# Nymi SDK for WebAPI on Windows Developer's Guide

**Nymi Connected Worker Platform**
**v1.0**
**2023-05-05**

# Contents

# Preface

Nymi™ provides periodic revisions to the Nymi Connected Worker Platform. Therefore, some functionality that is described in this document might not apply to all currently supported Nymi products. The product release notes provide the most up to date information.

## Purpose

This document is part of the Connected Worker Platform (CWP) documentation suite.

This document provides information about how to understand and develop Nymi-enabled Applications (NEA) by utilizing the functionality of the Nymi SDK, over a WebSocket connection that is managed by a web-based or other application. Separate guides are provided for Windows and iOS application development.

## Audience

This guide provides information to Developers.

## Revision history

The following table outlines the revision history for this document.

### Table 1: Revision history

| Version | Date | Revision history |
|---|---|---|
| 1.0 | April 28, 2023 | First release of this document for the CWP 1.7.0 release. |

## Related documentation

- **Nymi Connected Worker Platform—Overview Guide**

  This document provides overview information about the Connected Worker Platform (CWP) solution, such as component overview, deployment options, and supporting documentation information.
- **Nymi Connected Worker Platform—Deployment Guide**

  This document provides the steps that are required to deploy the Connected Worker Platform solution.

  Separate guides are provided for authentication on iOS and Windows device.
- **Nymi Connected Worker Platform—Administration Guide**

  This document provides information about how to use the NES Administrator Console to manage the Connected Worker Platform (CWP) system. This document describes how to

set up, use and manage the Nymi Band™, and how to use the Nymi Band Application. This document also provides instructions on deploying the Nymi Band Application and Nymi Runtime components.

- **Nymi SDK for C Developer's Guide**

  This document provides information about how to develop Nymi-enabled Applications by using the Nymi API(NAPI).
- **Connected Worker Platform with Evidian Installation and Configuration Guide**

  The Nymi Connected Worker Platform with Evidian Guides provides information about installing the Evidian components and configuration options based on your deployment. Separate guides are provided for Wearable, RFID-only, and mixed Wearable and RFID-only deployments.
- **Nymi Connected Worker Platform—Troubleshooting Guide**

  This document provides information about how to troubleshoot issues and the error messages that you might experience with the NES Administrator Console, the Nymi Enterprise Server deployment, the Nymi Band, and the Nymi Band Application.
- **Nymi Connected Worker Platform with Evidian Troubleshooting Guide**

  This document provides overview information about how to troubleshoot issues that you might experience when using the Nymi solution with Evidian.
- **Nymi Connected Worker Platform—FIDO2 Deployment Guide**

  The Nymi Connected Worker Platform—FIDO2 Deployment Guide provides information about how to configure Connected Worker Platform and FIDO2 components to allow authenticated users to use the Nymi Band to perform authentication operations.
- **Connected Worker Platform with POMSnet Installation and Configuration Guide**

  The Nymi Connected Worker Platform—POMSnet Installation and Configuration Guides provides information about how to configure the Connected Worker Platform and POMSnet components to allow authenticated users to use the Nymi Band to perform authentication operations in POMSnet.
- **Nymi Band Regulatory Guide**

  This guide provides regulatory information for the Generation 3 (GEN3) Nymi Band.
- **Third-party Licenses**

  The Nymi Connected Worker Platform—Third Party Licenses Document contains information about open source applications that are used in Nymi product offerings.
- **Connected Worker Platform Release Notes**

  This document provides supplemental information about the Connected Worker Platform, including new features, limitations, and known issues with the Connected Worker Platform components.

### How to get product help

If the Nymi software or hardware does not function as described in this document, you can submit a support ticket to Nymi, or email support@nymi.com

### How to provide documentation feedback

Feedback helps Nymi to improve the accuracy, organization, and overall quality of the documentation suite. You can submit feedback by using support@nymi.com

# Nymi SDK Overview

The Nymi SDK provides Developers with libraries, APIs, sample code and documentation to build a Nymi-enabled Application (NEA).

Nymi SDK delivers the Nymi API(NAPI) through a Windows Dynamically Linkable Library(DLL) named *nymi_api.dll* that developers include in a Windows application that supports a locally linked library.

# WebAPI Overview

Nymi WebAPI is an RFC-6455 compliant WebSocket. NEAs, such as web-based application use a standard WebSocket client to access Nymi WebAPI.

The Nymi WebAPI allows developers to utilize the websocket functionality of the Nymi SDK in a web-based or native application. The Nymi WebAPI architecture is part of the Nymi SDK.

**Note:** The WebAPI is supported on Microsoft Windows and iOS platforms only.

Nymi WebAPI provides bi-directional communication using requests/responses over a persistent connection. All messages sent and received are encoded in JSON format. The architecture provides continuous communication using WebSocket connections between the Nymi Agent and Nymi-enabled Application (NEA) running either as a native application or inside of a web client.

The WebAPI communicates with Nymi Bands over a WebSocket client and supports multiple NFC readers .

To enable NFC support, on the user terminal you must:

- Connect the NFC reader
- Install a compatible version of the Nymi Bluetooth Endpoint

To secure communication between Nymi Agent and WebAPI client applications, Nymi highly recommends that you enable TLS for the WebAPI interface.

When a user performs a Nymi Band tap, to complete an authentication or e-signature in WebAPI application, the Nymi Bluetooth Endpoint sends an intent event that represents the tap to the application through the interface of the Nymi Agent.

Configuration parameters are set in a TOML file, as described later in this document.

### WebSocket Keepalive Message

Nymi implements keepalive messages according to the RFC-6455 WebSocket Protocol standard for bi-directional communication. Nymi sends a ping message every 30 seconds to

the NEA and expects to receive a pong message response. The keepalive message indicates that the connection is still responsive.

Nymi-supported web browsers send a pong message in response to the ping control frame message. The pong control frame message ensures that the session is connected to the Nymi Bluetooth Endpoint. NEA supported web browsers do not require any additional configuration to support this functionality.

If you are using a native WebSocket client, additional implementation may be required.

**Note:** The WebSocket client, which is the NEA, disconnects from the Nymi Agent if there are no messages (including pings and pongs) sent or received for a period of 60 seconds.

# Development Tools

To develop NEAs on a Windows platform, you can use one of the following tools.

- Any Microsoft-supported version of Visual Studio.
- Visual Studio Code (or any other code editor).
- Any language that interfaces with a DLL, for example, Python

For C, C++, and C#, Nymi recommends that you use Visual Studio 2017.

# SDK Package

The SDK package contains the following folders:

- *..\nymi-sdk\windows\i686*—Contains the NAPI dll file for i686 user terminals.
- *..\nymi-sdk\windows\sampleApps*—Contains sample Nymi-enabled Applications(NEAs).
- *..\nymi-sdk\windows\x86_64*—Contains the NAPI dll file for i686 user terminals.
- *..\nymi-sdk\windows\setup\BleDriver_x64.msi*—64-bit Bluegiga driver installation file.
- *..\nymi-sdk\windows\setup\BleDriver_x86.msi* —32-bit Bluegiga driver installation file.
- *..\nymi-sdk\windows\setup\NymiRuntime-5.9.1.8.msi*—Nymi Runtime MSI installation file.
- *..\nymi-sdk\windows\setup\Nymi Runtime installer.`version`.exe* —Nymi Runtime installation file.

# Sample Application

The Nymi SDK package includes a sample application that demonstrates some of the key functionality of the Nymi solution.

The sample applications is a simple Javascript application that demonstrates all the basic functions that are supported by the API and allows a user to see both JSON request and response examples to help understand how the API works.

### Sample Application for Nymi WebAPI

The sample application for Nymi WebAPI is located in the *..\nymi-sdk\windows\javascript \webapiSample* folder. The application prompts you for the configuration parameters that are unique to your environment.

# Creating NEAs with Nymi WebAPI

Customer and partner developers can use the Nymi WebAPI to develop Nymi-enabled Application (NEAs) in programming languages, such as Java or C#. The API is based on JSON messages that are exchanged with the server over a websocket connection. This chapter provides information about the supported operations.

To deploy an NEA, developers must install the Nymi Runtime on each terminal where the NEA runs. The Nymi Runtime includes the following components: Nymi Bluetooth Endpoint, and Nymi Agent.

**Note:** In this document, the use of device refers to the Nymi Band.

The Nymi Band provides authentication information about a user to applications. An application can use this information on a point-in-time basis (for simple authentication) or continuously (for both authentication and de-authentication).

Additionally, the Nymi Band can authenticate by using NFC-only (which is significantly less secure) and NFC with Bluetooth (which is exceptionally secure). The following figures provide example workflows for both authentication use cases.

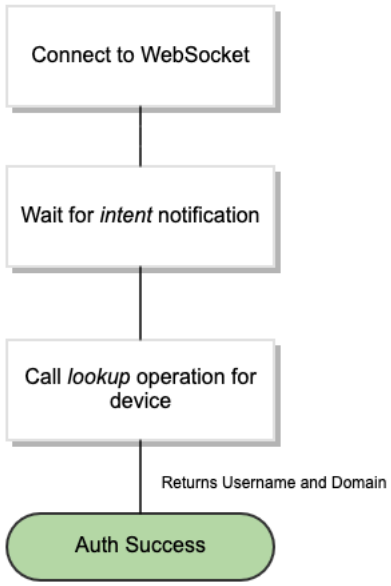
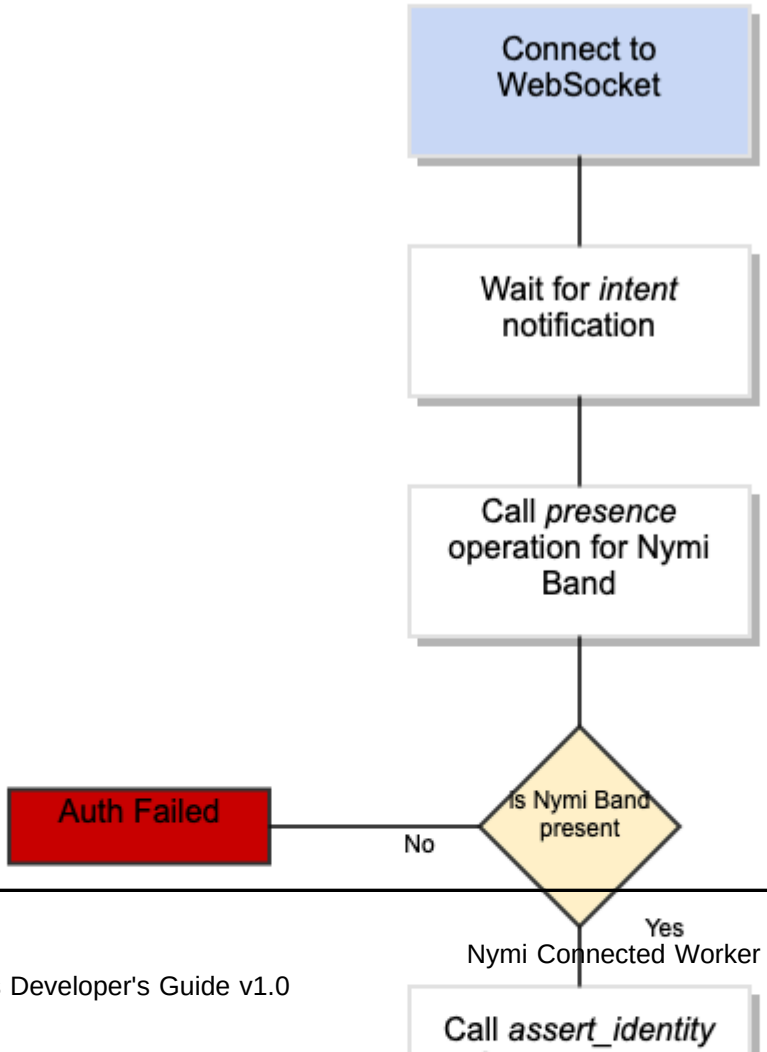



**Figure 1: NFC-only communications**



In both authentication examples, the first step is to wait for an intent notification. The intent operation tells the application that a user has placed their Nymi Band on an NFC reader that is connected to the workstation. The intent operation returns a device ID, which is the standard identifier of a Nymi Band in the CWP solution.

In the NFC-only example, the application requests a lookup operation, which returns the username and domain of the user that is associated with the Nymi Band. In applications that use the NFC-only model as a secure replacement for badges, the authentication is complete.

In the fully secure NFC with Bluetooth mode, after the intent notification returns a device ID, the application ensures that the device is present. This action is performed in one of the following ways:

- Passively as NAPI continuously sends notifications about present Nymi Bands.
- Actively by requesting a presence operation with the desired device ID, and then waiting for a response.

For passive notifications, since NAPI sends notifications for the full list of Nymi Band present at start-up, an application can track all present bands and then check its list of current Nymi Bands. After presence is established, the application can request an assert_identity operation for the Nymi Band. The assert_identity operation uses a bi-directional challenge-response to establish a secure channel between the Nymi Agent and the requested Nymi Band. When the action results in the establishment of the secure channel, the assert_identity verifies the authentication state of the Nymi Band. When the assert_identity operation completes successfully the operation passes the username and domain of the associated user back to the application, and the application can continue with an absolute assurance that the Nymi Band is present and authenticated to the correct user.

**Note:** The Nymi Band exchanges data over Bluetooth Low Energy(BLE) and the exchange consists of several cryptographic operations. As a result, the assert_identity operation can take up to two seconds to complete.

Continuous monitoring of the WebSocket to watch for presence notifications indicates to an application when a user has authenticated, de-authenticated (by removing their Nymi Band), or when the user leaves a physical area. The presence notifications always returns one of the following statuses for a single Nymi Band.

- Weak—The Nymi Band is present. A strong presence is represented by the successful return of an assert_identity operation).
- Absent—The Nymi Band is not present.
- Unauthenticated—The Nymi Band is not authenticated.

**Note:** The loss of presence triggers an application to log out, lock, or remove user access to functionality.

# Bluetooth Notifications

Nymi Bluetooth Endpoint is a client service that communicates with the Bluetooth Adapter. Bluetooth notifications for Bluetooth Adapter status are non-transactional.

The Bluetooth Adapter communicates to the Nymi Band. Each time that a Bluetooth Adapter becomes available, the *update* function retrieves a notification in the following format.

```
{
  "operation": "ble_ready",
  "exchange": null,
  "status": 0,
  "payload": {},
  "error ": {}
}
```

If a Bluetooth Adapter becomes unavailable, the *update* function retrieves an error notification in the following format.

```
{
  "operation": "error",
  "exchange": null,
  "payload": {},
  "status": "error_code",
  "error": {
      "error_description":"error_description>",
      "error_specifics":"error_specifics"
   }
 }
```

where *error_code* is one of the following values: 5000, 5010, 5100.

For more information about error codes, see *Error Handling*.

# Presence Operation

Using the *presence* request, you can retrieve the current state of the Nymi Band. Presence requests are non transactional. The presence request has no response and a presence response is not tied to a specific request.

When a *presence* request is sent, the system will replay the last presence update received. When a presence state changes you will receive automatic notifications. For information about these notifications, see *Presence notifications.*

Presence is relative to an endpoint (the response indicates if the Nymi Band is in range of the NEA). A Nymi Band can be present on some endpoints, but absent on others. If the presence state is false the presence state returns as `absent`.

### JSON Object Format

Define the *presence* request JSON object in the following format.

```
{
  "operation": "presence",
  "exchange":"exchange_value",
   payload":{
     "device": "NymiBandID",
     "proximity" : "proximity value",
     "service_request_state" : "service request state",
     "state" : "state"
   },
}
```

where:

- *NymiBandID*: Is the Nymi Band MAC address.
- *proximity_value*: Is determined by the distance between the Nymi Band and the BLE adapter. The *proximity_value* will change when the Nymi Band moves closer or farther from the BLE adapter.
- *state*: Is determined by the state of the Nymi Band; weak, absent, or unauthenticated. The following table describes the state values in more detail:

**Table 2: State values for presence**

| State Value | Definition |
|---|---|
| Absent | The Nymi Agent cannot communicate with the Nymi Band. This state also applies when a user wears an unenrolled Nymi Band.<br>Reasons for Nymi Band absence include:<br>• Nymi Band has been removed from the body.<br>• Nymi Band has not communicated with the Nymi Agent for at least 30 seconds.<br>• Nymi Band has not been within the range of the BLE Adapter for at least 30 seconds. |
| Unauthenticated | Nymi Band is enrolled and but not authenticated. |
| Weak | Nymi Band is in an authenticated state. |

- *service request state*: Is a flag that accompanies each presence notification and determines if there is a message in the Nymi Band that is ready to be downloaded. If the value of `service_request_state` is not zero, the Nymi Band has service level messages. If the value is '0', there are no messages

# Presence Notifications

When Nymi WebAPI detects a change in Nymi Band presence, Nymi WebAPI generates a presence notification.

After the Nymi-enabled Application establishes the websocket session, the system sends an updated notification each time any of presence parameters change.

It is recommended that you develop a method for your application that tracks when the Nymi Bands come in and out of range.

Presence notifications appear in the same format as the presence operation.

# Subscribe_endpoint Operation

The `subscribe_endpoint` operation allows an NEA to change the Nymi Bluetooth Endpoint to which it is subscribed.

*subscribe_endpoint* request operations appear in the following format:

```
{
"operation": "subscribe_endpoint",
"exchange":"exchange_value",
"payload": {
"endpoint_id": "ip_address"
}
}
```

where:

- *operation* is *subscribe_endpoint*.
- *exchange* is any value and is used to match the response to the request.
- *endpoint_id* is based on the endpoint IP address. Required when the configuration uses a centralized Nymi Agent.
- *endpoint_id* is a unique identifier that an NEA assigns to every iOS device. The NEA passes the same value to the Nymi Application, when the NEA invokes the Nymi Application.

The *subscribe_endpoint* operation returns a status code only, no errors are returned.

```
{
“operation”: “subscribe_endpoint”,
"exchange":"exchange_value",
“payload”: {}
“status”: 0,
“error”: {}
}
```

You can only subscribe an NEA to one endpoint at any given time. When you request the *subscribe_endpoint* operation, the NEA is automatically unsubscribed from the previously

subscribed endpoint. Any Nymi Bands that were present on the previously subscribed endpoint, become absent, and the NEA receives corresponding presence update notifications. The NEA will then receive a Bluetooth status notification. If the requested Nymi Bluetooth Endpoint has connected successfully and is in a ready state, the NEA will receive a ble_ready notification, followed by presence update notifications for any present bands on that endpoint. Otherwise, the NEA will receive an error message. See *Bluetooth Notifications* for more information about possible error messages.

**Note:** The NEA remains subscribed to the requested endpoint_id even if it is not able to connect to that Nymi Bluetooth Endpoint. If the Nymi Bluetooth Endpoint becomes ready at a later time (for example, when a user turns on the user terminal), then NEA receives a ble_ready message at that time.

# Intent Notification

An intent occurs when a user taps their authenticated Nymi Band next to an NFC reader or Bluetooth radio antenna, and is used to signal an intent to take an action. For example, an intent to provide an e-signature is generated when a user taps their authorized Nymi Band against an NFC reader.

To ensure that intent notifications are received, specify the NES server in the *nymi_agent.toml* .

Intent notifications appear in the following format:

```
{
  "operation": "intent",
  "exchange": null,
  "payload": {
     "device": "NymiBandID",
     "type": "see below",
  },
  "status": 0,
  "error": {}
}
```

where *device* is the Nymi Band MAC address.

*type* is used to identify the manner in which the action was initiated.

### Table 3: Intent Payload Types

| Type Field | Description |
|---|---|
| ble | A user tapped an authenticated Nymi Band against a BLE device or is in close proximity to a BLE radio antenna, such as a BLE adapter. |

| Type Field | Description |
|---|---|
| nfc | A user tapped an authenticated Nymi Band against an NFC reader or is in close proximity to read range of the NFC reader. |

### Status Codes

A 2201 status code is reported when the NFC reader is unsuccessful at mapping the NFC ID to the enrolled Nymi Band.

A 2200 status code is reported when a NES communication error (for example, NES is offline) occurs.

**Note:** The 2201 and 2200 status codes do not contain a NymiBandID in the payload.

# Lookup Operation

Use the *lookup* operation to determine the following values:

- Device ID ( MAC address) of the Nymi Band.

  **Note:** An intent notification includes the device ID or you can retrieve the device ID of a Nymi Band from NES by using the lookup operation.
- NfcUID of the Nymi Band.
- Domain and name of the user.
- User status in Active Directory (AD). The AD status for a user appears in the response when user status check is enabled in NES. The following table summarizes the possible user statuses.

#### Table 4: AD user statuses

| User Status | Definition |
|---|---|
| Active | User account is enabled. |
| NotExist | User account was deleted from AD. |
| Inactive | User account is disabled. |
| Active \| Locked | User account is locked. This status can appear with Active and Password Expired. |
| Active \| PasswordExpired | User account has an expired password. This status can appear with Active and Locked. |

By default, NES is not configured to perform user status checks in AD. Contact the NES Administrator to enable AD user status checking, and optionally the checking interval in the NES Administrator Console.

### JSON Object Format

Define the *payload* JSON object for the *lookup* command in the following format.

```
{
   "operation": "lookup",
   "exchange": "exchange_value",
   "payload":  {
      "nes_url": "https_url_to_nes",
      "query": "query_JSON",
      "lookup_keys": "key_JSON"
    }
 }
```

where:

- *nes_url* the NES URL.
- *query* field is a JSON object that defines the values that are passed during the request to retrieve the response. Acceptable values include *NfcUID*, *Domain* and *Username*, and *NymiBandID*.

  **Note:** The property names *Domain* and *Username* are case-sensitive.
- *lookup_keys* field is a JSON array that contains a list of values that you want to appear in the response. Supported values include *NfcUID*, *Domain* and *Username*, *NymiBandID*, and *UserStatus*.

**Example 1**

The following code block provides an example of a JSON object that instructs Nymi WebAPI to provide the NfcUID of a device and the user status for a user named *JSmith* in the *MyCorpDomain* domain.

```
{
   "operation": "lookup",
   "exchange": "rAndOm_IdeNtifyiNG_StrING_1218",
   "payload": {
    "nes_url": "https://nes.nymi.com/nes/",
    "query": {
       "Domain":"MyCorpDomain",
       "Username": "JSmith"
     }
   "lookup_keys": ["NfcUID", "UserStatus"]
   }
}
```

### Result 1

A successful *lookup* operation produces a response with the following properties.

In this example, the check user status in AD option is enabled in NES, as a result, the response includes the *UserStatus* property.

```
{
  "operation": "lookup",
  "exchange":"rAndOm_IdeNtifyiNG_StrING_1218",
  "payload": {
      "lookup_values":{"NfcUID": "1234xyz", "UserStatus":"Active|PasswordExpired"},
  },
  "status": "0",
  "error: {}
}
```

**Example 2**

The following code block provides an example of a JSON object that instructs Nymi WebAPI to provide the NfcUID of a device with Nymi Band (or device) ID *"C2:FA:D7:F0:D7:96"*.

```
{
  "operation": "lookup",
  "exchange": "rAndOm_IdeNtifyiNG_StrING_1218",
  "payload": {
      "nes_url": "https://nes.nymi.com/nes/",
      "query": {
      "NymiBandID": "C2:FA:D7:F0:D7:96"
      }
      "lookup_keys": ["NfcUID"]
  }
}
```

### Result 2

A successful *lookup* operation produces a response with the following properties.

```
{
  "operation": "lookup",
  "exchange":"rAndOm_IdeNtifyiNG_StrING_1218",
  "payload": {
      "lookup_values": {"NfcUID": "1234xyz"},
   },
  "status": "0",
  "error: {}
}
```

# Troubleshooting

Nymi API writes information to log files that allow you to monitor and troubleshoot the NEA.

For additional assistance, visit the Support page on the Nymi website, or contact your Nymi Solution Consultant.

The following table summarizes the log files that are available for troubleshooting.

**Table 5: Log file locations**

| Component | Log location | Files |
|---|---|---|
| Nymi API | By default, the current working directory. | *nymi_api.log* |
| Nymi Agent | *C:\Nymi\NymiAgent* | *nymi_agent.log* |
| Nymi Bluetooth Endpoint | *C:\Nymi\Bluetooth_Endpoint \logs* | *nymi_bluetooth_endpoint.log* |

# Enable debug mode

When testing Nymi WebAPI and builds, set the `NYMI_DEBUG` environment variable to any value to enable debug logging, and the restart the Nymi Agent and Nymi Bluetooth Endpoint services.

Published in Canada.
Nymi Inc.
Toronto, Ontario
www.nymi.com