Nymi SDK for WebAPI Developer's Guide

Nymi Connected Worker Platform
v1.0
2022-05-16

Contents

Nymi SDK Overview

The Nymi SDK provides Developers with libraries, APIs, sample code and documentation to build a Nymi-enabled Application (NEA). The `Nymi WebAPI` architecture is part of the Nymi SDK.

Nymi SDK delivers an API through two mechanisms:

- Dynamically linkable library that can be included in applications that are able to locally link library code.
- Over an RFC-6455 compliant WebSocket that is accessed by using a standard WebSocket client. This latter option is called the WebAPI for the rest of the document.

The Nymi SDK package contains the following components:

- `Nymi Runtime` - Handles the primary functions of the Nymi Band communication and consists of the following components:
  - `Nymi Agent` - 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.

    `Nymi Agent` provide an interface over port 9120 that listens for connections from the `Nymi Bluetooth Endpoint` and previously mentioned dynamic library. Additionally, you can enable the `WebAPI`, as described later in this document, and provide a websocket listener on a configurable port. This document provides you with information about the websocket mechanism.

    You can install `Nymi Agent` on each workstation or install `Nymi Agent` in a central location, and then specify the location of the `Nymi Agent` in an `Nymi Bluetooth Endpoint` configuration file.
  - `Nymi Bluetooth Endpoint` - Provides an interface between the Bluegiga Dongle (BLE) and the Nymi Agent. You deploy `Nymi Bluetooth Endpoint` on individual workstations to provide local BLE communication with Nymi Bands through the Nymi-provided Bluegiga Adapter.

Nymi SDK Package Contents

The Nymi SDK package contains the following artifacts:

- *nymi_api.dll* (`NAPI`)
- sample applications
- *BleDriver_xx.msi*
- `Nymi Runtime` installer

Nymi SDK Components and Communication

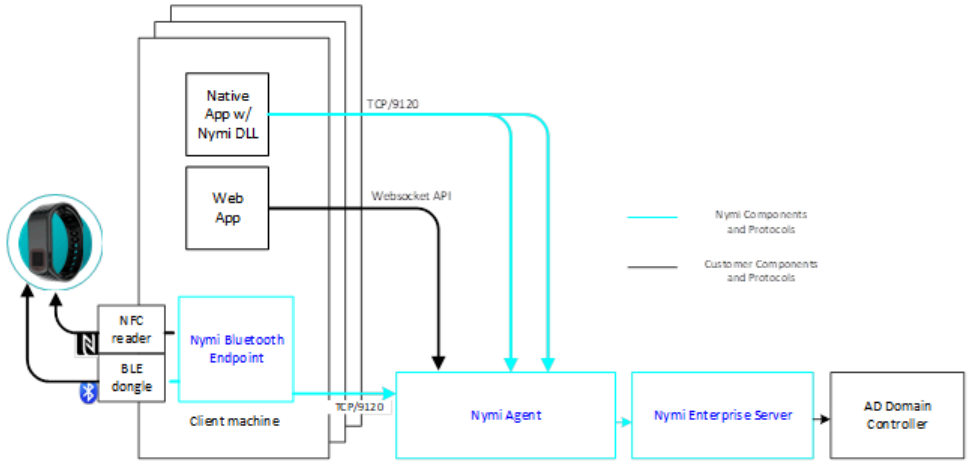


Figure 1: Nymi SDK Components and Communication

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.

Supported Platforms and Web Browsers

Platforms

The Nymi Agent is implemented on the following platforms.

- Microsoft Windows 10, 64-bit
- Microsoft Windows 7, 32-bit and 64-bit

Web Browsers

The `Nymi WebAPI` works with most modern web browsers. It has been tested with:

- Firefox 70 or later
- Chrome 78 or later
- Internet Explorer 11 or later
- Microsoft Edge 44.18362.387.0

Importing the TLS Certificate into Firefox

If you have issued your own TLS root certificate using a private certificate authority (CA), before Firefox can open a WebSocket connection for the NEA, you need to import the TLS certificate.

About this task

See *https://wiki.mozilla.org/CA/AddRootToFirefox* in the Mozilla documentation for more information.

Procedure

1. Open Firefox web browser.
2. In the right pane, navigate to **`Options`**.
3. Select **`Privacy and Security`**.
4. Under **`Certificates`** click **`View Certificates`** and then select **`Authorities`**.
5. Click **`Import`** and select the TLS root certificate from your machine.
6. Click **`OK`**.
7. Run the `Nymi WebAPI` and open the WebSocket connection by using Firefox.

Supported NFC Readers

When you connect a supported NFC reader to a user terminal where the Nymi Bluetooth Endpoint is installed, `Nymi Bluetooth Endpoint` automatically detects and monitors all attached NFC readers, and then forwards events from all NFC readers to the NEA through the `Nymi Agent`

A list of supported NFC Readers is found in the *Nymi Connected Worker Platform Administration Guide*.

WebAPI Architecture

This guide contains information that helps you to understand and develop web-based Nymi-enabled Applications. Nymi recommends that you read the guide before beginning development.

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

Note: The `WebAPI` is supported on Microsoft Windows platform only.

`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

When a user perform an NFC tap while 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`.

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

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 Nymi-enabled Application and expects to receive a pong message response. The keepalive message indicates that the connection is still responsive.

Nymi supported web browsers, see the *Supported Platforms and Web Browsers* page, will send a pong message, which is sent 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 Nymi-enabled Application, disconnects from the `Nymi Agent` if there are no messages (including pings and pongs) sent or received for a period of 60 seconds.

NEA Certificates

When you enable `WebAPI` in the `Nymi Agent` service, the `Nymi Agent` authenticates with NES by using Integrated Windows Authentication and the service account that runs the `Nymi Agent` service to get a user authentication token. `Nymi Agent` then passes the token to *nymi_api.dll* to allow `NAPI` to retrieve the device authentication certificates.

When using `Nymi WebAPI`, the keystore that contains the certificates resides on the machine that is running the Nymi Agent service.

By default, the keystore is in the *%APPDATA%\Roaming\Nymi* directory.

Alternative locations include:

- *C:\Windows\ServiceProfiles\LocalService\AppData\Roaming\Nymi* for the Local Service account.
- *C:\Windows\system32\config\systemprofile\AppData\Roaming\Nymi* for the LOCAL SYSTEM (64-bit binary) account.
- *C:\Windows\SysWOW64\config\systemprofile\AppData\Roaming\Nymi* for the LOCAL SYSTEM (32-bit binary) account.

Nymi WebAPI Sample Application

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

The sample applications are located within the package at: ..*nymi-sdk**windows**sampleApps**javascript**webapiSample*.

Configure the Development Terminal

On the development terminal, install the Nymi software and the required certificates.

Deployment of the Nymi WebAPI

You can deploy the `Nymi WebAPI` in a local or remote configuration.

In a local environment, you can deploy `Nymi Agent` on each workstation and connected to a single local `Nymi Bluetooth Endpoint` and Nymi-enabled Application(NEA) pair.

In a remote environment, such as Citrix and RDP published applications or desktops, deploy `Nymi Agent` in a central location that multiple workstations can access.

Note: For more information about how to deploy `Nymi Agent` see the Nymi Connected Worker Platform NES Deployment Guide.

The `Nymi Bluetooth Endpoint` and NEA must know the identity of the workstation to which the application wants to connect. By default, this identity is the IP address of the workstation. When you deploy `Nymi Agent` locally on the client workstation, both components use the loopback address, so they will connect automatically. When you install the `Nymi Agent` centrally, it will subscribe the Bluetooth Endpoint, the Nymi DLL, and WebSocket connections to the `Nymi WebAPI` by using the source IP of the connection. Therefore, if the Bluetooth Endpoint and application that is using the `Nymi WebAPI` are on the same host the application will work on connection.

For deployments in an RDP/Citrix environment or when the MES application (NEA) resides on a different host (such as a web or application server), the IP address of the client that runs the NEA is different from the IP address of the workstation. In these cases, ensure that the MES application can determine the IP address of the client workstation that runs the `Nymi Bluetooth Endpoint`.

- In remote desktop sessions, the IP address is usually available through Windows Terminal Services APIs.
- If you are not using RDP or Citrix, the IP address is usually available through vendor-specific environments or APIs.
- For remote applications, such as web-based application, you can determine the IP address by using the source IP address of the client requests.

When the application determines the IP address of the client workstation, the application must use the **`subscribe`** operation to connect to the correct `Nymi Bluetooth Endpoint`. Keep in mind that multiple IP addresses on the user workstation or NAT between components can interfere with determining client IP addresses and should be taken into consideration during deployment of an application.

If users might move between two or more client workstations, they must terminate their session before switching to another workstation, or the application must take this into account and start a new **`subscribe`** operation after reconnection.

Installing Silently

Use the command line to install the `Nymi Bluetooth Endpoint` and the `Nymi Agent`.

Use the following command to install the `Nymi Bluetooth Endpoint` only.

```
".\Nymi Runtime Installer 5.8.x.y.exe" -q InstallAgent=0
```

Where *x.y* is the version number.

Use the following command to install the `Nymi Agent` only.

```
".\Nymi Runtime Installer 5.8.x.y.exe" -q InstallEndpoint=0
```

Where *x.y* is the version number.

Nymi WebAPI Configuration Overview

Review the following requirements for the `Nymi WebAPI` and `Nymi Agent` components:

- Provide access to a distinct port for each component, port numbers are described later in this document.
- Configure transport layer security: on the server or by offloading.
- Ensure that both components have connectivity to NES.
- Ensure that there is no Network Address Translation (NAT) between the `Nymi WebAPI` of the `Nymi Agent` and the user terminals.
- When you use a centralized `Nymi Agent` on the same server as NES, ensure that each component can co-locate with the NES (ensure that you use distinct TCP ports).

SDK Package

The SDK package contains the following files

- ..*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.

Managing Security and Certificates

System Administrators need to obtain and import TLS key pairs that include certificates for your environment and for the `Nymi WebAPI`.

To ensure that trusted communication is configured and TLS is used by the WebSocket, obtain and install the following certificates:

- TLS Server Certificate chain including any intermediate CA certificates in base64 PEM format (note that the host certificate cannot be a wildcard certificate)
- Private key corresponding to the TLS server certificate, in base64 PEM format (the private key cannot be encrypted)
- Certificate of the root Certificate Authority (CA) issuing the TLS server certificate, in base64 PEM format

You must install these three PEM files at the location that is specified in the *nymi_agent.toml* file, to ensure that the `Nymi Agent` can communicate with the Nymi-enabled Application.

The `Nymi WebAPI` needs to connect to NES over HTTPS. The NES TLS server certificate must be issued by a Root CA trusted by the `Nymi WebAPI`. If the Root CA is not trusted by the workstation install the root CA certificate in the Trusted Root Certification Authorities container for the local machine. See Microsoft documentation for information about installing Trust Root Certificates: https://docs.microsoft.com/en-us/skype-sdk/sdn/articles/installing-the-trusted-root-certificate.html

For example, you can install the certificates in the certificate keystore at the following location: *C:\Windows\SysWOW64\config\systemprofile\AppData\Roaming\Nymi\NSL*

Note: For additional information about importing certificates, see the Nymi Connected Worker Platform NES Deployment Guide.

Note: Ensure that you have configured the certificate settings in the *nymi_agent.toml* configuration file. For more information see the, *Configuration Overview* section.

Importing the Root CA certificate

Perform the following steps only if the Root CA issuing the NES TLS server certificate is not a Trusted Root CA (for example, if a self-signed TLS server certificate is used for NES). Install the Root CA on each user terminal to support the establishment of a connection with the NES host.

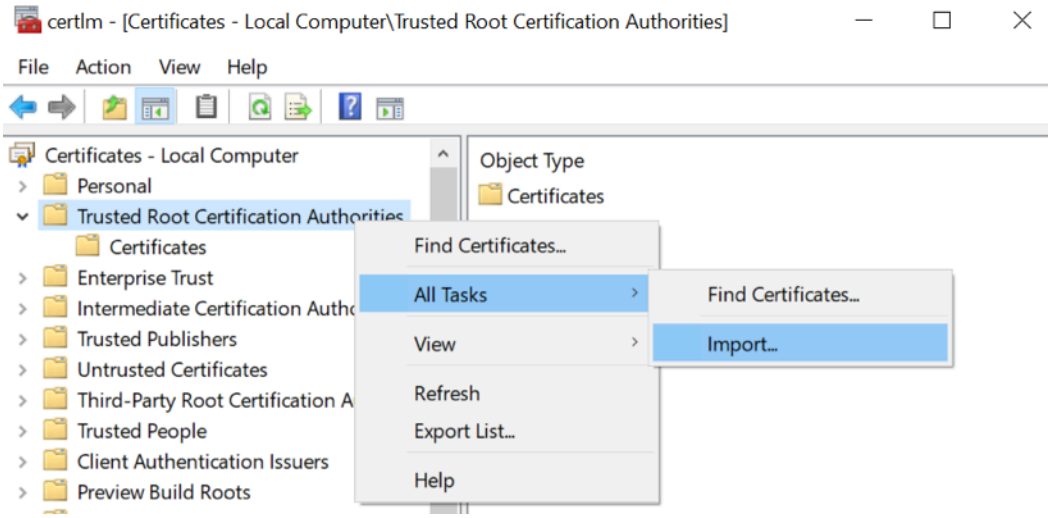
About this task

While logged into the user terminal as a local administrator, use the `certlm` application to import the root CA certificate into the Trusted Root Certification Authorities store. For example, on Windows 10, perform the following steps:

Procedure

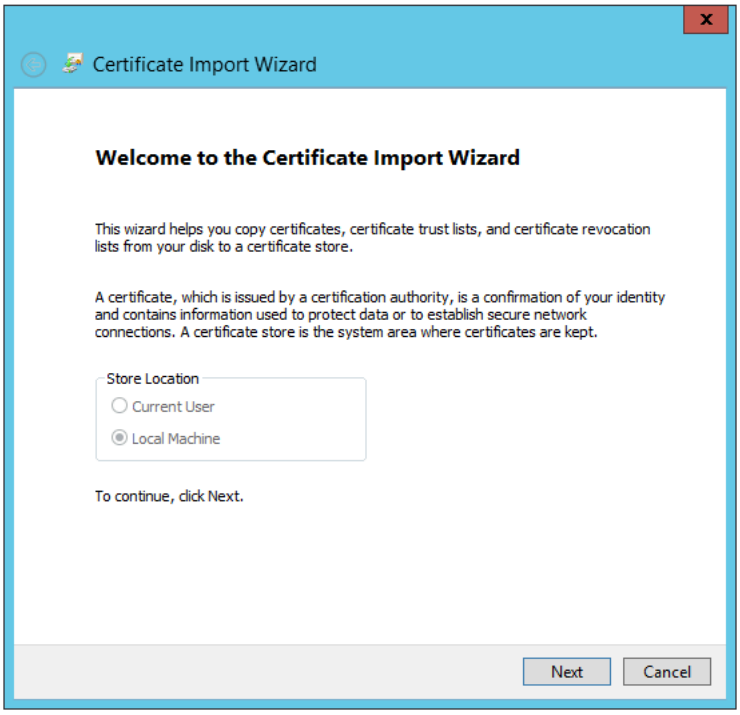
1. In `Control Panel`, select **`Manage Computer Certificates`**.

2. In the `certlm` window, right-click **`Trusted Root Certification Authorities`**, and then select **`All Tasks > Import`**.
   The following figure shows the `certlm` window.



`Figure 2: certlm application on Windows 10`

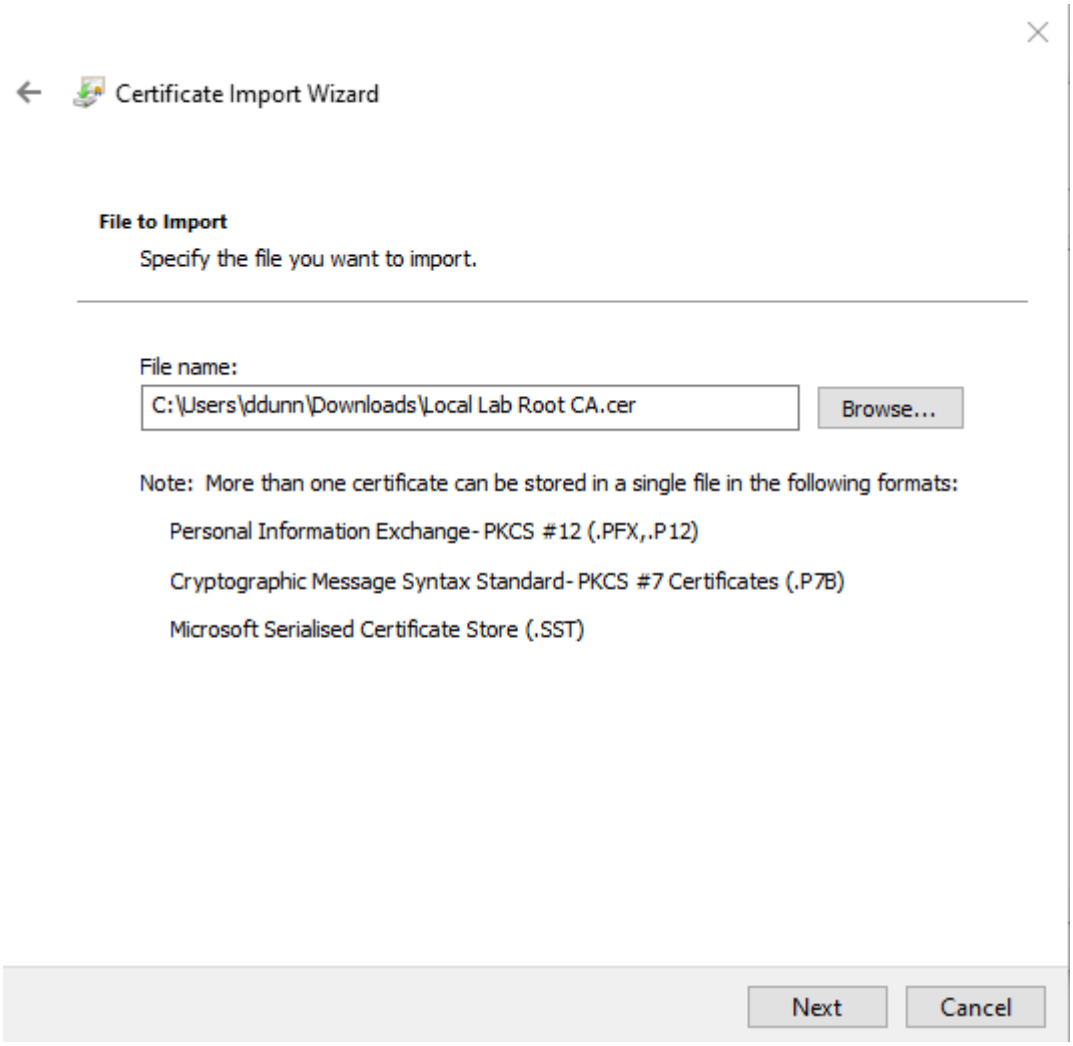
3. On the `Welcome to the Certificate Import Wizard` screen, click **`Next`**.
   The following figure shows the `Welcome to the Certificate Import Wizard` screen.



`Figure 3: Welcome to the Certificate Import Wizard screen`

4. On the `File to Import` screen, click **`Browse`**, navigate to the folder that contains the root certificate file, select the file, and then click **`Open`**.

5. On the `File to Import` screen, click **`Next`**.
   The following figure shows the `File to Import` screen.



`Figure 4: File to Import screen`

6. On the `Certificate Store` screen, accept the default value **`Place all certificates in the following store`** with the value **`Trusted Root Certification Authorities`**, and then click **`Next`**.
7. On the `Completing the Certificate Import Wizard` screen, click **`Finish`**.

Installing the Nymi Runtime

Perform the following steps to install `Nymi Runtime`.

Procedure

1. Extract the `Nymi SDK` package to the development machine.

2. To develop a native NEA using WebAPI, with c#, C, or C++ , copy the *nymi_api.dll* file from the .. *nymi-sdk**windows**x86_64* directory to the Visual Studio working directory.

   Note: In a remote environment where the NEA is running on a different machine than the runtime, Visual c++ 2013 and 2015 redistributables must be installed.

3. From the ..*nymi-sdk**windows**setup* folder, run the *Nymi Runtime Installer 5.8.x.y.exe* file.

   Where *x.y* is the version number.

   Note: In a physical environment, when you install the `Nymi Runtime`, accept all the defaults. For a virtual environment, install the `Nymi Bluetooth Endpoint` component only on the development machine. In a virtual environment, install the `Nymi Runtime` on the machine designated as the centralized `Nymi Agent`, and only install the `Nymi Agent` component.

Configuring the Nymi Agent toml File

Configuration settings for the `Nymi WebAPI` are contained in a toml file. If the `Nymi WebAPI` is not configured (the toml file does not exist), the `Nymi Agent` runs with the `Nymi WebAPI` disabled by default.

A sample configuration file named *nymi_agent_default.toml* file is included in the `Nymi SDK` package and is installed by default in the *C:\Nymi\NymiAgent* directory. In order to enable the WebSocket simply make a copy of the file and name it *nymi_agent.toml*.

When you edit the *nymi_agent.toml* to configure `Nymi Agent` with `WebAPI`, ensure that:

- Each interface uses a distinct TCP port (do not use 9120/tcp)
- Machine has connectivity to NES
- Network Address Translation (NAT) does not exist between the `Nymi Agent` machine and the user terminals.
- `Nymi Agent` can co-locate with the NES (ensure that you use distinct TCP ports)

General Application Settings

These settings are application-wide and enable you to define the logging level.

- error: log only errors
- warn: log both errors and warnings
- info: log errors, warnings, and activity
- debug: log everything including debugging information

The following is the default setting: *log_level = "warn"*

Enterprise Server Settings

The *NES* section defines settings that affect the embedded Nymi-enabled Application (NEA), which is the basis `Nymi WebAPI`.

Table 1: Enterprise Service

Description	Setting
Sets the NEA name for the `WebAPI` server application.	*`nea_name = "NymiAgent"`*
The host URL for the NES server. Include only the protocol and hostname portion of the URI. For example, replace `https://server.name.local` with your nes_url.	*`nes_url = "https://server.name.local.com"`* For example, https://myserver.name.local.com
'The directory service name for NES. For example if your NES URL is https://server.name.local.com/NES, the directory name is NES.	`directory_service_id = "NES"`
The TLS client root CA certificate bundle to use when communicating with NES. If it is not specified a built-in bundle containing well known root CAs is used. Specify the path to a CA certificate or bundle to customize the trusted CA bundle using the PEM format.	Certificate bundle in PEM format for the signing CA certificate chains for the TLS certificates that are used by NES and `WebAPI`.
cacertfile = "cacertfile.pem"	

Note: An NEA expects a Nymi Bluetooth Endpoint with an endpoint name that is the based on the network interface address used to communicate with the agent. If the NEA and `Nymi Bluetooth Endpoint` connect to the `Nymi Agent` on different network interfaces, the Nymi-enabled Application will not see the endpoint and will report it as missing with a status code of 5100. Nymi recommends that additional interfaces be disabled or to set a static well-known endpoint ID in the *nbe.toml* of the endpoint terminal and supply the endpoint ID to the Nymi-enabled Application to manually subscribe after connecting to the Nymi Agent.

on the network interface address that is used to communicate with the `Nymi Agent`. If the NEA and `Nymi Bluetooth Endpoint` connect to the `Nymi Agent` on different network interfaces, the Nymi-enabled Application does not find the endpoint and reports that the endpoint is missing (status code 5100).

For local environments where the `Nymi Bluetooth Endpoint` and the NEA runs on the same computer, and the computer has more than one interface that can contact the `Nymi Agent`, use the computer as the topic name for the `Nymi Bluetooth Endpoint` and the NEA to communicate.

For Citrix/RDP scenarios, use the client computer name as the topic name to support communications between the `Nymi Bluetooth Endpoint` that runs on the client computer and the NEA that runs on the Citrix/RDP host. Ensure that the NEA subscribes to the topic by using the client name of the session.

WebAPI Protocol Settings

The following table provides settings used to set the WebAPI application.

Table 2: WebAPI Protocol Settings

Description	Example
Specify the protocol supported by the `Nymi WebAPI` server. Use the default protocol *ws* to support a plain WebSocket using `ws://...` URL scheme. Use `wss` to support a secure WebSocket using TLS and the `wss://...` URL scheme. Set the protocol to *wss* in production environments.	*protocol = "wss"*
The server port to listen for `Nymi WebAPI` client WebSocket connections on. The default depends on the `protocol` settings. For the *ws* protocol the default port is 8080. For the *wss* protocol the default port is 4443. You can set an alternate port using this setting.	*port = 4443 port = 8080*
Certificate bundle in PEM format for the signing CA certificate chains for the TLS certificates that are used by NES and `WebAPI`.	`cacertfile = "/path/to/certfile.pem"`
The path to the server certificate in PEM format.	*certfile = "/path/to/certfile.pem"*
The path to the server certificate private key in PEM format.	*keyfile = "/path/to/keyfile.pem"*

Note: The `Nymi Agent` must be able to receive incoming WebSocket connections on TCP port 9120 (used for communication with `Nymi Bluetooth Endpoint`) and on the TCP port configured for `Nymi WebAPI` connections (default 8080 when using the ws protocol, and default 4443 when using the wss protocol). Ensure that these ports are open bi-directionally in the firewall on the machine that runs the `Nymi Agent`.

Configuring the Nymi Bluetooth Endpoint toml File

In order to create an environment that can also utilize the services contained in the `Nymi API C Interface`, specify the location of the `Nymi Agent` so that the `Nymi Bluetooth Endpoint` can connect to it.

Procedure

Install the `Nymi Bluetooth Endpoint` on each user terminal. The `Nymi Bluetooth Endpoint` service on each user terminal communicates with the `Nymi Agent`, through websocket port 9120.

Perform the following steps on each user terminal.

1. Edit the *C:\Nymi\Bluetooth_Endpoint\nbe.toml* file.
2. Update the location of the `Nymi Agent`
   - agent_url = 'ws://<FQDN>:9120/socket/websocket'

3. Optionally, change the location of the `Nymi Bluetooth Endpoint` by configuring the *endpoint_id* parameter.

   - endpoint_id = "<unique ID>"

   By configuring the *endpoint_id* parameter, you need to use the **`subscribe`** operation in the NEA to subscribe to the defined endpoint. For more information about the subscribe operation, see the *Request Operation* section in this guide.

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 written in JSON. 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.

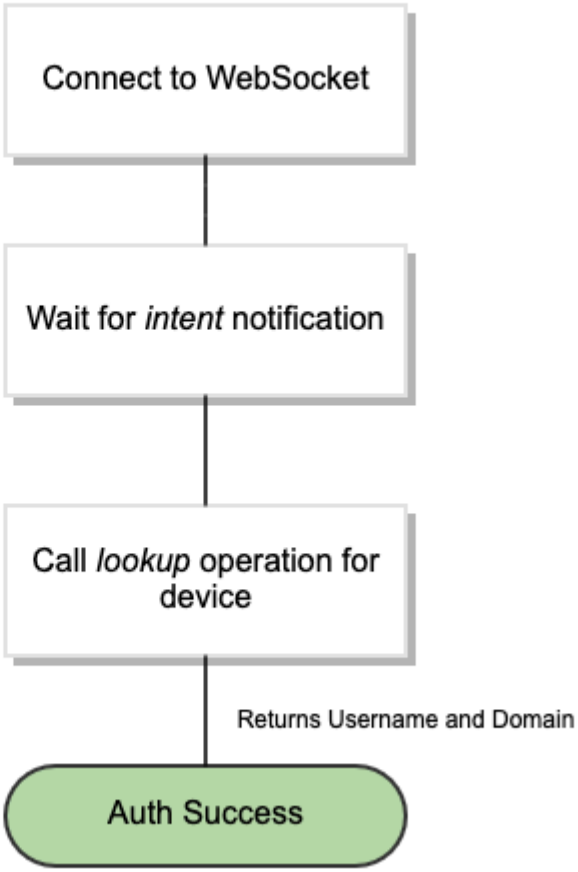


Figure 5: NFC-only communications

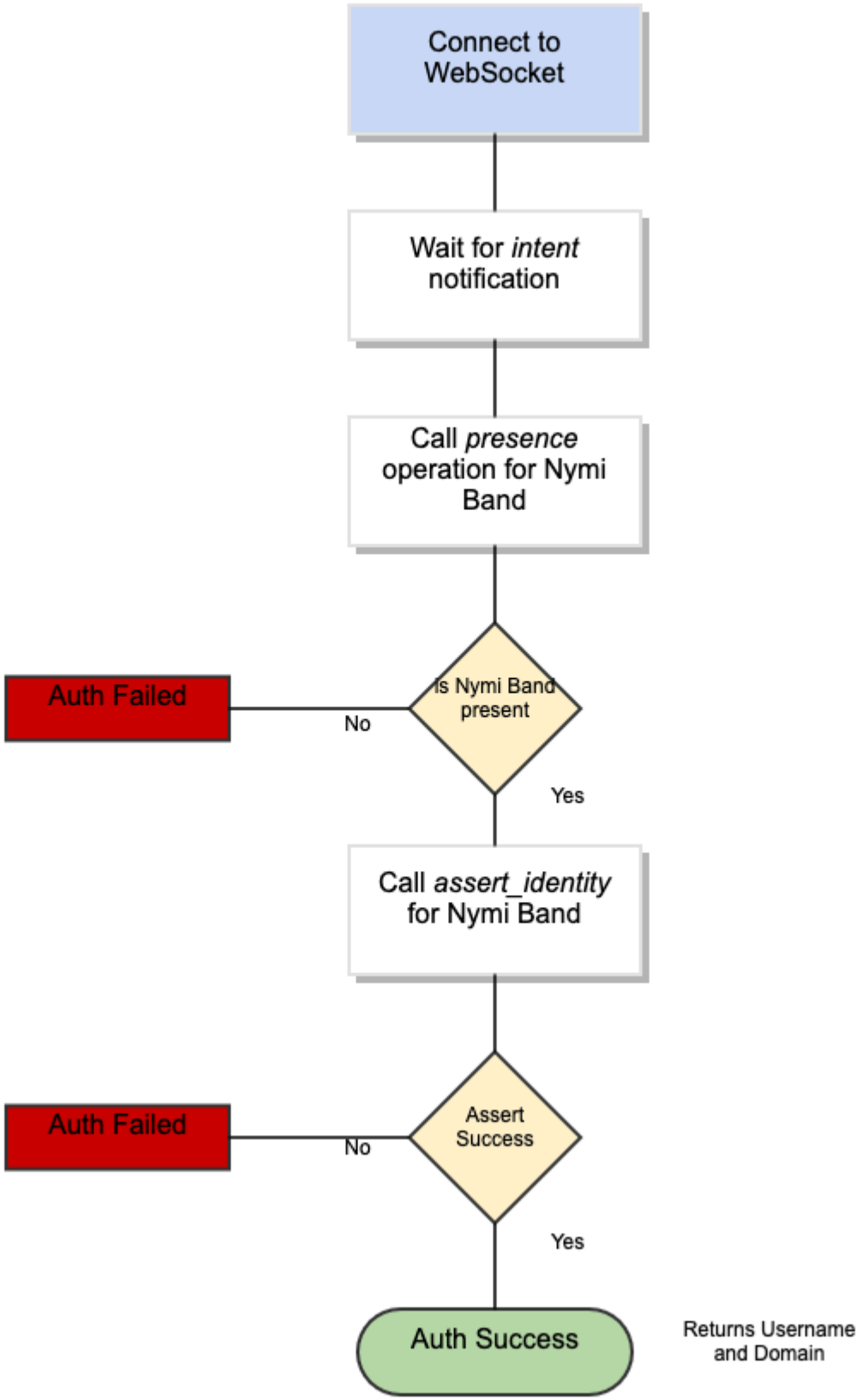


Figure 6: NFC with Bluetooth communications

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 user is not present.
- Unauthenticated - The user 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*.

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.

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.
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.

assert_identity operation

The *assert_identity* operation provides an NEA with the ability to confirm that a Nymi Band that is assigned to a specific user is authenticated and within Bluetooth range.

The *assert_identity* operation completes a cryptographic handshake with the Nymi Band and verifies user/band identity.

Note: The Nymi Band must be in an authenticated state when you call the *assert_identity* operation.

Define the *assert_identity* JSON object in the following format.

```
{
   "operation": "assert_identity",
   "exchange": "exchange_value",
   "payload": {
      "nes_url": "https_url_to_nes",
      "device": "NymiBandID",
      "assert_type": "assert_user"
    }
 }
```

where:

- *nes_url* field is optional if not provided it uses what is configured for the Nymi Agent. See the *Configuration Overview*.
- *NymiBandID* is the Nymi Band (or device) ID value that is returned in the *lookup* result.

Example

The following code block provides an example of a JSON object that instructs `Nymi WebAPI` to assert the identity of the user with device ID *C2:FA:D7:F0:D7:96*.

```
{
  "operation": "assert_identity",
  "exchange": "rAndOm_IdeNtifyiNG_StrING_5555",
  "payload": {
      "nes_url": "http://nes.nymi.com/nes/",
      "device": "C2:FA:D7:F0:D7:96",
      "assert_type": " assert_user "
  }
}
```

assert_identity response

The assert_identity request returns *Username* and *Domain.* properties

assert_identity Results

The *UserStatus* property is an optional property. The UserStatus is stored in the Active Directory (AD).

If the UserStatus option is set in the NES console in the *Policies* > *Active Directory* page, the Active Directory status appears in the assert_identity response. If the option is not set, it does not return in the response.

The *UserStatus* option has the following possible values:

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 Password Expired.
Active|PasswordExpired	User account has an expired password. This status can appear with Locked.

The last three properties can be combined into a comma separated list.

By default, NES disables support for 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`.

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

```
{
```

```
"operation": "assert_identity",
"exchange":"rAndOm_IdeNtifyiNG_StrING_5555",
"payload": {
"Username": "Jsmith",
"Domain": "Corp"
"UserStatus": "Active"

},
"status": "0",
"error: {}
}
```

lookup

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 3: 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"]
   }
}
```

Results 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"]
  }
}
```

Results 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: {}
}
```

subscribe 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": "bar"
  }
}
```

where:

- *operation* is *subscribe_endpoint*.
- *exchange* is any value and is used to match the response to the request.

payload:

- *endpoint_id* is based on the endpoint IP address. Required when the configuration uses a centralized `Nymi Agent`.

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.

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 4: 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.

Nymi Inc. (Nymi) believes the information in this document is accurate as of its publication date. The information is subject to change without notice.

The information in this document is provided as-is and Nymi makes no representations or warranties of any kind. This document does not provide you with any legal rights to any intellectual property in any Nymi product. You may copy and use this document for your referential purposes.

This software or hardware is developed for general use in a variety of industries and Nymi assumes no liability as a result of their use or application.Nymi, Nymi Band, and other trademarks are the property of Nymi Inc. Other trademarks may be the property of their respective owners.

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