kaspersky

Kaspersky IoT Secure Gateway 100

Contents

About Kaspersky IoT Secure Gateway 100

Kaspersky IoT Secure Gateway 100 (hereinafter also referred to as the *gateway*) is a software/hardware system based on the Siemens SIMATIC IOT2040 device for the industrial internet of things (hereinafter also referred to as the *Siemens SIMATIC IOT2040* or the *device*) that has the KasperskyOS operating system and application software installed. Kaspersky IoT Secure Gateway 100 is designed to work as a secure gateway for the industrial internet of things (IIoT) in an enterprise network.

This document contains a full description of the software portion of Kaspersky IoT Secure Gateway 100 and provides basic information about the hardware portion. For complete information on the Siemens SIMATIC IOT2040 device, please refer to the device User Guide. For information about the technical specifications of the Siemens SIMATIC IOT2040 device, please refer to the manufacturer's website.

Kaspersky IoT Secure Gateway 100 performs the following functions:

- Uses the OPC UA protocol to receive data from the OPC UA server residing in the internal enterprise network.
- Forwards data received over the MQTT protocol to the MQTT broker.
- Ensures the cybersecurity of the OPC UA server and supports the transmission of encrypted data.

Installation and initial configuration of Kaspersky IoT Secure Gateway 100 software is performed by experts of APROTECH LLC or its partners.

Kaspersky IoT Secure Gateway 100 is started when the Siemens SIMATIC IOT2040 device is powered on. Kaspersky IoT Secure Gateway 100 is stopped when the device is disconnected from the network.

Distribution kit

The distribution kit for Kaspersky IoT Secure Gateway 100 includes the following components:

- Siemens SIMATIC IOT2040 device for the industrial internet of things with the manufacturer-supplied configuration.
- Technical documentation for the Siemens SIMATIC IOT2040 device with the manufacturer-supplied configuration.
- MicroSD card with the Kaspersky IoT Secure Gateway 100 software pre-installed. The microSD card is installed in the microSD slot on the Siemens SIMATIC IOT2040 device.
- USB-UART adapter.
- File named `legal_notices.txt` that contains information about third-party code and resides on the microSD card.
- QR code containing a link to the electronic version of the User Guide for the software portion of Kaspersky IoT Secure Gateway 100 and information about the Kaspersky IoT Secure Gateway 100 software version (Release Notes).

Hardware and software requirements

Kaspersky IoT Secure Gateway 100 operating requirements

Kaspersky IoT Secure Gateway 100 works only on a Siemens SIMATIC IOT2040 device for the industrial internet of things.

To enable Kaspersky IoT Secure Gateway 100 to receive data from the OPC UA server over the OPC UA protocol, you need to configure the data transfer settings on this server. You can read about the OPC UA protocol specification on the developer's website. Kaspersky IoT Secure Gateway 100 version 2.0 supports OPC UA protocol version 1.04.

To enable Kaspersky IoT Secure Gateway 100 to forward data received from the OPC UA server to the MQTT broker over the MQTT protocol, you need to configure the MQTT broker to receive data from Kaspersky IoT Secure Gateway 100. You can read the MQTT protocol specification on the developer's website. Kaspersky IoT Secure Gateway 100 version 2.0 supports MQTT protocol version 3.1.1.

Kaspersky IoT Secure Gateway 100 configuration and diagnostic requirements

To perform gateway configuration and diagnostics, you will need a computer running a Linux operating system with a connected device that can read and write microSD memory cards and an available USB port for connecting the USB-UART adapter from the distribution kit.

The following applications must be installed on the computer:

- Simple text editor (for example, nano, mcedit, Vim, gedit, or Kate). We recommend using a text editor that supports highlighting of JSON syntax.

  To make changes to the gateway settings, you will need to connect the microSD card to the computer, mount the ext3 partitions of the card, and edit the JSON configuration files in the text editor.

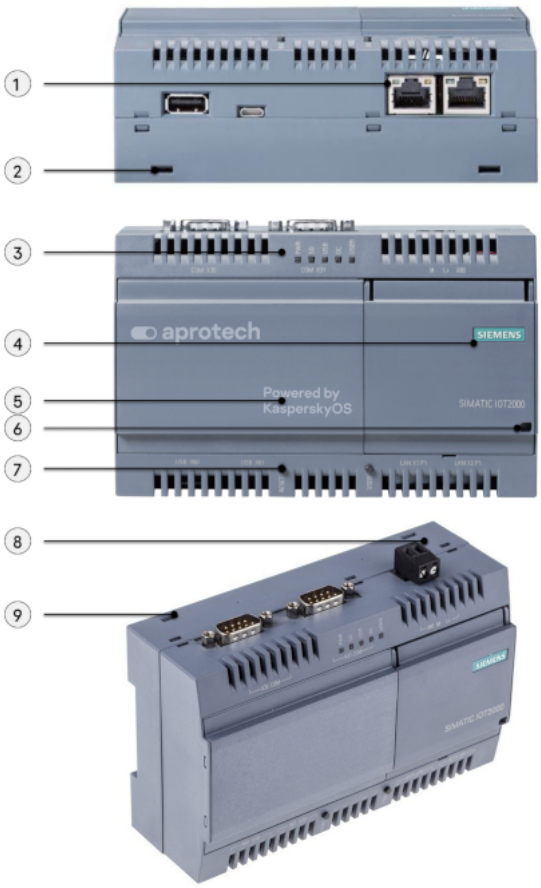
- Application for establishing a communication session through the serial port (for example, PuTTY, screen, or minicom).

  To run gateway performance diagnostics, you will need to connect the gateway to the computer using the USB-UART adapter and establish a communication session. This document provides a detailed analysis of gateway diagnostics using the PuTTY application, but you can use a different application (in which case you should refer to the documentation of that application for information about establishing a communication session). PuTTY or another similar application let you receive diagnostic information through the serial port to help identify the likely reason for gateway operating issues, such as a configuration error, issues with certificates, or lack of a response on the server side.

> Gateway diagnostics can also be performed using a computer running a Windows operating system and PuTTY application for Windows. However, you will not be able to use that same Windows computer to configure the gateway. The microSD card partitions containing the gateway configuration files are allocated in the ext3 file system, which is supported in the Linux operating system.

Hardware overview

This section contains a brief overview of the hardware components of Kaspersky IoT Secure Gateway 100 (see the figure below). For complete information on the employed hardware, please refer to the User Guide for the Siemens SIMATIC IOT2040 device.



Siemens SIMATIC IOT2040 device

The numbers on the diagram represent the following components:

1 – Ethernet 10/100 Mbps interfaces (two connectors).

2 and 9 – holes for mounting brackets on a wall (two holes on the upper edge of the device chassis and two on the lower edge).

3 – LED indicators (five light-emitting diodes).

4 – right-side cover.

5 – left-side cover.

6 – hole for sealing the left-side cover.

7 – restart button.

8 – power connector.

> The purpose of the USB and COM interfaces (RS232/422/485) and the **User** button are not examined in this document because these hardware components are not used in the software portion of Kaspersky IoT Secure Gateway 100 version 2.0.

What's new

Changes in Kaspersky IoT Secure Gateway 100 version 2.0:

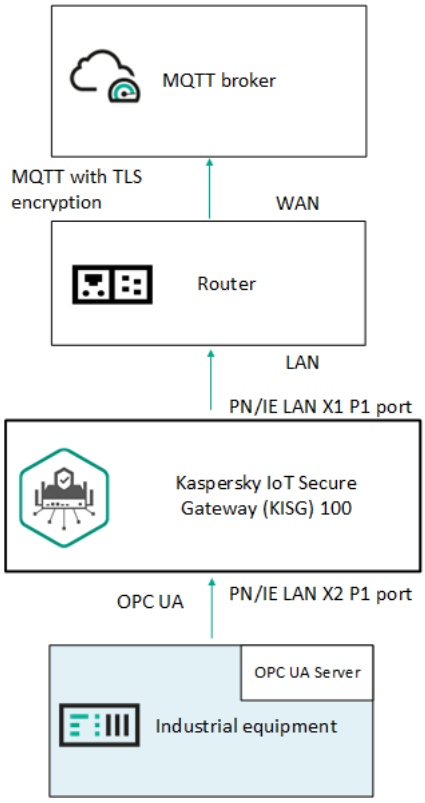
- You can now transfer data to an external network via the MQTT client using the TLS protocol. Version 2.0 no longer supports transferring data to the Siemens MindSphere cloud platform.

- The User Guide was revised and updated: it now contains more information on microSD memory card usage, on setting up connection over the MQTT protocol, on receiving diagnostic data through the UART connector, and on contacting Technical Support.

Standard deployment scenario

The standard deployment scenario for Kaspersky IoT Secure Gateway 100 assumes the following:

- Kaspersky IoT Secure Gateway 100 receives data from the OPC UA server over the OPC UA protocol.
- Kaspersky IoT Secure Gateway 100 receives data and forwards it to the MQTT broker over the MQTT protocol.



Kaspersky IoT Secure Gateway 100 deployment scenario

Security objectives and constraints

Security objectives

The security objectives of Kaspersky IoT Secure Gateway 100 include the following requirements:

- Kaspersky IoT Secure Gateway 100 ensures secure, unidirectional transfer of data from the gateway-connected OPC UA server to the MQTT broker over the MQTT protocol while eliminating the possibility of the external network having any impact on internal resources of the enterprise.
- Kaspersky IoT Secure Gateway 100 ensures the integrity and confidentiality of data transmitted to the MQTT broker.

Availability of Kaspersky IoT Secure Gateway 100 is not a security objective of Kaspersky IoT Secure Gateway 100.

Security constraints

The security constraints of Kaspersky IoT Secure Gateway 100 include the following limitations:

- The hardware platform is trusted, so threats associated with its vulnerabilities are not considered.
- No cybercriminal has physical access to the hardware platform, so threats associated with the corresponding vulnerabilities are not considered.
- The threat level from the external network is medium (basic elevated).
- The threat level from the internal network is low (basic).
- Kaspersky IoT Secure Gateway 100 does not have internal administration resources. The software portion and configuration are saved on an extractable microSD card that can be physically accessed only by the administrator.
- Kaspersky IoT Secure Gateway 100 cannot guarantee the integrity and confidentiality of data transmitted within the internal network from the gateway-connected OPC UA server to Kaspersky IoT Secure Gateway 100.
- Kaspersky IoT Secure Gateway 100 cannot ensure that connected devices will be protected against attacks launched from within the internal network.
- Kaspersky IoT Secure Gateway 100 is the only means to exchange data between the external network and internal network.
- The hardware platform has separate network controllers for connecting to the internal and external network.
- The MQTT broker supports connections over the TLS protocol.

For more detailed information on assessing the information security threat level, please refer to the website of the relevant government agency with jurisdiction over technical and export regulations.

Threats associated with breached availability of the infrastructure, such as inaccessible communication channels between the sides of network interaction, are not considered.

Configuring the gateway

Configuration of data transfer from the OPC UA server to the MQTT broker via Kaspersky IoT Secure Gateway 100 consists of the following steps:

1. **Generating an encryption key and certificates for connection over the OPC UA protocol**

   An encryption key and certificates must be generated if a secure OPC UA connection is used.

2. **Configuring nodes for data transfer over the OPC UA protocol on the OPC UA server**

   You can read about the OPC UA protocol specification on the developer's website. Kaspersky IoT Secure Gateway 100 version 2.0 supports OPC UA protocol version 1.04.

3. **Generating an encryption key and certificates for connection over the MQTT protocol**

   You will need to generate an encryption key and certificate for the MQTT client, and prepare a file containing the certificate chain that was used to sign the server certificate.

4. **Configuring the Kaspersky IoT Secure Gateway 100 settings for transferring data from the OPC UA server to the MQTT broker**

   Detailed instructions related to this step are provided in this Help section.

Extracting and inserting a microSD memory card

The microSD card is installed in the microSD slot on the Siemens SIMATIC IOT2040 device and contains the following files:

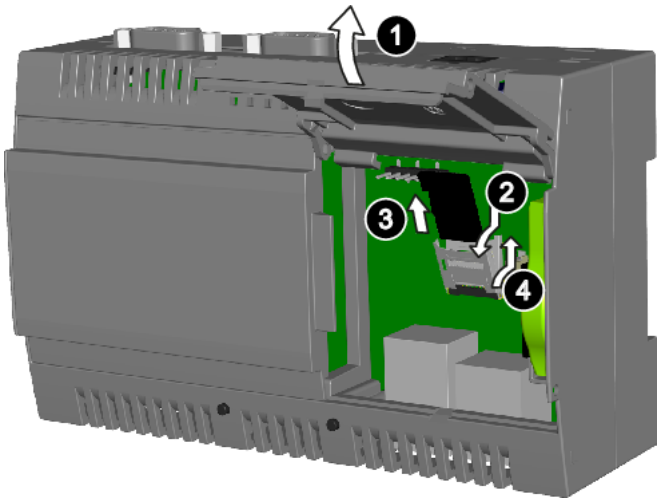
- configuration files with the settings of Kaspersky IoT Secure Gateway 100
- Kaspersky IoT Secure Gateway 100 health log files
- file with information about third-party code

To access these files, you need to extract the microSD card from the device and connect it to a computer that has a Linux operating system installed. To continue gateway operations, you should re-insert the microSD card into the device.

The Siemens SIMATIC IOT2040 device should be powered off before you extract and install the microSD memory card.

To extract and then re-insert the microSD memory card:

1. Open the right-side cover of the device.
2. Move the memory card slot tab downwards and open it.
3. Extract the microSD card.
4. Put the microSD card back, close the tab and move it upwards.



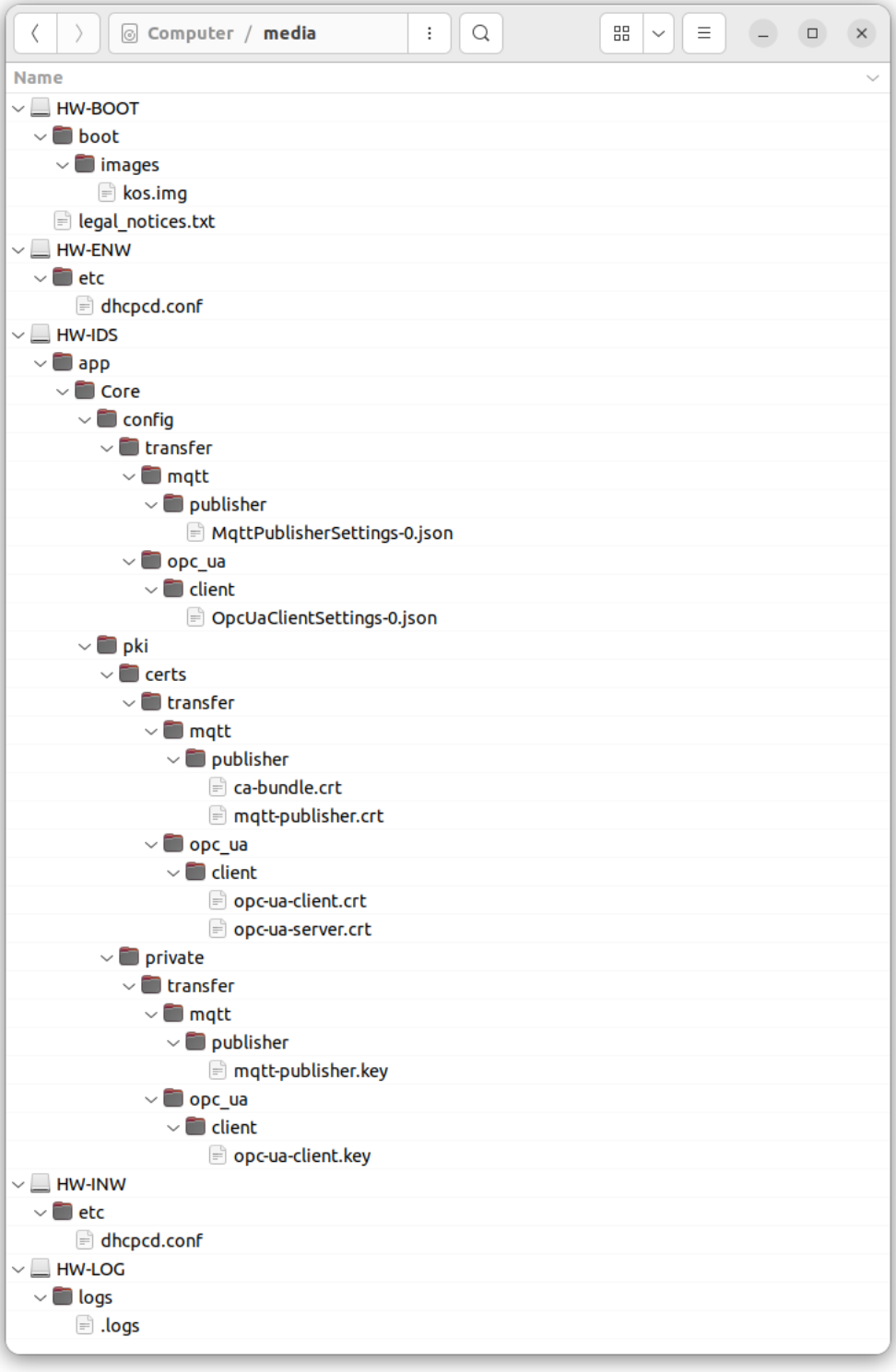
Extracting and inserting a microSD memory card

Location of files on the microSD memory card

To change the settings of Kaspersky IoT Secure Gateway 100, you will need to extract the microSD card from the microSD slot of the Siemens SIMATIC IOT2040 device, connect the microSD card to a computer running a Linux operating system, change the contents of the configuration files, and re-insert the microSD card into the device.

The microSD memory card includes five partitions (see the figure below):

- `HW-BOOT` – contains the boot image of KasperskyOS and a text file with information about third-party code.
- `HW-LOG` – contains the health log files and the `.log` file with the health log settings.
- `HW-IDS` – contains the following:
  - The `OpcUaClientSettings-0.json` file with the settings for receiving data over the OPC UA protocol.
  - The `MqttPublisherSettings-0.json` file with the settings for sending data over the MQTT protocol.
  - Files of certificates and encryption keys.
- `HW-INW` – contains the file with internal network settings.
- `HW-ENW` – contains the file with external network settings.



Location of files on the microSD memory card

Connecting to the network and configuring network settings

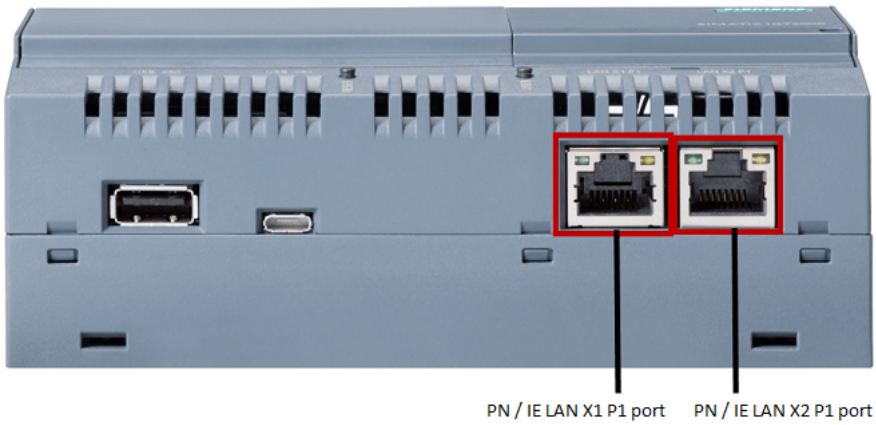
To connect a device to an external and internal network, you must establish physical connections (using an RJ45 cable) and configure the external and internal network settings if necessary.

Physical connection to the network

To connect Kaspersky IoT Secure Gateway 100 to a data transfer network:

1. Connect the network cable of the *internal* network (where the OPC UA server resides) to the PN/IE LAN X2 P1 port of the device.

2. Connect the network cable of the *external* network (where the MQTT broker resides) to the PN/IE LAN X1 P1 port of the device.



Layout of Ethernet ports on the Siemens SIMATIC IOT2040 device

Automatically obtaining network settings

By default, Kaspersky IoT Secure Gateway 100 is delivered with a dynamic (DHCP) configuration for the internal and external network. To automatically receive network settings in this configuration, there must be a configured DHCP server residing in the same network.

If there is no DHCP server or if you do not want to use dynamic configuration of the network, use a static configuration of the network.

Configuring static network settings

To configure static settings for the external and internal network:

1. Create two separate configuration files named `dhcpcd.conf` on the microSD card:

   - One for the external network in the directory `/etc` on the `HW-ENW` partition;
   - One for the internal network in the directory `/etc` on the `HW-INW` partition.

2. In the `dhcpcd.conf` files, define the network settings according to the dhcpd documentation.

   Sample configuration file defining the settings for an external or internal network:

   ```
   static ip_address=192.168.1.177/23
   static routers=192.168.1.1
   ```

3. Save the changes.

Configuring a connection over the OPC UA protocol

Kaspersky IoT Secure Gateway 100 receives data from the OPC UA server residing within the internal enterprise network over the OPC UA protocol, which is described by the OPC Unified Architecture specification. You can read about the OPC UA protocol specification on the developer's website. Kaspersky IoT Secure Gateway 100 supports OPC UA protocol version 1.04.

Securing a connection over the OPC UA protocol

To use a secure connection between the OPC UA client and the OPC UA server:

1. Copy the OPC UA client certificate to the directory `/app/Core/pki/certs/transfer/opc_ua/client` on the `HW-IDS` partition of the microSD card.
2. Copy the OPC UA server certificate to the directory `/app/Core/pki/certs/transfer/opc_ua/client` on the `HW-IDS` partition of the microSD card.
3. Copy the private encryption key of the OPC UA client to the directory `/app/Core/pki/private/transfer/opc_ua/client` on the `HW-IDS` partition of the microSD card.

Data can also be transmitted over the OPC UA protocol without using an encrypted connection. However, a secure connection between the OPC UA client and the OPC UA server cannot be ensured without encryption. It is not recommended to use an unsecured operating mode for Kaspersky IoT Secure Gateway 100.

Configuring data acquisition over the OPC UA protocol

To configure data acquisition using the OPC UA protocol:

1. Create the `OpcUaClientSettings-0.json` configuration file.
2. In the `OpcUaClientSettings-0.json` file, specify the OPC UA settings and their values in accordance with JSON syntax.
3. Put the configuration file into the directory `/app/Core/config/transfer/opc_ua/client` on the `HW-IDS` partition of the microSD card.

Sample configuration file:

OpcUaClientSettings-0.json

```
{
    "id": 0,
    "name": "Kaspersky IoT Secure Gateway 100 OPC UA Client",
    "description": "Collects data from CNC by Kaspersky IoT Secure Gateway 100",
    "url": "opc.tcp://192.168.177.7:4840",
    "readingCycle": 1,
    "security": {
        "mode": "SignAndEncrypt",
        "policy": "Basic256Sha256",
        "clientPkiData": {
            "certificate": "opc-ua-client.crt",
            "privateKey": "opc-ua-client.key"
        },
    "trustList": ["opc-ua-server.crt"]
    },
```

```
    "userCredentials":
    {
        "username": "KISG100",
        "password": "0R20jN#yZd~zaLKe?2J#@~|YC"
    },
        "heartbeat": {
        "id": 0,
        "name": "Heartbeat",
        "timeout": 60
    },
    "nodes": [
        {
            "id": 1,
            "name": "Temperature",
            "nodeId": "ns=1;s=VariableTemperature"
        },
        {
            "id": 2,
            "name": "Speed",
            "nodeId": "ns=2;i=2045"
        }
    ]
}
```

To edit files in JSON format, we recommend using a text editor that supports JSON syntax highlight. This will help avoid potential errors (for example, unbalanced braces).

Parameters description in the OpcUaClientSettings-0.json configuration file

The required parameters should be explicitly defined. The other parameters are optional. For optional parameters that are not included in the configuration file, the default value prescribed by the OPC UA protocol may be used.

Parameters in the OpcUaClientSettings-0.json file

Parameter name	Required parameter	Description	Possible values and notes
`id`	Yes	ID of the OPC UA client that receives data from the OPC UA server.	`0`. The value of this parameter must match the v of the `receivingHubId` parameter in the `GuideSettings-0.json` configuration file.
`name`	Yes	Name of the OPC UA client that receives data from the OPC UA server.	`<OPC UA client name>`. Example: `Kaspersky IoT Secure Gateway 100 OPC UA Client`.
`description`		Description of the OPC UA client that receives data from the OPC UA server.	`<OPC UA client description>`. Example: `Collect data from CNC by Kaspersky IoT Secure Gateway 100`.
`url`	Yes	OPC UA server address.	`<network protocol>://<host>:<port>`. Example: `opc.tcp://192.168.177.7:4840` Port 4840 is used by default.

`readingCycle`	No	Gateway data read frequency (in seconds).	`1`. Integer no less than `0`. `0` is a special value signifying the use of the maximum frequency available to the client an server.
`security`	Yes	OPC UA client security parameter block. Contains the `mode`, `policy`, and `trustList` parameters, and the `clientPkiData` parameter block.	• `{mode, policy, clientPkiData, trustList}` parameter block. • `null`. The `security` parameter block is used to configure the communication modes, securit policies, utilized certificates and encryption k for the digital signature and data encryptio If you do not need to configure the security parameters, enter `null` in the `security` blo The security mode will be set to `None` in this
`mode`	No	Mode for managing the security of the client application connection.	• `Sign` mode means that the connection requires a digital signature for data. • `SignAndEncrypt` mode means that the connection requires both a digital signatu and data encryption. • `None` mode means that the connection d not require a digital signature or data encryption. It is not recommended to use mode because it does not ensure a secur connection between the OPC UA client a the OPC UA server. • `Any` mode means that the connection will any of the listed modes that are supporte the server: `Sign`, `SignAndEncrypt`, `None`
`policy`	No	Name of the security policy used on the OPC UA server.	• `Basic128Rsa15` • `Basic256` • `Basic256Sha256` • `None` • `Any` policy means that any of the listed policies can be used (if supported by the server). The `None` security policy is the most compat with various types of data sources for an OP connection.
`clientPkiData`	No	Parameter block containing the certificate and private	`{certificate, privateKey}` parameter b

		encryption key of the OPC UA client for an encrypted connection.	The `clientPkiData` parameter block must b completed even if the `None` value is set for th `mode` and `policy` fields. For secure communication over OPC UA, you need to create a private encryption key and certificate and add them to the client and server configuration. When generating certificates for a connection between a clien (Kaspersky IoT Secure Gateway 100) and the UA server, make sure that the certificates co with the following requirements: • The settings of encryption keys and certificates are compliant with the selecte security policy. • DER or PEM format is used for certificate encryption keys of the client. • For the client certificate, the `Subject Al Name` field contains the value `URI:urn:aprotech:KISG100:OpcUaCl`
`certificate`	No	Certificate file name.	`opc-ua-client.crt`.
`privateKey`	No	Private encryption key file name.	`opc-ua-client.key`.
`trustList`	No	Array containing the names of trusted certificate files.	• `[opc-ua-server.crt]` – one or more na of trusted certificate files for the OPC UA server. • `AllowAll` – allows a connection to the O UA server without verifying its certificate. If the OPC UA server configuration prescribe use of a custom list of trusted certificates, ad the client certificate to this list. If certificate verification is not required, enter the `AllowA` value for this parameter.
`userCredentials`	No	Parameter block containing the account credentials of the OPC UA client on the OPC UA server.	• `{username, password}` parameter bloc containing the user account credentials. • `null` is indicated if you want to allow an anonymous connection of the OPC UA cli to the OPC UA server. In this case, you do need to provide values for the `username` `password`.
`username`	No	Name of the user account for authorization on the OPC UA server.	`<username>`.
`password`	No	Password of the user account for authorization on the OPC UA server.	`<password>`.

`heartbeat`	No	This parameter block is generated by the OPC UA client. It contains the parameters for the Kaspersky IoT Secure Gateway 100 heartbeat signal.	• `{id, name, timeout}` parameter block • `null`. If you do not add the `heartbeat` parameter you enter the `null` value, heartbeat signals w not be sent.
`id`	No	Data node ID.	`0`.
`name`	No	Data node name.	`<heartbeat node name>`. Example: `Heartbeat`.
`timeout`	No	Interval (in seconds) between the generation of heartbeat signals.	`60`. An integer no less than `0` must be entered. Tl default value is `30`.
`nodes`	Yes	Parameter block for data nodes.	`{id, name}` parameter block. Completed for each data node. The ID and na of data nodes are required for building routes transmitting data from the OPC UA server to MQTT broker.
`id`	Yes	ID of the outbound port.	`<id>`.
`name`	Yes	Name of the outbound port. This name must match the destination port `name` of one of the MQTT topics indicated in the `topics` parameter block in the `MqttPublisherSettings-0.json` configuration file.	`<node name>`. Example: `Temperature`. To correctly transfer data from the OPC UA server to the MQTT broker, you need to ma OPC UA data nodes to their corresponding MQTT topics. The `name` value is used for mapping.
`nodeId`	Yes	Data node ID.	`<namespace>, <nodeID>`.
`ns`	Yes	ID of the OPC UA server namespace.	`<namespace>`.
`nodeId`	Yes	ID of the data node in the OPC UA server namespace.	`<nodeID>`. Two types of IDs are possible: • `s` (string) – string value for the data node For example, `"nodeId": "ns=1;s=Vari temperature"`. • `i` (numeric) – numerical value for the data node ID. For example, `"nodeId": "ns=2;i=2045"`.

Special considerations when configuring OPC UA security settings

Kaspersky IoT Secure Gateway 100 does not establish a connection over the OPC UA protocol in the following cases:

- The server does not have a certificate, and an unsafe connection is not allowed.
- The `trustList` parameter lacks a defined server certificate, and the `AllowAll` value is not set.
- The client certificate, server certificate or encryption keys do not comply with the settings of the selected security policy.

The OPC UA server and client establish an unsafe connection in the following cases:

- The `null` value is set for the `security` and `userCredentials` settings blocks, and the server supports this type of connection.
- The `Any` value is set for the `mode` and `policy` fields, and the server offers the choice for an unsafe connection.

Any weakening of the security settings reduces the security of the connection. For example, the following settings reduce the security of a connection over the OPC UA protocol:

- Use of the `null` value for the `security` settings block will result in the use of a connection without encryption and without a signature.
- Use of the `AllowAll` value for the `trustList` field disables server certificate verification.
- Use of the `null` value for the `userCredentials` settings block disables the capability to connect to a server by using a username and password.
- The `Basic128Rsa15` and `Basic256` values for the `policy` field are considered to be obsolete in the OPC UA version 1.4 protocol specification because the SHA-1 hashing algorithm is no longer considered to be secure.
- Use of the `None` value for the `policy` or `mode` fields will result in the following:
  - use of a connection without encryption and without a data signature;
  - transmission of a plaintext password to the server.

Configuring a connection over the MQTT protocol

Kaspersky IoT Secure Gateway 100 forwards data to the MQTT broker over the MQTT protocol. You can read the MQTT protocol specification on the developer's website. Kaspersky IoT Secure Gateway 100 version 2.0 supports MQTT protocol version 3.1.1.

Securing a connection over the MQTT protocol

To securely transmit data over the MQTT protocol:

1. Copy the file containing the certificate chain of the MQTT client (a chain can consist of one leaf certificate of the client) to the directory `/app/Core/pki/certs/transfer/mqtt/publisher` on the `HW-IDS` partition of the microSD card.
2. Copy the file containing the certificate chain that was used to form the digital signature to the directory `/app/Core/pki/certs/transfer/mqtt/publisher` on the `HW-IDS` partition of the microSD card.

3. Copy the file of the private encryption key of the MQTT client to the directory `/app/Core/pki/private/transfer/mqtt/publisher` on the `HW-IDS` partition of the microSD card.

The certificates and encryption keys used by the MQTT client must be in PEM format. The key length of the client certificate must be at least 2048 bits.

Configuring data transmission over the MQTT protocol

To configure data transmission over the MQTT protocol:

1. Create the `MqttPublisherSettings-0.json` file.
2. In the `MqttPublisherSettings-0.json` file, specify the MQTT settings and their values in accordance with JSON syntax.
3. Put the configuration file into the directory `/app/Core/config/transfer/mqtt/publisher` on the `HW-IDS` partition of the microSD card.

Sample configuration file:

MqttPublisherSettings-0.json

```
{
    "id": 0,
    "name": "Kaspersky IoT Secure Gateway 100 MQTT Publisher",
    "description": "Transfer data to MQTT Broker by Kaspersky IoT Secure Gateway 100",
    "clientId": "KISG100",
    "serverUri": "ssl://192.168.188.8:8883",
    "security": {
        "clientPkiData": {
            "certificate": "mqtt-publisher.crt",
            "privateKey":  "mqtt-publisher.key"
        },
        "trustStore": ["ca-bundle.crt"]
    },
    "userCredentials":{
        "username": "KISG100",
        "password": "4vRXE84zsCy8bLQcZi2HH82TmC"
    },
    "lastWill": {
        "topicName": "LastWill",
        "message": "LastMessage"
    },
    "keepAlive": 800,
    "qualityOfService": 1,
    "topics": [
        {
            "id": 0,
            "name": "Heartbeat",
            "topicName": "Heartbeat"
        },
        {
            "id": 1,
            "name": "Temperature",
            "topicName": "Temperature"
        },
        {
```

```
            "id": 2,
            "name": "Speed",
            "topicName": "Speed"
        }
    ]
}
```

> To edit files in JSON format, we recommend using a text editor that supports JSON syntax highlight. This will help avoid potential errors (for example, unbalanced braces).

Parameters description in the MqttPublisherSettings-0.json configuration file

The required parameters should be explicitly defined. The other parameters are optional. For optional parameters that are not included in the configuration file, the default value prescribed by the MQTT protocol may be used.

Parameters in the MqttPublisherSettings-0.json file

Parameter name	Required parameter	Description	Possible values and notes
`id`	Yes	ID of the MQTT client that will send data to the MQTT broker.	`0`. The value of this parameter must match the value of the `sendingHubId` parameter in the `GuideSettings-0.json` configuration file.
`name`	Yes	Name of the MQTT client that will send data to the MQTT broker.	`<MQTT Publisher name>`. Example: `Kaspersky IoT Secure Gateway 100 MQTT Publisher`.
`description`	No	Description of the MQTT client that will send data to the MQTT broker.	`<MQTT Publisher description>`. Example: `Transfer data to MQTT Broker by Kaspersky IoT Secure Gateway 100`.
`clientId`	No	Unique identifier of the MQTT client.	`1`. The internal architecture of Kaspersky IoT Secure Gateway 100 involves interaction between a receiving hub and a sending hub. The `id` parameter is linked to the sending hub. The `clientId` parameter is linked to the MQTT protocol requirements and determines the ID of the MQTT client within this protocol. The `clientId` value must be unique among all clients connected to the MQTT broker.
`serverUri`	Yes	Address of the server that the MQTT client will connect to.	`<scheme>://<host>:<port>`. Example: `ssl://192.168.188.8:8883`.

			`ssl` is an architecture-prescribed scheme for querying a resource. `8883` is the default port.
`security`	Yes	Parameter block for configuring a secure connection. Contains the `clientPkiData` and `trustStore` parameter blocks.	`{clientPkiData, trustStore}` parameter block. The `security` parameter block is used to configure the certificates and encryption keys for the digital signature and data encryption.
`clientPkiData`	Yes	Parameter block containing the names of the certificate and private encryption key files of the MQTT client for an encrypted connection.	`{certificate, privateKey}` parameter block.
`certificate`	Yes	Name of the file containing the certificate chain to the MQTT client certificate.	`mqtt-publisher.crt`.
`privateKey`	Yes	Private encryption key file name.	`mqtt-publisher.key`. The key length must be at least 2048 bits.
`trustStore`	Yes	Array that includes the name of the file containing the certificate chain to the MQTT broker certificate.	`ca-bundle.crt`. Name of the file containing the certificate chain to the certificate of the Certification Authority that signed the MQTT broker certificate.
`userCredentials`	Yes	Parameter block responsible for authentication of the MQTT client on the server.	• `{username, password}` parameter block containing the user account credentials. • `null` is indicated if you want to allow an anonymous connection of the MQTT client to the MQTT broker. In this case, you do not need to fill in the `username` and `password` fields.
`username`	No	Name of the user account for authorization on the MQTT server.	`<username>`.
`password`	No	Password of the user account for authorization on the MQTT server.	`<password>`.
`lastWill`	No	Parameter block for configuring a message informing that the client was improperly disconnected (LWT message).	`{topicName, message}` parameter block.

			The client can specify the LWT message when connecting to the MQTT broker for the first time. The MQTT broker will store this message until it detects an improper disconnection of the client. Upon detection of an improper connection, it will send the LWT message to all clients that have subscribed to this type of message. The MQTT broker does not send this message when the client is properly disconnected.
`topicName`	No	Name of the MQTT topic that determines the information channel where the LWT message will be published.	`<topicName>`. Example: `LastWill`.
`message`	No	Contents of the LWT message.	`<message>`. Example: `LastMessage`.
`keepAlive`	No	Period of time that the MQTT broker can wait to receive a message from the MQTT client before terminating the connection due to inactivity.	`800`. The default value is `120`. Available values: `0`–`65535`. If the `keepAlive` value is equal to zero, the server will not be obligated to disconnect a client based on inactivity of the client. If the server deems a client to be inactive or if the client is not responding to queries, the server can disconnect the client at any time, irrespective of the `keepAlive` value provided by the client.
`qualityOfService`	No	Parameter defining a guarantee to send messages.	`1`. Agreement between the message sender (publisher) and message recipient (subscriber) that defines a guarantee to deliver a specific message. The MQTT specification defines the following three levels of `qualityOfService`: • `0` is no more than one time: the client publishes messages without verifying whether they are delivered to the broker. Messages may be lost or duplicated. • `1` is at least one time: the broker confirms delivery. Messages may be duplicated, but delivery is guaranteed. • `2` is exactly one time: message delivery is guaranteed, and any potential duplication is eliminated. The default value is `1`.
`topics`	Yes	Array from the parameter blocks of MQTT topics.	Array of `[{id, name, topicName}]` parameter blocks.

			A separate parameter block in the array is completed for each MQTT topic.
`id`	Yes	ID of the destination port.	`<id>`. Example: `0`.
`name`	Yes	Name of the destination port. This name must match the outbound port `name` of one of the OPC UA server data nodes indicated in the `nodes` parameter block in the `OpcUaClientSettings-0.json` configuration file.	`<name>`. Example: `Temperature`. To correctly transfer data from the OPC UA server to the MQTT broker, you need to map the MQTT topics to their corresponding OPC UA data nodes. The `name` value is used for mapping.
`topicName`	Yes	Name of the MQTT topic.	`<topicName>`. Example: `Heartbeat`. See also: Special considerations when generating names of MQTT topics.

Special considerations when generating names of MQTT topics

When filling in the values for `topicName`, please adhere to the following guidelines:

- Wildcard characters cannot be used in the names of MQTT topics. We also do not recommend using the `$` character in the names of MQTT topics.
- The name of an MQTT topic cannot be blank (it must contain at least one character).
- The names of MQTT topics are case sensitive.
- The names of MQTT topics can contain a blank space character.
- MQTT topics that are separated by only a `/` character at the beginning or end of the name are considered to be different MQTT topics.
- An MQTT topic name consisting of only a `/` character is permitted.
- The name of an MQTT topic must not contain the null character (NUL).
- Names of MQTT topics are UTF-8 strings of size that does not exceed 65535 bytes.

Required actions when an MQTT broker certificate is revoked

When an MQTT broker certificate is revoked, you will need to obtain a new certificate from the MQTT broker administrator and replace the revoked certificate. If you do not do this, Kaspersky IoT Secure Gateway 100 will trust both the revoked certificate and the new certificate until the revoked certificate expires. This could lead to a situation in which a connection established over a secure channel is not actually secure.

To use a new MQTT broker certificate instead of a revoked certificate:

1. In the `/app/Core/pki/certs/transfer/mqtt/publisher` directory on the `HW-IDS` partition of the microSD card, delete the file indicated in the `trustStore` parameter of the MqttPublisherSettings-0.json configuration file.

2. In the `trustStore` parameter of the `MqttPublisherSettings-0.json` configuration file, specify the name of the new certificate file.

3. Copy the new certificate file to the `/app/Core/pki/certs/transfer/mqtt/publisher` directory on the `HW-IDS` partition.

Diagnostics and contacting Technical Support

If you are unable to configure Kaspersky IoT Secure Gateway 100 and cannot find a resolution to your issue in the documentation, you can run diagnostics yourself and then contact Technical Support with the diagnostic information that you obtained.

The Kaspersky IoT Secure Gateway 100 diagnostics scenario consists of the following steps:

1. **Checking the connection to the MQTT broker**

   Perform the following checks on the MQTT broker side:

   - Check the gateway status (online or offline).
   - If the MQTT broker is maintaining an event log, check the diagnostic messages regarding the transmission of data from the gateway.
   - Check the regularity of MQTT messages.
   - If there is a limit on inbound traffic, make sure that this limit is not being exceeded.

2. **Inspecting the device**

   Conduct an inspection of hardware components. You will need to assess the following:

   - state of the LED indicators for PWR, SD, and OC;
   - proper connection of the power adapter and network cable connectors;
   - proper connection of Ethernet cables and the statuses of indicators for RJ45 connectors (PN/IE LAN X1 P1 and PN/IE LAN X2 P1);
   - absence of thermal or mechanical damage to the chassis, circuit board and electronic components.

3. **Searching for errors in the diagnostics log**

   Connect to the gateway via USB-UART adapter and use PuTTY to obtain a diagnostics log. In the obtained diagnostics log, check the following:

   - absence of errors at the hardware level;
   - startup of the KasperskyOS kernel and successful initialization of components;
   - startup of the OPC UA Client, MQTT Publisher and Navigation services;
   - connection to the OPC UA server based on the `OPC UA Client has established connection` message;
   - connection to the MQTT broker based on the `MQTT Publisher established connection` message;
   - receipt of messages over the OPC UA protocol and their transmission over the MQTT protocol.

   If you encounter errors establishing a secure connection, you should first verify that the date and time are set correctly. If the date and time are correctly set on the gateway but a certificate validation error occurs, you should check the format and validity dates of the certificates and make sure that the important fields have been completed correctly.

   To search for errors, you can also use the health log files stored on the microSD memory card.

4. **Contacting Technical Support**

   If you are not able to restore the correct operation of the gateway on your own, please contact Technical Support by emailing your query to support@aprotech.ru.

   Please attach the following to your query:

   - detailed description of the issue;
   - settings of network interfaces (`dhcpcd.conf` files);
   - OPC UA settings (`OpcUaClientSettings-0.json` file);
   - MQTT settings (`MqttPublisherSettings-0.json` file);
   - diagnostics log and/or health log files;
   - settings of the OPC UA server and MQTT broker.

Diagnostics using PuTTY

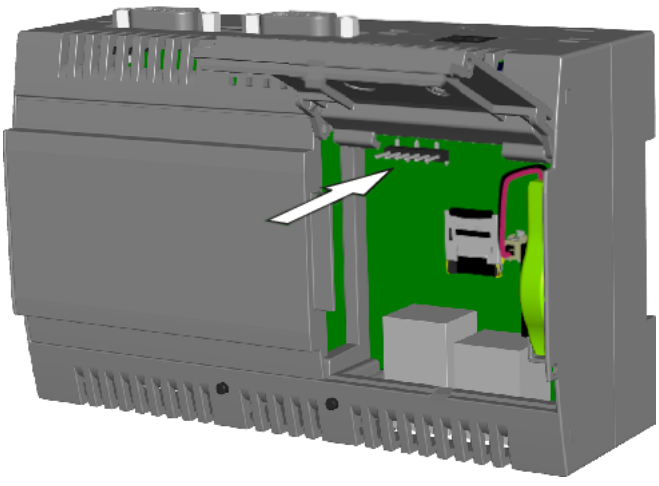
This section provides instructions on using the PuTTY application to obtain diagnostic information about gateway operation. PuTTY is a freely distributed client for remote access protocols, including SSH, Telnet and rlogin. PuTTY can also connect to devices via serial port.

For additional information on configuring and using PuTTY, please refer to the documentation on the PuTTY application.

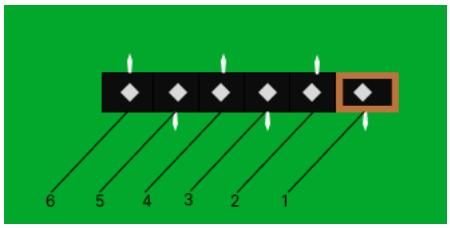
Connecting a computer to the gateway via USB-UART adapter

To use PuTTY for diagnostics of Kaspersky IoT Secure Gateway 100,

Use the USB-UART adapter available in the distribution kit to connect a computer to the UART diagnostic connector located under the right-side cover of the device (see the figure below).



Location of the UART diagnostic connector

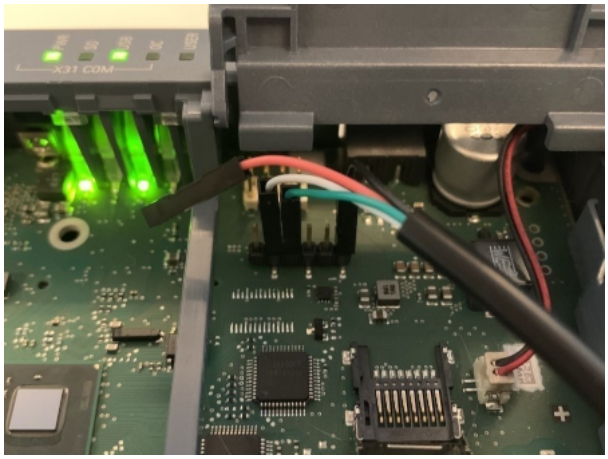


UART diagnostic connector contacts

When connecting the USB-UART adapter to the UART diagnostic connector, make sure you match the proper contacts of the diagnostic connector (see the table below).

Purpose of UART diagnostic connector contacts

Number of the contact	Purpose of the contact	USB-UART adapter wire
1	GND	Ground (black)
2	RTS_N	–
3	n. c.	–
4	TXD	Send data (green)
5	RXD	Receive data (white)
6	CTS_N	–



Example of connecting a USB-UART cable to a UART diagnostic connector

If the colors of the adapter wires differ from those presented in the table and in the figure, refer to the text label on the connectors or circuit board of the USB-UART adapter.

Configuring a connection to the gateway in PuTTY

To configure a connection in PuTTY:

1. Run the PuTTY application.

   The **PuTTY Configuration** dialog box is displayed at startup. The **Session** option is selected by default in the **Category** panel of the dialog box.

2. In the **Specify the destination you want to connect to** → **Connection type** settings block, select the **Serial** option.

3. In the **Serial line** field, specify the name of the serial port corresponding to the connected USB-UART adapter (for example, `/dev/ttyUSB0` in Linux or `COM8` in Windows).

   In Linux, you can identify the port name by running the `ls /dev/ttyUSB*` command. If this command returns multiple names, you can run it again with the USB-UART adapter connected and then disconnected. The name that appears when the adapter is connected will be the one you're looking for.

   In Windows, you can identify the port name by running Device Manager. In the console tree, open the **Ports (COM and LPT)** folder and find the object corresponding to the connected USB-UART adapter. The object name will contain the name of the port in parentheses. For example, **Prolific USB-to-Serial Comm Port (COM8)**.

4. In the **Speed** field, set the data transfer speed value at `115200` baud/sec.

5. In the **Category** panel, select **Session** → **Logging**.

6. In the **Session logging** settings group, select the **All session output** option.

7. In the **Log file name** field, specify the path to the event log file.

   You can specify the file name in the following format: `KISG100LOG_&Y_&M_&D_&T.log`. In this case, a new log file will be created for each new PuTTY connection session, and the date and time of the session start will be automatically added to the file name.

8. In the **Category** panel, select **Session**.

9. In the **Load, save or delete a stored session** settings block, in the **Saved Sessions** field, specify the name of the connection with the gateway (for example: `KISG100`).

10. Click **Save**.

Obtaining diagnostic information in PuTTY

To obtain a log file containing diagnostic information:

1. Connect a computer to the gateway via USB-UART adapter.

2. Run the PuTTY application.

3. In the **Load, save or delete a stored session** settings block, select a previously saved connection with the gateway (for example: `KISG100`).

4. Click **Load**.

   As a result, the previously saved values for the gateway connection settings will be loaded.

5. Click **Open**.

PuTTY will open a terminal window that displays the diagnostic information received from the gateway. The displayed information is also written to the event log file that was specified in the **Log file name** field when configuring the connection with the gateway.

The table below lists possible problems with PuTTY when receiving diagnostic information in Linux, and the ways to resolve these problems.

Potential problems with PuTTY when working in Linux

Error message	Solution
`Unable to open serial port`	Make sure that you specified the correct port name when configuring the connection with the gateway. If the port name was indicated correctly but the error persists, run PuTTY under the superuser at step 2: `sudo putty`
`unable to load font "server:fixed"`	After completing step 4, in the **Category** panel, select **Window** → **Fonts**. In the **Font used for ordinary text** field, replace `server:fixed` with one of the fonts available in the system.

Verifying information from the diagnostics log

This section provides instructions on performing diagnostics for Kaspersky IoT Secure Gateway 100 based on information from the diagnostics log file. To obtain the diagnostics log file, connect the gateway to a computer via the USB-UART adapter and use the PuTTY application.

Diagnostics of KasperskyOS startup

Open the diagnostics log file. Successful startup of KasperskyOS is indicated by the following messages:

```
Info: [BLKDEVSR] Port 'mmc0' successfully registered.
Info: [BLKDEVSR] Port 'mmc0_p0' successfully registered.
Info: [BLKDEVSR] Port 'mmc0_p0ro' successfully registered.
Info: [BLKDEVSR] Port 'mmc0_p1' successfully registered.
Info: [BLKDEVSR] Port 'mmc0_p1ro' successfully registered.
Info: [BLKDEVSR] Port 'mmc0_p2' successfully registered.
Info: [BLKDEVSR] Port 'mmc0_p2ro' successfully registered.
Info: [BLKDEVSR] Port 'mmc0_p3' successfully registered.
Info: [BLKDEVSR] Port 'mmc0_p3ro' successfully registered.
Info: [BLKDEVSR] Port 'mmc0_p4' successfully registered.
Info: [BLKDEVSR] Port 'mmc0_p4ro' successfully registered.
```

Diagnostics of OPC UA operation

The OPC UA client settings are assumed to be correct if the diagnostics log contains the following messages:

```
Info: Starting the OPC UA Client Manager Server service
Info: The OPC UA Client Manager Server service has been started
Info: Starting the OPC UA Client Manager service
Info: Loading all OpcUaClientSettings
Info: All OpcUaClientSettings has been loaded
Info: The OPC UA Client Manager service has been started
```

If these messages are absent from the event log file or are different from the messages listed above, the OPC UA client is likely experiencing operational problems.

Diagnostics of MQTT client operation

The MQTT client settings are assumed to be correct if the diagnostics log contains the following messages:

```
Info: Starting the MQTT Publisher Manager Server service
Info: The MQTT Publisher Manager Server service has been started
Info: Starting the MQTT Publisher Manager service
Info: Starting the MQTT Publisher service
Info: The MQTT Publisher service has been started
Info: The MQTT Publisher Manager service has been started
```

If these messages are absent from the event log file or are different from the messages listed above, the MQTT client is likely experiencing operational problems.

Diagnostics of the navigation (routing) service for transmitted data

The routing service was successfully started if the diagnostics log contains the following messages:

```
Info: Starting the Navigation Service service
Info: Loading all GuideSettings
Info: All GuideSettings has been loaded
Info: The Navigation Service service has been started
```

If these messages are absent from the event log file or are different from the messages listed above, the OPC UA client is likely experiencing operational problems.

Checking and configuring the date and time

You can check the date, time and time zone set on the hardware clock of the gateway based on the following message in the diagnostics log:

```
[2022-10-18 15:26:40.139 (UTC+0000)]
```

If the date on the gateway hardware clock is incorrect or exceeds the validity date of certificates, certificate verification will result in an error and the gateway will not be able to establish a secure connection.

If the date and time do not match the current exact time according to UTC, it is recommended to configure the date and time and set the time zone. Prior to changing the date, time or time zone, you should extract the microSD card from the device.

The time is configured in the terminal window of the PuTTY application.

To configure the time,

Run the command `time hh:mm:ss`, where `hh` refers to hours, `mm` refers to minutes, and `ss` refers to seconds.

To configure the time zone,

Run the command `timezone -s hh:mm`, where `hh:mm` refers to the UTC offset in hours and minutes.

To display the current time,

Run the command `time`.

To configure the date,

Run the command `date mm/dd/yyyy`, where `mm` refers to the month, `dd` refers to the day, `yyyy` refers to the year.

To display the current date,

Run the command `date`.

Health log

Kaspersky IoT Secure Gateway 100 lets you configure the settings of the system health log. System health logs are saved in the directory `/logs` located on the `TGW-HW-LOG` partition of the microSD card.

To configure the health log settings:

1. Use a text editor to open the `/logs/.log` file located on the `TGW-HW-LOG` partition of the microSD card.

   The `.log` file is hidden because the file name begins with a dot. If the file is not displayed in the file manager, enable the display of hidden files.

2. In the `LogFileSizeLimit` parameter value, enter the maximum size (in bytes) of one system health log file. For example, `LogFileSizeLimit=100000000`.

3. In the `DirectorySizeLimit` parameter value, enter the maximum size (in bytes) of the directory containing all system health log files. For example, `DirectorySizeLimit=1500000000`.

   Kaspersky IoT Secure Gateway 100 does not process disk space overflow events. For this reason, make sure that the size of the log storage directory does not exceed the available disk space of the partition.

4. Save the changes in the `.log` file.

Sample configuration file:

```
.log

LogFileSizeLimit=100000000
DirectorySizeLimit=1500000000
```

If the `.log` file is absent, the log will not be written.

Log files writing and rotating rules

Log files are written and rotated according to the following rules:

- If the size limit of one log file is exceeded during data write operations, a new file is created and the data is written to the new file.
- If a data string could not be completely written due to the size limit of one log file, a new file is created, the data string is written to this new file, and subsequent data will be written to the new file.
- If the size limits of one log file and all log files are exceeded when writing a data string, the old log file is deleted, a new file is created, and subsequent data will be written to this new file.
- If the limit for all log files is exceeded when writing a data string, the old file is deleted and the data string is written to the current file.
- Old log files are deleted in such a way to ensure that there is sufficient memory for writing a data string.
- A new file for log writing is created each time the gateway is loaded or restarted.

Information written to log files

The following information is recorded to log files:

- Initialization status of hardware components
- KasperskyOS loading status
- Initialization status of system components
- Initialization and operating status of:
  - network services
  - data repository
  - system logging service
  - components of applications:
    - OPC UA client and manager
    - MQTT client and manager
    - data transfer component
- Transmitted data elements
- Initialization and loading errors

Licensing

The application's terms of use of the are set forth in the End User License Agreement or in a similar document regulating usage of the application.

Data provision

Kaspersky IoT Secure Gateway 100 does not obtain, use, or process personal user data.

Known limitations

OPC UA limitations

Kaspersky IoT Secure Gateway 100 version 2.0 has the following OPC UA protocol support limitations:

- Kaspersky IoT Secure Gateway 100 must be restarted before it can apply new security settings for the OPC UA server after reconnection.
- The OPC UA client certificate does not undergo verification.
- To verify the server certificate, the OPC UA client checks the following:
  - The server certificate matches one of the certificates from the list of trusted certificates.
  - The server certificate is valid according to its validity term.
- If it is configured to trust all certificates via `"trustList": "AllowAll"`, the OPC UA client does not verify the server certificate.
- When the `None` security policy and mode are indicated in the Kaspersky IoT Secure Gateway 100 settings, the OPC UA client certificate and encryption key must also be provided.
- Only the following data types described in the OPC UA specification are supported:
  - Boolean
  - SByte
  - Byte
  - Int16
  - UInt16
  - Int32
  - UInt32
  - Int64
  - UInt64
  - Float
  - Double
  - String
  - DateTime
  - XmlElement
  - NodeId (only numeric and string)

    - ExpandedNodeId (only numeric and string)
    - StatusCode
    - QualifiedName
    - LocalizedText (partially)
    - Variant
- Double- and Float-type data received over the OPC UA protocol is rounded to the nearest six significant digits.
- To transmit data over OPC UA, the server must support the MonitoredItem and Subscription service sets.
- Only one OPC UA client connection to one OPC UA server is available.

MQTT limitations

Kaspersky IoT Secure Gateway 100 version 2.0 has the following MQTT protocol support limitations:

- Only one MQTT client connection to one MQTT broker is available.
- A delivery guarantee (`qualityOfService` parameter) is configured for all messages from the MQTT client.
- The MQTT client does not use the `retain` flag when sending messages nor for the LWT message (message informing that the client was improperly disconnected).
- Setting the `keepAlive` parameter of the MQTT client to `0` will not disable the "keep alive" mechanism (this mechanism disconnects a client that is inactive for too long).
- The MQTT client ignores the lack of response from the MQTT broker for a prolonged period of time and does not close the connection.
- If the connection is disrupted, a small number of published messages may be lost after the connection is restored, even if the buffer has sufficient free space.
- On the early stages of the MQTT client initialization, information from the client is not written to the health log of Kaspersky IoT Secure Gateway 100. This is due to the restrictions imposed by KasperskyOS security policies.

TLS limitations

Kaspersky IoT Secure Gateway 100 version 2.0 has the following TLS protocol support limitations:

- Only TLS protocol versions 1.2 or later are supported.
- Only the following TLS suites are supported:
    - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_AES_128_GCM_SHA256
- TLS_AES_128_CCM_SHA256

- Only the following digital signature algorithms are supported:
  - `ecdsa_secp521r1_sha512`
  - `ecdsa_secp384r1_sha384`
  - `ecdsa_secp256r1_sha256`
  - `ed25519`
  - `ed448`
  - `rsa_pss_pss_sha512`
  - `rsa_pss_rsae_sha512`
  - `rsa_pss_pss_sha384`
  - `rsa_pss_rsae_sha384`
  - `rsa_pss_pss_sha256`
  - `rsa_pss_rsae_sha256`
  - `rsa_pkcs1_sha384`
  - `rsa_pkcs1_sha512`
  - `rsa_pkcs1_sha256`

Other limitations

Other limitations of Kaspersky IoT Secure Gateway 100 version 2.0:

- Kaspersky IoT Secure Gateway 100 does not apply new network settings received from the DHCP server until the lease time of the IP address expires.
- In the Kaspersky IoT Secure Gateway 100 health log, the name `en0` is used for both network interfaces.

- Messages regarding a lost connection on the internal or external network interface are not written to the Kaspersky IoT Secure Gateway 100 health log immediately after disconnection. This type of message may appear in the health log if a connection is absent for a prolonged period of time, or it may never appear at all.

- If powered off, Kaspersky IoT Secure Gateway 100 does not retain any unsent data because it does not store this data in non-volatile memory.

- Kaspersky IoT Secure Gateway 100 does not process disk space overflow events when maintaining the health log. When configuring the health log settings, make sure that the size of the log storage folder does not exceed the available disk space on the `TGW-HW-LOG` partition of the microSD card.

Other sources of information

During installation, configuration, and use of Kaspersky IoT Secure Gateway 100, you can refer to the following additional documents:

- Technical specifications of the Siemens Simatic IOT2040 device
- Selection of technical documentation for the Siemens Simatic IOT2040 device
- Documentation for the dhcpcd client
- OPC UA protocol specification
- MQTT protocol specification

Glossary

Availability

State of information (or resources of an information system) whereby persons with the appropriate access rights have unobstructed access to the specific information or resource.

Certificate

Data structure with a digital signature containing a public encryption key and the ID of the client or server.

Certificate chain

Combination of any number of intermediate certificates between the end-entity certificate and the root certificate.

Client

Participant of client-server interaction that sends requests to the server and receives responses to those requests.

Confidentiality

Property of information meaning that it cannot be accessed by unauthorized users and/or processes.

Cyberimmune information system

A system that guarantees the fulfillment of specific security objectives in all possible scenarios of system usage as stipulated by the developers.

Data node

Structural element of an OPC UA information model containing data and metadata.

Data source

Standalone data source for exchanging messages between devices on the internet of things. For example, a data source could be an OPC UA server at the management controller of an industrial machine.

Digital signature

A value calculated with an encryption algorithm and added to data in such a way that any data recipient can use the signature to verify the origin and integrity of the data.

Encryption

Conversion of data from readable format to encoded format. Encrypted data can be read or processed only after decryption.

Encryption key

Component of a pair of encryption keys used for asymmetric encryption. Keys can be public or private.

End-entity certificate

Certificate containing a public encryption key that can be used to verify or validate an end-entity, such as an MQTT client.

Integrity

State of information (or resources of an information system) whereby changes can be made only by persons who have the permissions to make such a change.

Internet of things (IoT)

A network of interrelated electronic devices ("things") that are equipped with built-in capabilities for interaction with the external environment or with each other without human involvement.

Internet of things (IoT) Secure Gateway

A system that ensures secure transmission of user traffic between sensors and an IoT platform.

Kaspersky IoT Secure Gateway 100

A software/hardware system based on the Siemens SIMATIC IOT2040 device for the industrial internet of things that has the KasperskyOS operating system and application software installed. Kaspersky IoT Secure Gateway 100 is designed to work as a secure gateway for the industrial internet of things.

KasperskyOS

A microkernel operating system for building secure solutions.

Message Queuing Telemetry Transport (MQTT)

A network protocol that works on top of the TCP/IP protocol stack to exchange messages between devices on the internet of things.

MQTT broker

A server that receives, filters, and forwards messages over the MQTT protocol.

MQTT topic

A hierarchical path to the data source used for sending messages over the MQTT protocol.

OPC UA client security settings

All security-related settings of Kaspersky IoT Secure Gateway 100. You can make changes to these settings by editing the configuration files on the microSD card.

OPC UA security policies

Set of mechanisms and characteristics, including signature and encryption algorithms and encryption key algorithm that can ensure the security of the connection between the OPC UA server and client.

Open Platform Communications Unified Architecture (OPC UA)

Specification defining the protocols and mechanism for data transfer in industrial networks as well as interaction between devices in these networks.

Root certificate

Certificate of the root Certification Authority.

Root Certification Authority

Top Certification Authority that is not subordinate to any higher Certification Authority.

Security constraints

Additional restrictions imposed on the system operating conditions that either simplify or complicate the fulfillment of security objectives.

Security mode

Mode in which the client and server agree on whether or not to use encryption and a digital signature for data transmission.

Security objectives

Requirements imposed on a cyberimmune information system that must be fulfilled to ensure that the system operates securely in any possible usage scenario with consideration of the necessary security constraints.

Server

Participant of client-server interaction that processes requests from the client.

TLS

Secure protocol that uses encryption to transfer data in local networks and on the internet. TLS is used in web applications to create secure connections between a client and a server.

Information about third-party code

Information about third-party code is contained in the file named legal_notices.txt, which is located on the microSD memory card in the root directory of the `HW-BOOT` partition.

Trademark notices

Registered trademarks and service marks are the property of their respective owners.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Windows is a trademark of the Microsoft group of companies.

Siemens, MindSphere, and Simatic are registered trademarks of Siemens AG.