Nymi SDK for WebAPI on iOS Developer's Guide

Nymi Connected Worker Platform
v1.0
2023-04-20

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 21, 2023	First release of this document for the CWP 1.5.6 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).

Overview of an iOS Environment

The following diagram provides a high-level overview of the components in a Connected Worker Platform environment that uses iOS devices to access web-based NEAs.

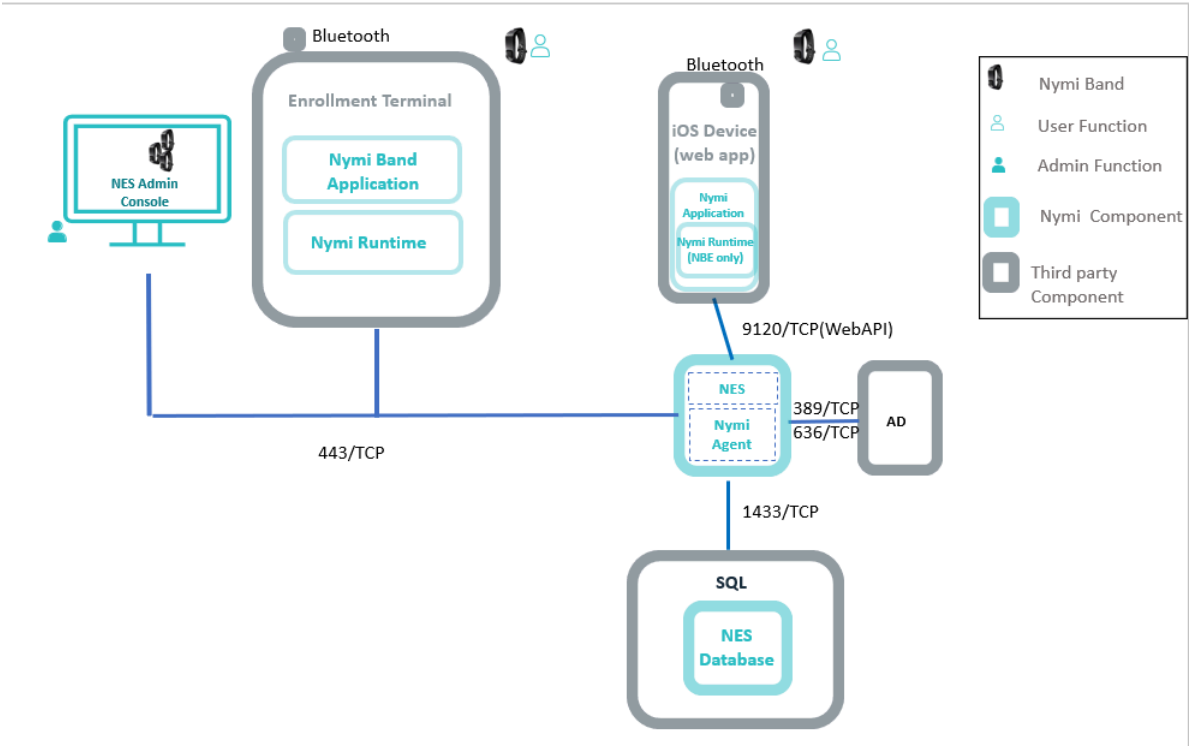

Figure 1: Connected Worker Platform Solution Components and Communication

Table 2: Connected Worker Platform Solution Components

Component	Description
Enrollment Terminal	Windows 10 endpoint that users access to enroll their Nymi Band.

Component	Description
Nymi Band Application (NBA)	A Windows application that you install on the enrollment terminal and is used to enroll a new user and link them to their Nymi Band. The Nymi Band Application requires the Nymi Runtime application, which the Nymi Band Application automatically installs. The Nymi Band Application communicates with the Nymi Band through the Nymi-supplied Bluetooth adapter, which you plug into a USB port on the enrollment terminal.
Nymi Band	A wearable device that is activated by the assigned user's biometrics. An authenticated Nymi Band is Bluetooth Low Energy (BLE) and Near Field Communication (NFC)-enabled. See the Nymi Band section in this guide for more information.
Nymi-enabled Application	Developers can create corporate applications that integrate with Connected Worker Platform by using the WebAPI component in the Nymi API. These applications are called Nymi-enabled Applications (NEAs) and include Manufacturing Execution Systems (MES), Single Sign-On (SSO), and Human Machine Interface (HMI) applications.
iOS Device	An iPad endpoint that users use to: • Perform authentication tasks in a web-based Nymi-enabled Application(NEA). • Perform authentication tasks in a native iOS NEA.
Nymi Application	Required when the iOS device connects to a web-based NEA to perform authentication tasks. A Nymi-supplied native iOS application that: • Embeds the Nymi Bluetooth Endpoint application, which provides an interface between the native Bluetooth Adapter (BLE) and the Nymi Agent. • Detects an intent to perform an authentication task with a Nymi Band (a tap) and passes the request to the NEA.

Component	Description
Centralized Nymi Agent	Required for environments that use iOS devices. Provides BLE management, manages operations and message routing. Facilitates communication between NEAs and the Nymi Band, and maintains knowledge of the Nymi Band presence and authenticated states. The Nymi Agent is installed in a central location on a single machine or a cluster of two or more machines that is accessible to all user terminals, for example on the NES server. To enable Nymi WebAPI communications between the Nymi Agent and the Nymi Bluetooth Endpoint, you must configure a *nymi_agent.toml* file. The *Nymi Connected Worker Platform—Deployment Guide* provides more information about how to configure the *nymi_agent.toml* file.
Nymi Enterprise Server (NES)	• A management server and collection of services that provides the NES Administrator Console and coordinates communication between the Nymi Band and the customer identity ecosystem (Active Directory) to manage policies and certificates. Includes the following services: • Enrollment Service (ES)—Authenticates, validates, and authorizes certificate requests from requesters, such as the Nymi Band Application and NEAs. • Directory and Policy Services (DPS)—Maintains the NES database, which contains a list of Active Directory (AD) users and the Nymi Bands that are associated with each user. Provides IIS web services, which allows the NES Administrator Console access to the NES database. • Authentication Service (AS)—Provides authentication and authorization support for domain users and computers. AS uses adapters to interface with external directory and database systems, such as an AD adapter to interface with Active Directory.

Component	Description
SQL Server	Database that contains table that store information about NES configuration and Nymi Bands. For Proof of Concept (POC) and pre-production environments, you can use the Nymi-provided SQL Server Express software. For production environments Nymi recommends that you use SQL server.
Domain Controller (DC)	Windows server with Active Directory.

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 the integrated Bluetooth device on iOS .

To enable Bluetooth support on the iOS device when you access a web-based Nymi-enabled Application(NEA), you must install the Nymi Application

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.

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 the iOS platform, use use XCode version 14 or later.

SDK Package

The SDK package contains the following folders:

- *nymi-sdk/ios/readme.md*—Readme file.
- *....nymi_sdk/iOS/Apps/NymiApplication*—Installation file and configuration file for the Nymi Application.
- *..nymi_sdk/iOS/NBE_iOS_Framework*—Nymi Bluetooth Endpoint compiled as an iOS framework to integrate with the native iOS application.
- *..nymi_sdk/iOS/sampleApps/browserApp*—Web-based NEA sample application for iOS devices.
- *....nymi_sdk/iOS/sampleApps/NymiSDK_iOSSampleApp*—Contains the sample native iOS Nymi-enabled Applications(NEAs).

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

Nymi WebAPI (iOS) includes two sample applications:

- Nymi SDK iOS Sample App—A sample application that demonstrates to iOS developers how to integrate the SDK into their native iOS application, which is located within the package at: *nymi-sdk/ios/sampleApps/NymiSDK_iOSSampleApp*.
- Nymi Sample Browser App—A sample browser application that demonstrates to iOS developers how to integrate the SDK into their web-based applications and interact with

the Nymi Application to support the completion of e-signatures on an iOS device, which is located within the package at: *nymi-sdk/iOS/sampleApps/browserApp*

Creating NEAs with Nymi WebAPI

Customer and partner developers can use the Nymi WebAPI to develop Nymi-enabled Application (NEAs) for iOS. 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 support communications between web-based NEAs and the Nymi Band, the environment requires an NES server and a centralized Nymi Agent that is accessible from the iOS devices. Additionally, you must install the Nymi Application on each iOS device. *Nymi Connected Worker Platform—Deployment Guide* provides more information.

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).

The Nymi Band performs authentication activities over Bluetooth. The following figure summarizes the authentication workflow that the solution follows when a user performs authentications tasks with a Nymi Band on an iOS device that accesses a web-based NEA (web application).

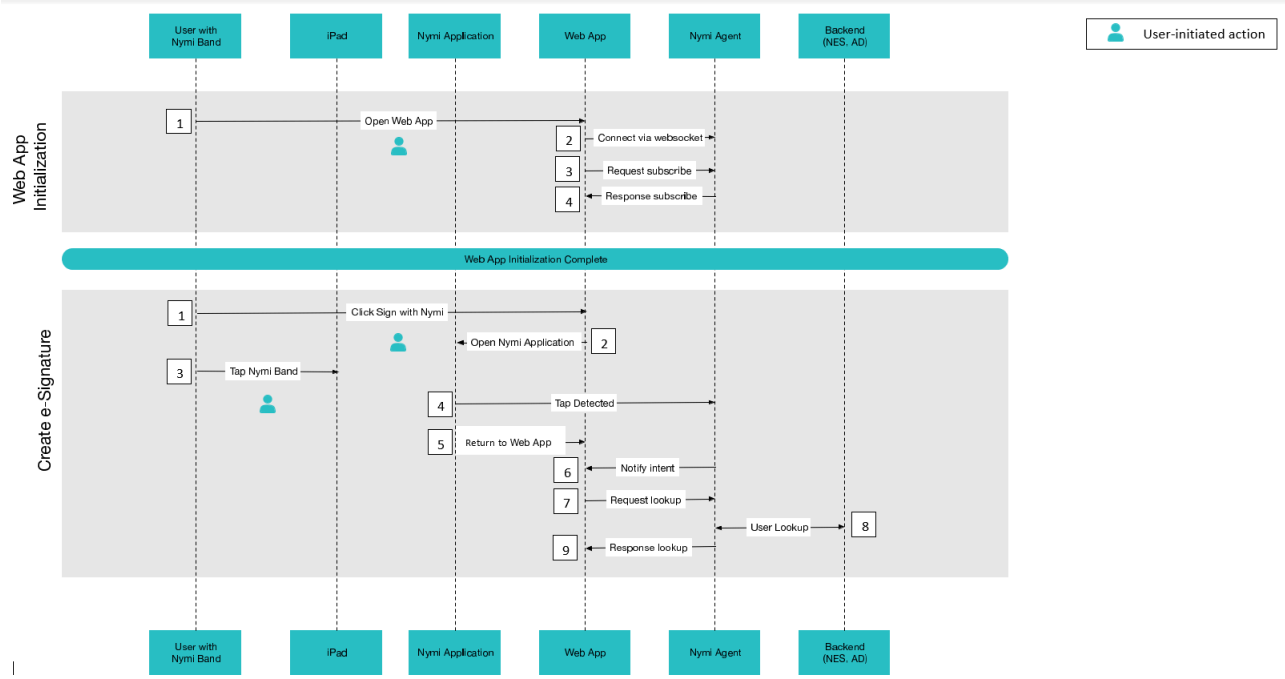

Figure 2: Authentication Workflow for iOS Devices

The authentication workflow includes two distinct phases. Each phase includes user-initiated and application-initiated actions.

Phase 1—Initialize Web Application

This phase occurs each time a user connects to the web application and results in the establishment of connectivity between the web application and the Nymi components.

1. User opens the web application from a browser on an iOS device.
2. NEA establishes a WebSocket connection to the Nymi Agent.
3. Web application sends a *subscribe* request to the Nymi Agent.
4. Nymi Agent responds to the request. Nymi Agent returns a success response .

Phase 2—Create E-signature

This phase occurs each time a user performs an e-signature with the Nymi Band and results in the completion of an e-signature with the tap of a Nymi Band.

1. From a window within the web application that requires an e-signature, the user clicks the button on the screen to use the Nymi Band.
2. Nymi Application appears on the screen and prompts the user to tap their Nymi Band on the iOS device.
3. User taps the Nymi Band on the iOS device near the Bluetooth antenna.
4. Nymi Application detects the tap and send information to the Nymi Agent.

5. Nymi Application returns control to the web application. The iOS screen focus changes back to the web application.
6. Nymi Agent notifies the web application that there is an intent request to use the Nymi Band to perform the e-signature.
7. Web application sends a *lookup* request to the Nymi Agent
8. Nymi Agent sends the *lookup* request to Nymi Enterprise Server(NES), NES contacts Active Directory, confirms the user credentials, and returns the response to the Nymi Agent.
9. Nymi Agent sends the response results to the web application. The web application reviews the response and based on the results, completes the e-signature or does not complete the e-signature.

Using Nymi WebAPI to Create Web-Based NEAs

To use the Nymi WebAPI to create web-based Nymi-enabled Applications(NEAs), perform the following high-level steps.

Procedure

1. Setup a web server and setup a webpage, ensure that you enable websocket support.
2. If the Nymi Application is running on an iOS device, close the Nymi Application.
3. Define the following parameters in your web application:

   - *return_url*—A string value that defines the URL of the current page, which the Nymi Application uses to navigate back to the browser. Do not include spaces.
   - *endpoint_id*—A string value that defines a unique identifier for the iOS device. Do not include spaces.
   - *nymi_agent_host*—A string value that defines the Nymi Agent host name or IP address, which the NEA passes to the Nymi Application. Do not include spaces.
   - *nymi_nes_url*—A string value that defines URL of the Nymi Enterprise Server(NES). Do not include spaces.
   - *nymi_agent_port*—A number value between 1 and 65535 that defines the port number on which the Nymi Application can communicate with the Nymi Agent. The default port is 9120.The value must match the value that is defined in the configuration of the Nymi Agent.
   - *integrator_name*—A string value that that defines the name of the web integrator that appears in the Nymi Application.

     Consider the following when you specify the *integrator_name*:

     - Do not include spaces.
     - Do not include any of the following characters: *% < > ? : " ^ @ # { } | \ [ ]*

     The following figure provides an example of the Nymi Application when the integrator name is *nymi*.

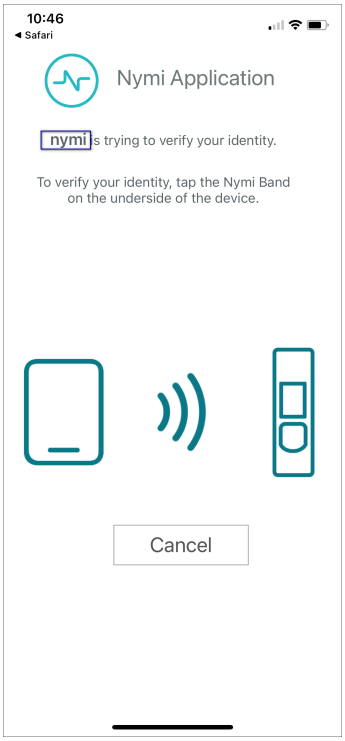

Figure 3: Integrator Name

4. Implement or use an existing method in your web application that assigns a unique and persistent identifier to each iOS device.

   Your application will use this identifier as *endpoint_id*. Nymi WebAPI uses *endpoint_id* to create an association between the websocket sessions and the Nymi Application instances that connect to the system. You will pass the *endpoint_id* to the *subscribe* operation, as explained later in this guide.

   Note: This document does not provide implementation details about how to generate, store, and retrieve the unique value.

5. Include in your web page a mechanism, such as a button, that invokes the Nymi Application through the use of the custom URL scheme named *nyminbe*. Include all the required parameters, as shown in the following example:

   The format of the call to invoke the schema is:

```
var target_url = "nyminbe://" + integrator_name + "?nymi_agent_host=" + nymi_agent_host + "&nymi_agent_port=" +
 nymi_agent_port + "&return_url=" + return_url + "&endpoint_id=" + endpoint_id;
```

What to do next

The Nymi SDK package contains a *Readme.md* file that provides detailed information about how to create a web-based NEA that invokes the Nymi Application scheme and how to modify the Nymi-supplied sample web-based application.

Note: Before you make changes to the Nymi-supplied sample web-based application, ensure that you close any running Nymi Application.

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 3: 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. Reasons for Nymi Band absence include: • Nymi Band has been removed from the body. • Nymi Band has not communicated with the Nymi Agent for at least 30 seconds. • 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": "mobile_endpoint_id"
  }
}
```

where:

- *operation* is *subscribe_endpoint*.
- *exchange* is any value and is used to match the response to the request.
- *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”: {}
}
```

An NEA can only be subscribed to one endpoint at any given time. When a subscribe operation is requested, the NEA is automatically unsubscribed from the endpoint it was previously subscribed to. If any Nymi Bands were present on that endpoint, they will become absent, and the NEA will receive 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 will remain 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, that workstation is powered on), the NEA will receive 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* or the *index.js* for a web-based application..

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 4: 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.

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 5: 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 6: Log file locations

Component	Log location	Files
Nymi Agent	*C:\Nymi\NymiAgent*	*nymi_agent.log*
Nymi Application	Open the Nymi Application, and then select **Logs**.	n/a

Server Closes Web Connection

When a Nymi Band user performs an authentication task, such as an e-signoff, the web server does not receive the `intent` message.

The following message appears in the developer tools window of the web browser:

> WebSocket connection to '`nymi_agent_server` failed. The operation couldn't be completed. (kNWErrorDomainPOSIX error 53 - Software caused a connection abort)

The *nymi_agent.log* file displays the following error:

```
DISCONNECT "nymi_agent_server:63839": {:error, :closed}
```

Cause

By default Safari closes the websocket connection after 2 seconds of inactivity. If the user does not complete the Nymi Band tap within the two seconds time period in the Nymi Application, Safari closes the web connection.

Resolution

Extend the time that the connection remains opened to 1 minute.

From the **Settings**, navigate to **Safari > Advanced > Experimental Features** and disable the **NSURL Session WebSocket** option.

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