usage

Overview

Tag services play a crucial role in configuring and viewing device parameters associated with unique UUIDs. These services provide users with a customizable option to facilitate the creation of complex applications. Each device or peripheral has a unique UUID (RAW Topic), allowing users to easily locate and configure the required parameters. The UUID may change when parameters are removed and then added, preventing any duplication.

Benefits of TAG:

Customization: Tags offer a customizable option for easy configuration of complex applications.

UUID Identification: Each device/peripheral has a unique UUID (RAW Topic), simplifying parameter identification.

Avoid Duplication: Changing UUIDs when parameters are added or removed helps avoid duplication.

Real-time Updates: Users can receive real-time updates when devices are connected using SDK or Node-RED.

Payload Changes: Users can modify payloads using JSON parsing methods.

How to Use Tag Services

Open the webUI page with the provided URL by CIMCON support (e.g., 192.168.1.100).

Login with the following credentials:

User Name: iEdgeAdmin

Password: iEA@12345

Click on "Tags" as highlighted in the menu.

Adding Tags

Click on the "+" icon to add tags from the list of created devices and peripherals.

Search for tags by name, device, or tag.

Select the device and tag parameters from the drop-down menu.

Click "Save" to add the tag. The added tag will appear on the main Tags page.

Example:

User searches by the device name created under peripherals, selects the parameter from the drop-down, and clicks "Save."

Viewing Tag Parameters

Tags can be viewed in two ways:

A. Using WebUI:

Log in to WebUI.

Go to Peripherals > Analog Input.

View the current parameter values under "Value."

B. Using C SDK:

Log in using the provided credentials (username: root, password: cimcon).

Connect the iEDGE 360 device to the PC via Ethernet or RS232 port.

Navigate to the home list to see available SDKs.

View the test file from the list under the selected service.

Mention the UUID as presented in WebUI.

Display all parameter values for parameters configured in WebUI. The values update automatically based on the defined sampling rate.

Integration

Overview

This Document also help to configure any CIMCON iCloud using Webui. This Document help user to configure their peripheral device data to be send.

How to configure CIM 10 device using Webui to enable services of CIMCON iCloud

Open CIM 10 device web page with provided URL by CIMCON support team (192.168.1.100) and webpage look like as below. Default configuration is as below

IP:192.168.3.100

Subnet:255.255.255.0

Gateway:192.168.3.1

Login with Valid Username and Password. User Name: iEdgeAdmin Password: iEA@12345.

After successfully login in, you can able see the status page of the application as below. On the left side of the status page, there is an option "Integration", click on the "Integration".

Configuration of the CIMCON iCLOUD at WebUI

Click on integration. Refer to the below image for how to create an Integration using Transport. Webui provides user to configure and select transport from the drop-down menu. Below are the steps to create integration.

1. Click on Integration

2. Click on “+” as shown below. Pop window will open.

3. In the Pop-up menu, the user first creates an Integrator for MQTT as a transporter and the Thingsboard as a transporter.

MQTT transporter is used between Vibit to iEDGE 360 device.

Thingsboard transport is used between iEDGE 360 device and CIMCON iCloud.

4. User can configure the name

5. Click on save

Once the integration is added successfully it will show on the integration list as below images.

After that need to click the edit option, as highlighted below, and provide details to connect the device.

MQTT

Note: MQTT is used for external peripheral device (like vibration sensor) which is sending data over Wi-Fi to CIMCON iCloud.

After clicking on the edit option, the below screen will show, the user needs to update the required as per below.

1. Client id

2. End Point

3. Port (MQTT port)

Incoming Message and Outgoing Message Configuration

- Webui support to add multiple Incoming Message and Outgoing message. To Add user needs to click “+ Add Incoming Message” to add Incoming Message and click “+ Add Outgoing Message” to add Outgoing Message.

Incoming Message required input

Type: Data or Event

App Name: User need to mention the destination files for example “service_thingsboard_2”

Topic: need to provide the Topic name for example “v1/devices/me/telemetry”

Outgoing Message required input

Type: Data or Event

App Name: User need to mention the destination files for example “service_thingsboard_2”

Topic: need to provide the Topic name for example “v1/devices/me/telemetry”

CIMCON iCLOUD

After clicking on the edit option, The below screen will show, the user needs to update the required as per below.

1. Client id

2. End point

3. Port (MQTT port)

4. Add authentication

ID: need to mention Vibit Sensor ID define in CIMCON iCloud

User: Need to mention the Vibit Sensor user name as mention on CIMCON iCloud. CIMCON customer success team will provide User name.

Password: Need to mention the Vibit Sensor password as mention on CIMCON iCloud. CIMCON customer success team will provide Password.

Incoming Message and Outgoing Message Configuration

Webui support to add multiple Incoming Message and Outgoing message. To Add user needs to click "+ Add Incoming Message" to add Incoming Message and click "+ Add Outgoing Message" to add Outgoing Message.

Incoming Message required input

Uplink: Data from Vibration sensor to incoming of MQTT and outgoing to thingsboard

ID: Need to mention peripheral id as mention CIMCON iCloud

App Name: User need to mention the destination files for example "service_thingsboard_2"

Topic: need to provide the Topic name for example "v1/devices/me/telemetry"

Outgoing Message required input

Downlink: Data from Vibration sensor to incoming Things board and outgoing to MQTT

ID: Need to mention peripheral id as mention CIMCON iCloud.

App Name: User need to mention the destination files for example "service_thingsboard_2"

Topic: need to provide the Topic name for example "v1/devices/me/telemetry"

Users can see the parameter on CIMCON iCloud against the peripheral id mentioned.

SDK

What is SDK

SDK stands for software development kit or devkit for short. It’s a set of software tools and programs used by developers to create applications for several platforms.

SDK tools will include a range of things, including libraries, documentation, code samples, processes, and guides for developers to integrate into their apps. SDKs are designed to be used for specific platforms or programming languages.

Thus you would need an Android SDK toolkit to build an Android app, an iOS SDK to build an iOS app, a VMware SDK for integrating with the VMware platform, or a Nordic SDK for building Bluetooth or wireless products, and so on.

SDK vs API: Difference

API is a code that allows two Applications (Apps) to communicate and exchange data.

This API is as a truck delivery truck bringing your app’s request to some other software, then bringing the response back to your app

SDK is a larger kit that can contain multiple API’s Plus many other tools to connect the software together.

Benefit of SDK

SDK makes the task much easier to quickly integrate customer’s existing tech stack, which further helps in reducing the Sales cycle.

Increase scalability of product to provide a platform to integrate other tools with our product.

Efficient development and faster deployment as SDK provide libraries, tools which help developers not to start code from scratch.

SDK provides better control over the element of your UI that shows in other Apps.

Thorough documentation to explain how your code works

Enough functionality so its adds value to other apps.

Supported Language

C

Python

Configuration and Installation Steps

C SDK compile & run with gcc compiler as below:

· gcc -o destination source file(.c file) –lrtucore (w/o transport)

· gcc -o destination source file(.c file) –lrtucore –ljson-c (for Json extract e.g. Transport)

· ./destination

To run Python SDK as below:

· python3 source file

Prerequisite

Device must be installed with the latest package for all devices and services.

For any changes in device configuration, Device restart is a must.

Device must be configured and check parameters on webUI.

For Python SDK edgeSDKpy.py and subscriber.py and other modules must be available in the same directory.

Client can see the last saved payload details can be read by name, Address and all method.

Note: Consider device is configured once and get data one time, still client can see the last saved payload details by defined method. In case server disconnected, device not responding over a medium, any change in webUI etc. still user able to see the last saved payload using define method.

SDK Overview

What is SDK?

SDK stands for Software Development Kit, or devkit for short. It is a set of software tools and programs used by developers to create applications for specific platforms. SDK tools include a variety of resources such as libraries, documentation, code samples, processes, and guides that developers can integrate into their applications. These kits are tailored for particular platforms or programming languages.

Developers use an Android SDK to build Android apps, an iOS SDK for iOS apps, a VMware SDK for integrating with the VMware platform, a Nordic SDK for building Bluetooth or wireless products, and so on.

SDK vs API: Difference

API (Application Programming Interface):

An API is a code that enables communication and data exchange between two applications.

It acts like a delivery truck, bringing requests from one app to another and returning with the response.

SDK (Software Development Kit):

An SDK is a larger kit that can contain multiple APIs and various tools to connect software components.

It provides a comprehensive set of resources, including APIs, to facilitate software development.

Benefits of SDK

Easier Integration:

SDKs make it easier to integrate with existing tech stacks, reducing the sales cycle.

Increased Scalability:

They provide a platform to integrate other tools with the product, enhancing scalability.

Efficient Development:

SDKs offer libraries and tools that help developers avoid starting from scratch, leading to efficient development and faster deployment.

Better UI Control:

SDKs provide better control over UI elements that appear in other apps.

Thorough Documentation:

SDKs come with thorough documentation explaining how the code works.

Added Value:

They offer enough functionality to add value to other apps.

Supported Languages

C

Python

Configuration and Installation Steps

C SDK

To compile and run C SDK with GCC compiler:

bash

Copy code

```
gcc -o destination source.c -lrtucore     # Without transport
gcc -o destination source.c -lrtucore -ljson-c     # With JSON extraction (e.g., Transport)
./destination
```

Python SDK

To run Python SDK:

bash

Copy code

python3 source.py

Prerequisites

Device must be installed with the latest package for all devices and services.

For any changes in device configuration, device restart is required.

Device must be configured, and parameters must be checked on the webUI.

For Python SDK, edgeSDKpy.py, subscriber.py, and other modules must be available in the same directory.

Clients can view the last saved payload details by name, address, and all methods.

Note: Even if the server is disconnected, the device is not responding, or there are changes in the webUI, clients can still see the last saved payload details using the defined methods.

Peripheral Functionality with CIM 10 Device

The CIM 10 device supports SDK APIs for peripherals, specifically for Analog Input (AI) and Digital Input (DI). Here's a guide on how users can configure and interact with these peripherals using the C SDK.

Using C SDK for Peripheral Configuration

CIM 10 Device Connection

Connect the CIM 10 device to a PC via the Ethernet port or RS232 port.

Login with the following details:

Username: root

Password: cimcon

Peripheral Configuration

Provide the path where test files are stored.

Run # ls to see the list of peripheral devices.

Select "analog" from the list of peripheral devices.

Run # ls again to see the list of test files to be executed for configuration.

Peripheral Functions in C SDK

1. Read by Address

c

Copy code

char *readAiByAddress(int address);

// Passing Argument: Pin number of AI

2. Read by Name

c

Copy code

```
char *readAiByName(const char *name);
// Passing Argument: Name of AI pin
```

3. Read by All

c

Copy code

```
int readAiByAll(<callback(data)>);
// Show all values at once
```

4. Read by Change

c

Copy code

```
readAiByChange(const char *pinname, <callback(data, dataSize)>);
// Show all values and update regularly as per the defined sampling rate
```

Sample Code for AI SDK

c

Copy code

```
// Sample code for Read by Address
char *data = readAiByAddress(1);
printf("AI Value: %s\n", data);
```

How to Use

Read by Address:

Write 1 to select Read by address.

Select the pin number of the device mentioned in the webUI.

Read by Name:

Write 2 to select Read by name.

Mention the AI name as shown in the webUI (e.g., AI_1).

Read by All:

Write 3 to select Read by All.

Show all parameter values configured in the webUI.

Read by Change:

Write 4 to select Read by Change.

Show all parameter values, updating regularly as per the defined sampling rate.

For Digital Input (DI)

The process is similar, with corresponding methods for DI.

Python SDK for Transport Configuration

Connect the CIM 10 device to a PC via the Ethernet port or RS232 port.

Login with the following details:

Username: root

Password: cimcon

Go to the home list to see available SDKs.

Select the desired file from the list.

Choose the test file required for the selected protocol.

Transport Methods in Python SDK

Read By Address:

python

Copy code

```
edgeSdkPy.SendToEndPoint_Py(char *devicename, char *payload, int payloadLen)
# Passing Argument: Device Name, Device ID, Object (0 numeric), Identifier ID, Peripheral ID
```

Read By Name:

python

Copy code

edgeSdkPy.ReceiveFrom_Py(const char *devicename, callback(data, dataSize))

# Passing Argument: Device Name, Device ID, Parameter Name, Peripheral ID

Read By All:

python

Copy code

edgeSdkPy.Sendto_py(char *devicename, char *payload, int payloadLen)

# Passing Argument: Device Name

How to Use

Read By Address:

python

Copy code

edgeSdkPy.SendToEndPoint_Py("Device1", "payload", 8)

Read By Name:

python

Copy code

edgeSdkPy.ReceiveFrom_Py("Device1", callback)

Read By All:

python

Copy code

edgeSdkPy.Sendto_py("Device1", "payload", 8)

These methods facilitate communication with peripherals and transport configurations using the provided SDKs.

BACnet Functionality with CIM 10 Device

The CIM 10 device supports the C SDK for BACnet services, allowing users to read hardware data from BACnet-supported devices. This SDK provides the ability to read data for both BACnet RTU and BACnet IP protocols.

Using C SDK for BACnet Configuration

CIM 10 Device Connection

Connect the CIM 10 device to a PC via the Ethernet port or RS232 port.

Login with the following details:

Username: root

Password: cimcon

BACnet Configuration

Go to the home list to see available SDKs.

Show the path of the selected protocol.

View the test file under the selected protocol.

Show the common methods: Read by Address, Read by Name, Read by All, Read by Change.

BACnet Methods in C SDK

Read By Address:

c

Copy code

```
char *readBACnetByAddress(<DeviceName>, int deviceid, int objtype, int instanceid);
// Passing Argument: Device name, Device ID, Object Type, Instance
```

Read By Name:

c

Copy code

```
char *readBACnetByName(<DeviceName>, int deviceid, const char *paramName);
// Passing Argument: Device name, Device ID, Parameter Name
```

Read By All:

c

Copy code

```
int readBACnetByAll(<DeviceName>,<callback(data)>);
// Passing Argument: Device name
```

Read By Change:

c

Copy code

```
readBACnetByChange(<DeviceName>, <callback(data, dataSize)>);
// Passing Argument: Device name
```

Sample Code for BACnet SDK

1. Read By Address

c

Copy code

```
// Sample code for Read by Address
char *data = readBACnetByAddress("Device1", 1, 0, 1);
printf("BACnet Value: %s\n", data);
```

2. Read By Name

c

Copy code

```
// Sample code for Read by Name
char *data = readBACnetByName("Device1", 1, "Temperature");
printf("BACnet Value: %s\n", data);
```

3. Read By All

c

Copy code

```
// Sample code for Read by All
int result = readBACnetByAll("Device1", callback);
```

4. Read By Change

c

Copy code

```
// Sample code for Read by Change
readBACnetByChange("Device1", callback);
```

BACnet Object Type Table

Sr. NO	Object Type - Name	Object Type - Value
1	Analog Input	0
2	Analog Output	1
3	Analog Value	2
4	Binary Input	3
5	Binary Output	4
...	...	...

How to Use

Read By Address:

Write 1 to select Read by address.

Provide the Device name, Device ID, Object type, and Instance.

Read By Name:

Write 2 to select Read by name.

Provide the Device name, Device ID, and Parameter name.

Read By All:

Write 3 to select Read by All.

Show all parameter values configured in the webUI.

Read By Change:

Write 4 to select Read by Change.

Show all parameter values, updating as per the defined sampling rate.

BACnet Configuration with Python SDK

CIM 10 Device Connection

Connect the CIM 10 device to a PC via the Ethernet port or RS232 port.

Login with the following details:

Username: root

Password: cimcon

Python SDK for BACnet Configuration

Show the path of the selected protocol.

View the test file under the selected protocol.

Show the Test file selected.

Show the common methods: Read by Address, Read by Name, Read by All, Read by Change.

BACnet Methods in Python SDK

Read By Address:

python

Copy code

```
edgeSdkPy.readBACnetByAddress_py(<DeviceName>, int deviceid, int objtype, int instanceid);
# Passing Argument: Device name, Device ID, Object Type, Instance
```

Read By Name:

python

Copy code

edgeSdkPy.readBACnetByName_py(<DeviceName>, int deviceid, const char *paramName);

# Passing Argument: Device name, Device ID, Parameter Name

Read By All:

python

Copy code

edgeSdkPy.readBACnetByAll_py(<DeviceName>,<callback(data)>);

# Passing Argument: Device name

Read By Change:

python

Copy code

edgeSdkPy.readBACnetByChange_py(<DeviceName>, <callback(data, dataSize)>);

# Passing Argument: Device name

How to Use

Read By Address:

python

Copy code

edgeSdkPy.readBACnetByAddress_py("Device1", 1, 0, 1)

Read By Name:

python

Copy code

edgeSdkPy.readBACnetByName_py("Device1", 1, "Temperature")

Read By All:

python

Copy code

edgeSdkPy.readBACnetByAll_py("Device1", callback)

Read By Change:

python

Copy code

edgeSdkPy.readBACnetByChange_py("Device1", callback)

These methods provide a way to configure and interact with BACnet peripherals using the CIM 10 device through both C SDK and Python SDK.

MODBUS Functionality with CIM 10 Device

The CIM 10 device supports the C SDK for MODBUS protocol supported devices, allowing users to read hardware data from MODBUS-supported devices. This SDK provides the ability to read data for both MODBUS RTU and MODBUS IP protocols.

Using C SDK for MODBUS Configuration

CIM 10 Device Connection

Connect the CIM 10 device to a PC via the Ethernet port or RS232 port.

Login with the following details:

Username: root

Password: cimcon

MODBUS Configuration

Go to the home list to see available SDKs.

Show the test file from the list under the selected protocol.

Mention the Device name as configured in the webUI.

Show the common methods: Read by Address, Read by Name, Read by All, Read by Change.

MODBUS Methods in C SDK

Read By Address:

c

Copy code

readModbusByAddress(<DeviceName>, <queryNum>, <ParamAddress>);

// Passing Argument: Device Name, Query Number, Parameter Address

Read By Name:

c

Copy code

readModbusByName(<Devicename>, <ParamName>);

// Passing Argument: Device Name, Parameter Name

Read By All:

c

Copy code

```
readModbusByAll(<DeviceName>, <callback(data)>);
// Passing Argument: Device Name
```

Read By Change:

c

Copy code

```
readModbusByChange(<DeviceName>, <callback(data, dataSize)>);
// Passing Argument: Device Name
```

Sample Code for MODBUS SDK in C

1. Read By Address

c

Copy code

```
// Sample code for Read by Address
readModbusByAddress("Device1", 1, "1001");
```

2. Read By Name

c

Copy code

```
// Sample code for Read by Name
readModbusByName("Device1", "Temperature");
```

3. Read By All

c

Copy code

```
// Sample code for Read by All
readModbusByAll("Device1", callback);
```

4. Read By Change

c

Copy code

```
// Sample code for Read by Change
readModbusByChange("Device1", callback);
```

How to Use

Read By Address:

Write 1 to select Read by address.

Provide the Device name and Query number.

Users need to add Query address mentioned at webUI under the selected Query number.

Read By Name:

Mention the Device name as mentioned in webUI.

Write 2 to select Read by name.

Provide the Parameter name mentioned at webUI.

Read By All:

Mention the Device name as mentioned in webUI.

Write 3 to select Read by All.

Show all parameter values configured in webUI.

Read By Change:

Mention the Device name as mentioned in webUI.

Write 4 to select Read by Change.

Show all parameter values, updating as per the defined sampling rate.

MODBUS Configuration with Python SDK

CIM 10 Device Connection

Connect the CIM 10 device to a PC via the Ethernet port or RS232 port.

Login with the following details:

Username: root

Password: cimcon

Python SDK for MODBUS Configuration

Go to the home list to see available SDK files.

Show the test file from the list under the selected protocol.

Select the Test file.

Show the common methods: Read by Address, Read by Name, Read by All, Read by Change.

MODBUS Methods in Python SDK

Read By Address:

python

Copy code

readModbusByAddress_py(<DeviceName>, <queryNum>, <ParamAddress>);

# Passing Argument: Device Name, Query Number, Parameter Address

Read By Name:

python

Copy code

readModbusByName_py(<Devicename>, <ParamName>);

# Passing Argument: Device Name, Parameter Name

Read By All:

python

Copy code

readModbusByAll_py(<DeviceName>, <callback(data)>);

# Passing Argument: Device Name

Read By Change:

python

Copy code

readModbusByChange_py(<DeviceName>, <callback(data, dataSize)>);

# Passing Argument: Device Name

Sample Code for MODBUS SDK in Python

1. Read By Address

python

Copy code

# Sample code for Read by Address

readModbusByAddress_py("Device1", 1, "1001")

2. Read By Name

python

Copy code

# Sample code for Read by Name

readModbusByName_py("Device1", "Temperature")

3. Read By All

python

Copy code

# Sample code for Read by All

readModbusByAll_py("Device1", callback)

4. Read By Change

python

Copy code

# Sample code for Read by Change

readModbusByChange_py("Device1", callback)

These methods provide a way to configure and interact with MODBUS peripherals using the CIM 10 device through both C SDK and Python SDK.

OPC UA Functionality with CIM 10 Device

The CIM 10 device supports the C SDK for OPC UA protocol supported devices, enabling users to read hardware data from OPC UA supported devices. This SDK provides the ability to read data for the OPC UA protocol.

Using C SDK for OPC UA Configuration

CIM 10 Device Connection

Connect the CIM 10 device to a PC via the Ethernet port or RS232 port.

Login with the following details:

Username: root

Password: cimcon

OPC UA Configuration

Go to the home list to see available SDKs.

Select the file from the list.

Show the test file under the selected protocol.

Select the required test file.

Show the common methods: Read by Address, Read by Name, Read by All, Read by Change.

OPC UA Methods in C SDK

Read By Address:

c

Copy code

readOPCUAByAddress(<DeviceName>, <id, dtype, ParamAddress>);

// Passing Argument: Device Name, Name space index (Index no.), Identifier type, Identifier

Read By Name:

c

Copy code

readOPCUAByName(<Devicename>, <ParamName>);

// Passing Argument: Device Name, Parameter Name

Read By All:

c

Copy code

readOPCUAByAll(<DeviceName>, <callback(data)>);

// Passing Argument: Device Name

Read By Change:

c

Copy code

readOPCUAByChange(<DeviceName>, <callback(data, dataSize)>);

// Passing Argument: Device Name

Sample Code for OPC UA SDK in C

1. Read By Address

c

Copy code

// Sample code for Read by Address

readOPCUAByAddress("Device1", 1, 0, "Temperature");

2. Read By Name

c

Copy code

// Sample code for Read by Name

readOPCUAByName("Device1", "Temperature");

3. Read By All

c

Copy code

```
// Sample code for Read by All
readOPCUAByAll("Device1", callback);
```

4. Read By Change

c

Copy code

```
// Sample code for Read by Change
readOPCUAByChange("Device1", callback);
```

These methods provide a way to configure and interact with OPC UA peripherals using the CIM 10 device through C SDK.

OPC UA Configuration with Python SDK

CIM 10 Device Connection

Connect the CIM 10 device to a PC via the Ethernet port or RS232 port.

Login with the following details:

Username: root

Password: cimcon

Python SDK for OPC UA Configuration

Go to the home list to see available SDK files.

Show the test file from the list under the selected protocol.

Select the test file.

Show the common methods: Read by Address, Read by Name, Read by All, Read by Change.

OPC UA Methods in Python SDK

Read By Address:

python

Copy code

```
readModbusByAddress_py(<DeviceName>, <queryNum>, <ParamAddress>);
# Passing Argument: Device Name, Name space index (Index no.), Identifier type, Identifier
```

Read By Name:

python

Copy code

readModbusByName_py(<Devicename>, <ParamName>);

# Passing Argument: Device Name, Parameter Name

Read By All:

python

Copy code

readModbusByAll_py(<DeviceName>, <callback(data)>);

# Passing Argument: Device Name

Read By Change:

python

Copy code

readModbusByChange_py(<DeviceName>, <callback(data, dataSize)>);

# Passing Argument: Device Name

Sample Code for OPC UA SDK in Python

1. Read By Address

python

Copy code

# Sample code for Read by Address

readModbusByAddress_py("Device1", 1, "Temperature");

2. Read By Name

python

Copy code

# Sample code for Read by Name

readModbusByName_py("Device1", "Temperature");

3. Read By All

python

Copy code

# Sample code for Read by All

readModbusByAll_py("Device1", callback);

4. Read By Change

python

Copy code

Sample code for Read by Change

readModbusByChange_py("Device1", callback);

These methods provide a way to configure and interact with OPC UA peripherals using the CIM 10 device through both C SDK and Python SDK.

Configuring OPC DA using CSDK on the CIM 10 device involves several steps. Follow the detailed guide below:

1. Connect CIM 10 Device to PC

Connect the CIM 10 device to a PC via either the Ethernet port or RS232 port to enable remote communication.

2. Login to CIM 10 Device

Use the following login credentials:

Username: root

Password: cimcon

3. Navigate to OPCDA List

Go to the OPCDA list to access the available SDK files.

4. Select Test File

Choose the appropriate test file from the list under the OPC DA protocol.

5. Common Methods Overview

Under OPC DA, there are four common methods used in the C SDK to read hardware data:

a. Read By Address

c

Copy code

char * readOPCDAByAddress(const char * devicename, const char * address)

Device Name: Specify the name of the device.

Address: Provide the address information.

b. Read By Name

c

Copy code

char * readOPCDAByName(const char * devicename, const char * parameter_name)

Device Name: Mention the device name.

Parameter Name: Specify the parameter name.

c. Read By All

c

Copy code

readOPCDAByAll(const char *devicename, Callback (data))

Device Name: Provide the device name.

Callback (data): Utilize a callback function to handle the retrieved data.

d. Read By Change

c

Copy code

readOPCUAByChange(<DeviceName>, <callback(data, dataSize)>)

Device Name: Specify the device name.

Callback (data, dataSize): Use a callback function to manage data changes.

6. Code for OPC DA

The sample C SDK code for OPC DA functions is as follows:

1. Read By Address

To read the value by address, follow these steps:

Write 1 to select Read by Address.

Specify the device name as mentioned in the webUI.

Provide the OPC name defined in the webUI.

The function will display details of the OPC name, including the present value.

2. Read By Name

For reading the value by name:

Write 2 to select Read by Name.

Mention the device name as specified in webUI.

Specify the parameter name as mentioned in webUI.

The function will display details of the parameter name, including the present value.

3. Read By All

To read values by addressing all parameters:

Write 3 to select Read by All.

Specify the device name as mentioned in webUI.

The function will show all parameter values configured in the webUI at once.

4. Read By Change

For reading values that change over time:

Write 4 to select Read by Change.

Specify the device name as mentioned in webUI.

The function will display all parameter values, updating automatically as per the defined sampling rate.

Configuring Transport using CSDK on the CIM 10 device involves several steps. Below is a detailed guide:

1. Connect CIM 10 Device to PC

Connect the CIM 10 device to a PC via either the Ethernet port or RS232 port to enable remote communication.

2. Login to CIM 10 Device

Use the following login credentials:

Username: root

Password: cimcon

3. Navigate to SDK in Home List

Navigate to the home list to access available SDK files.

4. Select Transport SDK

Choose the relevant transport SDK file from the list under the selected protocol.

5. Select Test File

Select the required test file from the list to begin configuring the transport functionality.

6. Transport Methods Overview

Under Transport, there are three main methods in the C SDK to handle data transfer:

a. Send to Endpoint (Uplink)

c

Copy code

SendToEndPoint(char * devicename, char *payload, int payloadLen)

Device Name: Specify the name of the device.

Payload: Provide the data payload to be sent.

Payload Length: Specify the length of the payload.

b. Received from (Downlink)

c

Copy code

ReceiveFrom(const char * devicename, callback(data, dataSize))

Device Name: Mention the device name.

Callback (data, dataSize): Utilize a callback function to handle received data.

c. Send to (Uplink)

c

Copy code

SendTo(char * devicename, char *payload, int payloadLen)

Device Name: Provide the device name.

Payload: Specify the data payload to be sent.

Payload Length: Specify the length of the payload.

7. Code for Transport

The sample C SDK code for transport functions is as follows:

1. Send to Endpoint

To send data to the endpoint:

Select the "Transport by Address" method.

Specify the device name, device ID, object type, instance ID, and peripheral ID as defined in the webUI.

After entering the required details, the function will display parameter details.

2. Received from (Downlink)

For receiving data from the downlink:

Select the "Transport by Address" method.

Specify the device name, device ID, parameter name, and peripheral ID as defined in the webUI.

After entering the required details, the function will display parameter details.

8. Configure Transport Using Python

To configure transport using Python, follow these steps:

a. Go to Home List

Access the home list to view available SDK files.

b. Select SDK File

Choose the relevant SDK file from the list under the selected protocol.

c. Select Test File

Choose the required test file from the list.

d. Transport Methods in Python

The Python SDK provides similar transport methods as the C SDK:

edgeSdkPy.SendToEndPoint_Py: To send data to the endpoint.

edgeSdkPy.ReceiveFrom_Py: To receive data from the downlink.

edgeSdkPy.Sendto_py: To send data to the uplink.

9. Adjusting Parameters

Ensure to adjust parameters such as device name, device ID, object type, instance ID, peripheral ID, and payload as required based on the webUI configuration.

1. Connect CIM 10 Device to PC

Connect the CIM 10 device to a PC via either the Ethernet port or RS232 port to enable remote communication.

2. Login to CIM 10 Device

Use the following login credentials:

Username: root

Password: cimcon

3. Navigate to SDK in Home List

Navigate to the home list to access available SDK files.

4. Select Transport SDK

Choose the relevant transport SDK file from the list under the selected protocol.

5. Select Test File

Select the required test file from the list to begin configuring the transport functionality.

6. Transport Methods Overview

Under Transport, there are three main methods in the C SDK to handle data transfer:

a. Send to Endpoint (Uplink)

c

Copy code

SendToEndPoint(char * devicename, char *payload, int payloadLen)

Device Name: Specify the name of the device.

Payload: Provide the data payload to be sent.

Payload Length: Specify the length of the payload.

b. Received from (Downlink)

c

Copy code

ReceiveFrom(const char * devicename, callback(data, dataSize))

Device Name: Mention the device name.

Callback (data, dataSize): Utilize a callback function to handle received data.

c. Send to (Uplink)

c

Copy code

SendTo(char * devicename, char *payload, int payloadLen)

Device Name: Provide the device name.

Payload: Specify the data payload to be sent.

Payload Length: Specify the length of the payload.

7. Code for Transport

The sample C SDK code for transport functions is as follows:

1. Send to Endpoint

To send data to the endpoint:

Select the "Transport by Address" method.

Specify the device name, device ID, object type, instance ID, and peripheral ID as defined in the webUI.

After entering the required details, the function will display parameter details.

2. Received from (Downlink)

For receiving data from the downlink:

Select the "Transport by Address" method.

Specify the device name, device ID, parameter name, and peripheral ID as defined in the webUI.

After entering the required details, the function will display parameter details.

8. Configure Transport Using Python

To configure transport using Python, follow these steps:

a. Go to Home List

Access the home list to view available SDK files.

b. Select SDK File

Choose the relevant SDK file from the list under the selected protocol.

c. Select Test File

Choose the required test file from the list.

d. Transport Methods in Python

The Python SDK provides similar transport methods as the C SDK:

edgeSdkPy.SendToEndPoint_Py: To send data to the endpoint.

edgeSdkPy.ReceiveFrom_Py: To receive data from the downlink.

edgeSdkPy.Sendto_py: To send data to the uplink.

9. Adjusting Parameters

Ensure to adjust parameters such as device name, device ID, object type, instance ID, peripheral ID, and payload as required based on the webUI configuration.