MAKE THE
WORLD SEE

Milestone Systems

XProtect® Webhooks 2025 R1

Administrator manual

milestone

Contents

Copyright, trademarks, and disclaimer

Trademarks

Disclaimer

This text is intended for general information purposes only, and due care has been taken in its preparation.

Any risk arising from the use of this information rests with the recipient, and nothing herein should be construed as constituting any kind of warranty.

Milestone Systems A/S reserves the right to make adjustments without prior notification.

All names of people and organizations used in the examples in this text are fictitious. Any resemblance to any actual organization or person, living or dead, is purely coincidental and unintended.

This product may make use of third-party software for which specific terms and conditions may apply. When that is the case, you can find more information in the file `3rd_party_software_terms_and_conditions.txt` located in your Milestone system installation folder.

Overview

Introduction

This manual describes the webhooks feature.

Webhooks are HTTP requests that enable web applications to communicate with each other and facilitates the sending of real-time data from one application to another when a predefined event occurs, for example sending event data to a predefined webhook endpoint when a user logs on to the system or when a camera reports an error.

A webhook endpoint (webhook URL) is the predefined address which the event data is to be sent to, much like a one-way telephone number.

You can use webhooks to build integrations which subscribe to selected events in XProtect. When an event is triggered, an HTTP POST is sent to the webhook endpoint you have defined for that event. The HTTP POST body contains event data in JSON.

Webhooks do not poll the system for data or triggered events, instead, the system pushes event data to the webhook endpoint when an event occurs which makes webhooks less resource-demanding and faster to set up compared to polling-solutions.

Webhooks can be set up to integrate with or without using code scripts.

You should verify that the event data sent from XProtect complies with the existing data and privacy protection legislation of your country.

Installation

Webhooks installation

The Webhooks functionality is by default installed and ready to use on XProtect 2023R1 or later and displays the **Webhooks** action on the **Rules** tab in Management Client.

Configuration

Creating webhooks in XProtect

To create and configure a webhook, you should:

1. Create the webhook URL.
2. Create a webhook in Management Client.
3. Create a rule to trigger the webhook in Management Client.
4. Test the webhook rule and actions from XProtect (Optional).

1. Create the webhook URL

Generate or obtain the URL of the webserver or application you want to send event data to.

The URL must be static and secure as the URL will be registered in XProtect and used repeatedly to receive event data from XProtect.

If the URL of the webserver is updated, you must update the webhook URL in XProtect.

> ⚠ You can use http:// instead of https://, but using http:// through non-secure networks may expose event data in plain text. Using the more secure https:// is strongly recommended.

2. Create a webhook in Management Client

1. In **Management Client** > **Rules and Events** > **Webhooks**, right-click **Webhooks** and select **Add New...**
2. In **Webhook Information**:
    a. **Name** field: Enter a unique name for the webhook.
    b. **Address** field: Enter the URL for the webhook starting with https://. You can use http:// instead of https://, but using http:// through non-secure networks may expose event data in plain text. Using the more secure https:// is strongly recommended.
    c. **Token (optional)** field: Enter a token if you want to sign the webhooks requests.
3. Click **Save** in the toolbar to save the new webhook.

3. Create a rule to trigger the webhook

1. In **Management Client** > **Rules and Events** > **Rules**, right-click **Rules** and select **Add Rule...** to start the **Manage Rule wizard** at **Step 1: Type of rule**.
2. In **Step 1: Type of rule**, specify a name and a description of the new rule in the **Name** and **Description** fields respectively.
3. In the **Select the rule type you want you to create** pane, select **Perform an action on <event>**.
4. In the **Edit the rule description** pane, click **event** and select an event.
5. Click **OK** to create the event.
6. Click **Next** to go to **Step 2: Conditions**.
7. In **Step 2: Conditions**, apply any conditions relevant to the event you are creating.
8. Click **Next** to go to **Step 3: Actions**.
9. In **Step 3: Actions**, scroll down and select **Send event info to <webhooks>**.
10. Click **address** on the **Edit the rule description** pane.
11. Add a predefined webhook and click **OK**.
12. Click **Next** to go to the wizard's third step.
13. In **Step 4: Stop criteria**, click **Finish** to create the rule.

When the event you have created is triggered, an HTTP POST will be sent to the URL you configured in the **Address** field of the webhook specified for the event.

The payload (body) of the HTTP POST will contain event data from the event that triggered the rule.

Depending on your requirements, you may also have to set up a procedure on the webhook receiver to process the HTTP POST requests and react on the received event data.

4. Test the webhook from XProtect (Optional)

You should test the event to verify that the event triggers and performs the actions you define before utilizing the event and webhook in daily operations.

1. Open **Management Client** > **Rules and Events** and select the event you want to test.
2. Click **Test Event** to test the event.
3. Verify that the actions defined for the event have been successfully performed, for example an email or SMS has been sent, scripts have been run, alarms have been updated, etc.

Integrations

Webhook integrations

Webhooks can be set up to integrate with or without using code scripts.

No-code integrations

You can use webhooks to integrate with 3rd party no-code web applications such as Zapier, IFTTT, or Automate.io.

No-code applications help you to integrate with applications that do not themselves contain webhook integrations and help you share event data between them, with the application functioning as the facilitator of the data exchange.

For example, when an event is triggered, webhooks can be set up to:

- Send an SMS message to one or more recipients
- Send a message in Microsoft Teams to one or more recipients
- Flash a control light in a control room
- Open a support ticket
- Add a row to a spreadsheet
- and many more...

Code integrations

You can also use webhooks to execute specific scripts on an endpoint of a web server when the event data is received.

For example, you can use webhooks to do the following when the event is triggered:

- Run Python code in a Flask server
- Run Node code in a Next.js server
- Trigger an Amazon Lambda function
- and many more...

Integrations - examples

No-code webhook integrations

The following is an example of a webhook integration using Low-code or No-code on the receiving server.

The IF This Then That (IFTTT) web application is used as an example here, but the general guidelines can also be applied to similar web applications such as Zapier.

When integrating 3rd party No-code web applications with webhooks, the following actions generally apply:

- In the 3rd party application, set up the actions to be performed based on the event that will trigger the webhook.
- In the 3rd party application, copy the URL as this will be used to set up the webhook in XProtect.
- In XProtect Management Client create and set up the webhook, using the URL from the 3rd party application.
- In XProtect Management Client, create the rule that will trigger the webhook for a given event.

Remember to test your event before deploying your changes.

Example - Send an email with IFTTT when an event is triggered

This example contains several procedures that illustrate how you can use the IF This Then That (IFTTT) web application to send an email message to a Gmail account when an event is triggered in your XProtect system.

1. Connect a webhook in IFTTT

First, create the IFTTT Applet that will send the Gmail message for the event and save the resulting URL.

You must create an IFTTT user in order to utilize the features of the IFTTT website.

1. Go to the IFTTT website and log in with your IFTTT credentials.
2. Create a new Applet: https://ifttt.com/create
3. Set up the **If This** part:
    a. Click **If This**
    b. Search for **Webhooks**
    c. Click the blue **Webhooks** tile
    d. Click **Receive a web request with a JSON payload**
    e. Click **Connect**
    f. Write **xprotect_test** in **Event** Name
    g. Click **Create trigger**
4. Set up the **Then That** part:
    a. Click **Then That**
    b. Search **Gmail**
    c. Click the blue **Gmail** tile
    d. Click **Send yourself an email**
    e. Click **Connect**
    f. Click **Create action**
5. Click **Continue**
6. Click **Finish**
7. Go to https://ifttt.com/maker_webhooks
8. Click **Documentation**
9. Identify your URL. The URL should be something similar to: https://maker.ifttt.com/trigger/**{event}**/json/with/key/lPY4YYsvGSfTSJBM54MrDDSOGWTmk6z9VSD113hxoDY
10. Replace **{event}** with **xprotect_test**. The URL should end up similar to: https://maker.ifttt.com/trigger/**xprotect_test**/json/with/key/lPY4YYsvGSfTSJBM54MrDDSOGWTmk6z9VSD113hxoDY
11. Copy and save this address. The address should not be shared with unauthorized users.

2. Create a webhook in XProtect

After you have set up the applet in IFTTT, you must create the webhook in XProtect, using the URL you created in steps 9-11 above when setting up the IFTTT applet.

1. In **Management Client** > **Rules and Events** > **Webhooks**, right-click **Webhooks** and select **Add New...**
2. In **Webhook Information**:
    a. **Name** field: Enter "IFTTT Gmail"
    b. **Address** field: Set the address to the one from step 11 of the previous section. It should be something similar to: https://maker.ifttt.com/trigger/xprotect_test/json/with/key/lPY4YYsvGSfTSJBM54MrDDSOGWTmk6z9VSD113hxoDY
3. Click **Save** in the toolbar to save the new webhook.

3. Trigger the webhook with a rule in XProtect

After you have created and set up a webhook, you must create and set up an event to trigger the webhook.

1. In **Management Client** > **Rules and Events** > **Rules**, right-click **Rules** and select **Add Rule...** to start the **Manage Rule** wizard at **Step 1: Type of rule**.
2. In **Step 1: Type of rule** > **Name** field, enter **Send Gmail** and add an optional description of the rule in the **Description** field.
3. In the **Select the rule type you want you create** pane, select **Perform an action on <event>**
4. In the **Edit the rule description** pane, click **event** and in **Events** > **External Events** > **User-defined Events**, select **Event High**.
5. Click **OK** to create the event.
6. Click **Next** to go to **Step 2: Conditions**.
7. In **Step 2: Conditions**, apply any conditions relevant to the event you are creating.
8. Click **Next** to go to **Step 3: Actions**.
9. In **Step 3: Actions**, scroll down and select **Send event info to <Webhook>**.
10. Click **address** on the **Edit the rule description** pane
11. Add the **IFTTT Gmail** webhook and click **OK**.
12. Click **Next** to go to **Step 4: Stop criteria**.
13. In **Step 4: Stop criteria**, click **Finish** to create the rule.

4. Test the webhook from XProtect (Optional)

You should test the event to verify that the event triggers and sends the email as it should before utilizing the event and webhook in daily operations.

1. Open **Management Client** > **Rules and Events** and click **User-defined Events**.
2. Select **Event High** and click **Test Event** to test the event.
3. Open the inbox of the Gmail account you specified in the IFTTT web application above to verify you have received the correct email with the event you triggered.

5. Changing the email content (Optional)

You can change the content of the existing email message sent through the IFTTT web application.

1. Go to the IFTTT website and log in with your IFTTT credentials.
2. Go to **My Applets - IFTTT**
3. Click the **If Maker Event 'xprotect_test'**, then **...Applet**
4. Click **Settings**
5. Click **Then**
6. Edit the text of the email subject or email body, for example rename the subject to **Event High from XProtect**
7. Click **Update action** and then click **Update**.

Code webhook integrations

The following are examples of webhook integrations using code on the receiving server.

When integrating 3rd party server with webhooks, the following actions generally apply:

- Start a web server with en endpoint listening for webhooks.
- Locate and copy the address to the receiving server as this will be used to set up the webhook in XProtect.
- In XProtect Management Client, create and set up the webhook, using the address of the receiving server.
- In XProtect Management Client, create the rule that will trigger the webhook for a given event.

Remember to test your event before deploying your changes.

Example: Receiving events from XProtect in Node.js (with Express)

This example illustrates how to receive events from XProtect in a Node.js Express server using webhooks.

1. Create and start a Node.js Express server

First you must create and start a web server that listens for incoming events.

1. Download and install Node.js from https://nodejs.org.
2. Create a folder anywhere in your file system and name the folder **webhooks**.
3. Open a terminal and navigate to the **webhooks** folder you just created.
4. Run **npm init** and use the default values.

5. Run **npm install express**.
6. Inside the **webhooks** folder, create a file and name the file **index.js**.
7. Open the **index.js** file in a text editor and paste the sample code below into the **index.js** file.

```
const express = require('express');

const crypto = require('crypto');


// change SECRET_TOKEN for your own token

// never hard code the token in the code

// we do it only for demonstration purposes

// instead, store it as an environment variable

const SECRET_TOKEN = "32212c72863c01a931609c5ebfe1abc5";
```

```
const isSignatureValid = (body, headerSignature, secretToken) => {

    const digest = crypto.createHmac("sha256", secretToken).update(body).digest();

    const expectedSignature = `sha256=${digest.toString('base64').toString('utf-8')}`;

    if (expectedSignature.length !== headerSignature?.length) return false;

    return crypto.timingSafeEqual(Buffer.from(headerSignature), Buffer.from
(expectedSignature));

}

```

```
const app = express();

app.use(express.json());

```

```
app.post('/webhooks', (req, res) => {
    if (!isSignatureValid(
            JSON.stringify(req.body),
            req.headers['x-hub-signature-256'],
            SECRET_TOKEN)){
        console.log('Received event with invalid signature');
        return res.status(403).end();
    }
    console.log('Received event from XProtect through Webhook:');
    console.log(req.body);
    res.send('');
})
```

```
app.listen(5000, () => {

  console.log(`Server started...`);

})
```

8. In the terminal, run **node -p "require('crypto').randomBytes(64).toString('hex');"** to generate your own secret token.
9. Copy the token string and paste it in line 8 of **index.js** replacing the example token.
10. Save the **index.js** file
11. Use the terminal to run **node index.js**. The log message **Server started** should be displayed.

2. Create a webhook in XProtect

After you have started the server, you must create the webhook in XProtect.

1. In **Management Client** > **Rules and Events** > **Webhooks**, right-click **Webhooks** and select **Add New...**
    a. In **Webhook Information**:
    b. **Name** field: Enter **Node integration**
    c. **Token (optional)** field: Enter the token string you copied into the **index.js** file.
    d. **Address** field: Set the address of the receiving server.

    If you created the Node.js server on the same machine as the Event Server, enter **http://127.0.0.1:5000/webhooks**

    If you created the Node.js server on a server with a public IP address, enter **https://<IP>:5000/webhooks** where **<IP>** is the IP address of the server with a public address.

    For Node.js servers on machines without public addresses, see the section below.

    You can use http:// instead of https://, but using http:// through non-secure networks may expose event data in plain text. Using the more secure https:// is strongly recommended.
2. Click **Save** in the toolbar to save the new webhook.

For Node.js servers without public addresses

If you created the Node.js server on a machine on a different network and without a public IP address, for example on a development or testing machine, you can use NGrok for testing purposes.

1. Install https://ngrok.com/ in the same machine you installed the Node.js server on.
2. On your machine, locate and run **ngrok http 5000**
3. Copy the generated public address and insert the address in the **Address** field above. The NGrok public address should be something similar to: https://0c60-12-212-221-50.eu.ngrok.io

3. Trigger the webhook with a rule in XProtect

After you have created and set up a webhook, you must create and set up an event to trigger the webhook.

1. In **Management Client** > **Rules and Events** > **Rules**, right-click **Rules** and select **Add Rule...** to start the **Manage Rule** wizard at **Step 1: Type of rule**.
2. In **Step 1: Type of rule** > **Name** field, enter **Send Event High to Node** and add an optional description of the rule in the **Description** field.
3. In the **Select the rule type you want you create** pane, select **Perform an action on <event>**
4. In the **Edit the rule description** pane, click **event** and in **Events** > **External Events** > **User-defined Events**, select **Event High**.
5. Click **OK** to create the event.
6. Click **Next** to go to **Step 2: Conditions**.
7. In **Step 2: Conditions**, apply any conditions relevant to the event you are creating.
8. Click **Next** to go to **Step 3: Actions**.
9. In **Step 3: Actions**, scroll down and select **Send event info to <Webhook>**.
10. Click **webhook** on the **Edit the rule description** pane
11. Add the **Node integration** webhook and click **OK**.
12. Click **Next** to go to **Step 4: Stop criteria**.
13. In **Step 4: Stop criteria**, click **Finish** to create the rule.

4. Test the webhook from XProtect (Optional)

You should test the event to verify that the event triggers and is recieved in the Express server before utilizing the event and webhook in daily operations.

1. Open **Management Client** > **Rules and Events** and click **User-defined Events**.
2. Select **Event High** and click **Test Event** to test the event.
3. Open the console that is running the server. The **Received event from XProtect through Webhook** log message should be displayed.

5. Troubleshooting (Optional)

If you don't receive the events in your web server, open the MIP Logs from the Event Server tray icon to troubleshoot any potential errors.

> The log of the error can take up to 90 seconds to appear because of the retry policy.

Example: Receiving events from XProtect in Python (with Flask)

This example illustrates how to receive events from XProtect on a Python Flask server using webhooks.

1. Create and start a Python Flask server

First you must create and start a web server that listens for incoming events.

1. Download and install the latest version of Python from https://www.python.org/.
2. Create a folder anywhere in your file system and name the folder **webhooks**
3. Open a terminal and navigate to the **webhooks** folder you just created.
4. Create a virtual environment inside the folder by using the command **python -m venv venv**
5. Activate the virtual environment by using the command **\venv\Scripts\activate**
6. Install the Flask server by using the command **pip install flask**
7. Inside the **webhooks** folder, create a file and name the file **main.py**.
8. Open the **main.py** file in a text editor and paste the sample code below into the **main.py** file.

```

```

```
from flask import Flask, request
```

```
import hashlib

import hmac

import base64



# change SECRET_TOKEN for your own token

# never hard code the token in the code

# we do it only for demonstration purposes

# instead, store it as an environment variable

SECRET_TOKEN = bytes("383b9d27c4a892626881d73b0f70c5b62b213ab89d33c8788ac85bd750dbdf59",
'utf-8')



def is_signature_valid(body: bytes, header_signature: str, secret_token: bytes) -> bool:
```

```
    digest: bytes = hmac.HMAC(key=secret_token, msg=body, digestmod=hashlib.sha256).digest()

    expected_signature: str = f"sha256={base64.b64encode(digest).decode('utf-8')}"

    return hmac.compare_digest(header_signature, expected_signature)




app = Flask(__name__)




@app.route('/webhooks', methods=['POST'])

def webhooks():

    body = request.data

    header_signature = request.headers.get('X-Hub-Signature-256', '')

    if not is_signature_valid(body, header_signature, SECRET_TOKEN):
```

```
        print('Received event with invalid signature')

        return '', 403


    print("Received event from XProtect through Webhook:")

    print(request.json)

    return '', 200



app.run(host='127.0.0.1', port=5000)
```

9. In the Python interpreter, run the **import secrets; print(secrets.token_hex(32))** command to generate your own secret token string.
10. Copy and paste the token string in line 10 of the **main.py** script file, replacing the dummy token sting of the sample code.
11. Save your changes to the **main.py** script file.
12. Use the terminal to run the **python main.py** command. The *** Serving Flask app 'main'** log message should be displayed.

2. Create a webhook in XProtect

After you have started the server, you must create the webhook in XProtect.

1. In **Management Client** > **Rules and Events** > **Webhooks**, right-click **Webhooks** and select **Add New...**
    a. In **Webhook Information**:
    b. **Name** field: Enter **Python integration**
    c. **Token (optional)** field: Enter the token string you copied into the **main.py** file.
    d. **Address** field: Set the address of the receiving server.

    If you installed Python on the same machine where the Event Server, enter **http://127.0.0.1:5000/webhooks**

    If you installed Python on a server with a public IP address, enter **https://<IP>:5000/webhooks** where **<IP>** is the IP address of the server with a public address.

    For Python installed on machines without public addresses, see the section below.

    You can use http:// instead of https://, but using http:// through non-secure networks may expose event data in plain text. Using the more secure https:// is strongly recommended..
2. Click **Save** in the toolbar to save the new webhook.

For Python on machines without public addresses

If you installed Python on a machine on a different network and without a public IP address, for example on a development or testing machine, you can use NGrok for testing purposes.

1. Install https://ngrok.com/ on the same machine you installed Python on.
2. On your machine, locate and run **ngrok http 5000**
3. Copy the generated public address and insert the address in the **Address** field above. The NGrok public address should be something similar to: https://0c60-12-212-221-50.eu.ngrok.io

3. Trigger the webhook with a rule in XProtect

After you have created and set up a webhook, you must create and set up an event to trigger the webhook.

1. In **Management Client** > **Rules and Events** > **Rules**, right-click **Rules** and select **Add Rule...** to start the **Manage Rule** wizard at **Step 1: Type of rule**.
2. In **Step 1: Type of rule** > **Name** field, enter **Send Event High to Python** and add an optional description of the rule in the **Description** field.
3. In the **Select the rule type you want you create** pane, select **Perform an action on <event>**
4. In the **Edit the rule description** pane, click **event** and in **Events** > **External Events** > **User-defined Events**, select **Event High**.
5. Click **OK** to create the event.

6. Click **Next** to go to **Step 2: Conditions**.
7. In **Step 2: Conditions**, apply any conditions relevant to the event you are creating.
8. Click **Next** to go to **Step 3: Actions**.
9. In **Step 3: Actions**, scroll down and select **Send event info to <Webhook>**.
10. Click **address** on the **Edit the rule description** pane
11. Add the **Python integration** webhook and click **OK**.
12. Click **Next** to go to **Step 4: Stop criteria**.
13. In **Step 4: Stop criteria**, click **Finish** to create the rule.

4. Test the webhook from XProtect (Optional)

You should test the event to verify that the event triggers and is received in the Flask server before utilizing the event and webhook in daily operations.

1. Open **Management Client** > **Rules and Events** and click **User-defined Events**.
2. Select **Event High** and click **Test Event** to test the event.
3. Open the console that is running the server. The **Received event from XProtect through Webhook** log message should be displayed.

5. Troubleshooting (Optional)

If you don't receive the events in your web server, open the MIP Logs from the Event Server tray icon to troubleshoot any potential errors.

The log of the error can take up to 90 seconds to appear because of the retry policy.

API Documentation

The webhooks API

This overview of the XProtect webhooks API is based on the Webhooks API, version 1.0.

The HTTP POST object consists of a header and the body (payload).

HTTP POST Body (version 1.0)

The payload of the HTTP POST is an application/json object with the following keys:

Key	Type	Description
Event	BaseEvent (serialized to JSON)	The event that triggered the rule.
Site	<pre>{ ServerHostName: string, AbsoluteUri: string, </pre>	Information about the system where this event originated. Use this field when using the same webhook endpoint for multiple XProtect installations.

	`` ServerType: string, } ```	

HTTP POST Body example

```
{
   "Event": {
       "EventHeader": {
           "ID": "2388f93d-5cb0-48e0-9524-d8bb981e1629",
```

```
            "Timestamp": "2022-11-26T15:59:39.2988877Z",
            "Type": "System Event",
            "Version": "1.0",
            "Priority": 1,
            "PriorityName": "High",
            "Name": "External Event",
            "Message": "External Event",
            "Source": {
                "Name": "Event High",
                "FQID": {
                    "ServerId": {
                        "Type": "XPCO",
                        "Hostname": "ec2amaz-111k11j",
```

```
                        "Port": 80,

                        "Id": "2f04e7ee-2ee2-49d4-8415-
de28dba7ef2e",

                        "Scheme": "http"

                    },

                    "ParentId": "2f04e7ee-2ee2-49d4-8415-
de28dba7ef2e",

                    "ObjectId": "13860010-ded3-42ff-a2a9-
cc6ba4a49636",

                    "FolderType": 0,

                    "Kind": "c9bdac3f-41dc-4afa-b057-61767a3914b7"

                }

            },

            "MessageId": "0fcb1955-0e80-4fd4-a78a-db47ee89700c"

        }
```

```
    },
    "Site": {
        "ServerHostname": "ec2amaz-111k11j",
        "AbsoluteUri": "http://ec2amaz-111k22j/",
        "ServerType": "XPCO"
    }
}
```

HTTP POST Headers

The HTTP POST headers contains the following keys:

Key	Type	Description
X-Milestone-Api-Version	String	The version of the body. “v1.0” is currently the only supported version.
X-Hub-Signature-256	String	The HMAC hex digest using SHA-256 of the body. The value always starts with “sha256=”. For more information, see “Improving webhook security with tokens”.

HTTP POST header example

```
{
   "X-Hub-Signature-256":
"sha256=LBr6+XIEOZKgMck2/aG1CNCaCCOAwHT4o+vYRE4D3JM=",
   "X-Milestone-Api-Version": "v1.0",
   "content-type": "application/json; charset=utf-8"
}
```

Failed requests and retry policy

If the POST requests fails with a 400 error, it is not retried again.

If the POST requests fails with any other status code, or a timeout, the request will be retried 2 more times, with each attempt spaced 30 seconds apart.

The same "Event.EventHeader.ID from the original HTTP POST Body will be used in all retries so the subscribing/receiving service can track any potential duplicate event notifications.

Improving webhook security with tokens

If you expose a webserver to listen to incoming webhooks, anyone who knows the URL can trigger the server response to the webhook by sending an HTTP POST request to the server. You can reduce the risk of this unauthorized triggering by defining a token for the webhook.

If you add a token to your webhook, you can validate that your XProtect installation is the actual source of the HTTP POST.

Other methods can be used to help secure the integration of the webserver with the webhooks, for example not exposing the server to the internet or using an IP whitelist.

To improve your webhook security, you must:

1. Generate a secret token.
2. Configure a secret token in Management Client.
3. Validate incoming requests on the receiving web server.

Generate a secret token

To randomly generate a token string, you can use an online generator, for example LastPass.com or use a code-based password generator function, for example the secrets.token_hex() function in Python 3.6 or greater.

Example:

```
import secrets; print(secrets.token_hex
(32))
```

The python command in the example creates a 32-byte hexadecimal text string token.

Configure a secret token in Management Client

1. Copy the text string you want to use as the token to the clipboard
2. In **Management Client** > **Rules and Events** > **Webhooks**, select the webhook you want to add a token to
3. In **Webhook Information** > **Token (optional)** field: Paste the token string for the webhook.
4. Click **Save** in the toolbar to update the webhook.

Validate incoming requests on the receiving webserver

Before processing a request, you must validate that the signature from the header matches the signature you compute.

Generate the signature using the base 64 encoded HMAC SHA-256 digest, for example:

```

```

```
signature = `sha256=${HMAC256(secret_token, body).toBase64().toString('utf-
8')}`
```

Signature validation examples

The following are examples of token string generation using various programming languages: Python, Node.js and C#.

Signature validation example – Python

```
import hashlib
```

```
import hmac
```

```
import base64
```

```
def is_signature_valid(body: bytes, header_signature: str, secret_token: bytes) ->
bool:
```

```
    digest: bytes = hmac.HMAC(key=secret_token, msg=body,
digestmod=hashlib.sha256).digest()
```

```
    expected_signature: str = f"sha256={base64.b64encode(digest).decode('utf-')}"
```

```
    return hmac.compare_digest(header_signature, expected_signature)
```

Signature validation example - Node.js

```
const crypto = require('crypto')

const isSignatureValid = (body, headerSignature, secretToken) => {

    const digest = crypto.createHmac("sha256", secretToken).update(body).digest();

    const expectedSignature = `sha256=${digest.toString('base64').toString('utf-8')}`

    return crypto.timingSafeEqual(Buffer.from(headerSignature), Buffer.from
(expectedSignature))

}
```

Signature validation example - C#

```
private static bool isSignatureValid(string data, string headerSignature, string token)

{

    var hasher = new HMACSHA256(Encoding.UTF8.GetBytes(token));
```

```
    var expectedSignature = $"sha256={Convert.ToBase64String(hasher.ComputeHash
(Encoding.UTF8.GetBytes(data)))}";
```

```
    return CryptographicOperations.FixedTimeEquals(Encoding.UTF8.GetBytes(expectedSignature),
Encoding.UTF8.GetBytes(headerSignature));
```

```
}
```

helpfeedback@milestone.dk

About Milestone

Milestone Systems is a leading provider of open platform video management software; technology that helps the world see how to ensure safety, protect assets and increase business efficiency. Milestone Systems enables an open platform community that drives collaboration and innovation in the development and use of network video technology, with reliable and scalable solutions that are proven in more than 150,000 sites worldwide. Founded in 1998, Milestone Systems is a stand-alone company in the Canon Group. For more information, visit https://www.milestonesys.com/.