Ayyeka REST API

v1.0

September 2017

Contents

Introduction

Ayyeka provides programmatic access to your Ayyeka Wavelets' information using a simple and secure REST API. To use the Ayyeka API, you should have a basic familiarity with software development, RESTful web services, and Ayyeka's User Interface (UI).

Overview

Ayyeka Wavelet System Overview

Ayyeka provides end-to-end remote monitoring solutions consisting of a sensor or multiple sensors, a Wavelet data acquisition system, wireless connectivity, as well as data and device management. These solutions are provided to customers as Wavelet Kits for various applications. Wavelet Kits provide users with continuous data on the state of their infrastructure or environment.

The Wavelet device obtains samples from the connected sensors at a configurable sampling frequency. The data obtained from the sensors is logged on the device and transmitted to a server at a configurable transmission frequency.

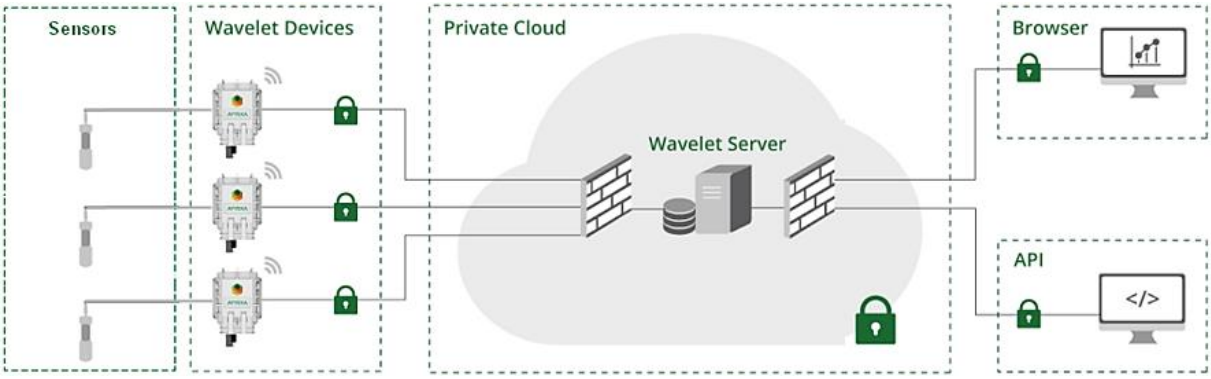

Figure 1. Ayyeka Wavelet System

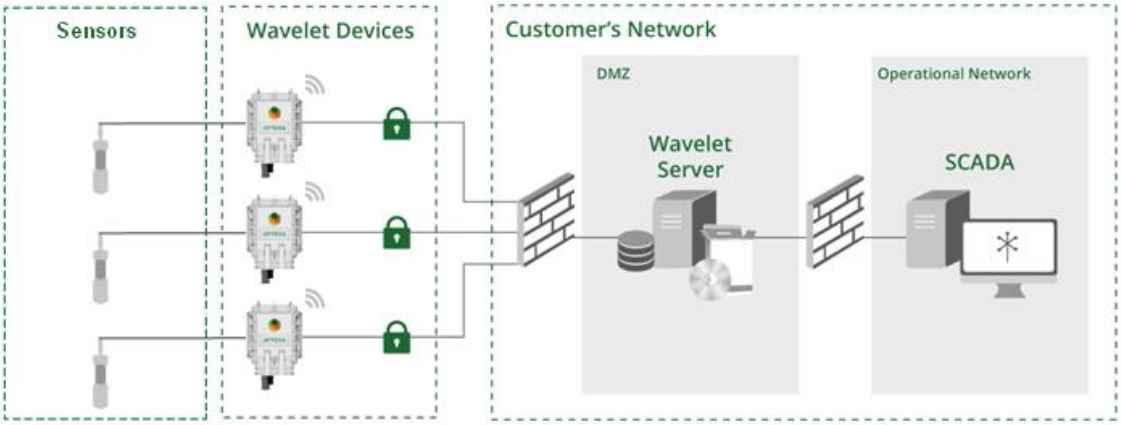

Figure 2. Ayyeka Wavelet On-Premises System

Device Management and Monitoring

Wavelet devices can be managed and monitored in the Ayyeka Management UI and data consumed both through the UI and Ayyeka REST API.

Ayyeka Management User Interface

The Ayyeka Management UI provides the following functionalities:

- Visualizing stream (i.e. parameter) data
- Generating reports
- Defining threshold levels for email and/or SMS alerts for various parameters
- Over-the-air remote device configuration

API System Overview

Ayyeka Data Model

Sensor measurements are stored in Ayyeka's database in a hierarchical structure that matches the actual deployment of the Ayyeka devices and their sensors. For each customer account, the Ayyeka data model includes the following entities:

- **Site** – A logical entity representing a Wavelet installation site. A site contains one or more devices.
- Device – Represents a physical Wavelet device installed in the field. Each device belongs to a specific Site.
- **Sample** – Represents a measurement at a specific moment in time. Each sample belongs to a specific Stream.
- **Stream** – A logical entity representing a sequence of measurements from a specific sensor parameter. Each stream is unique in the Ayyeka system.

Each Site, Device, Stream and Sample has a unique ID that identifies it and is used across the API methods.

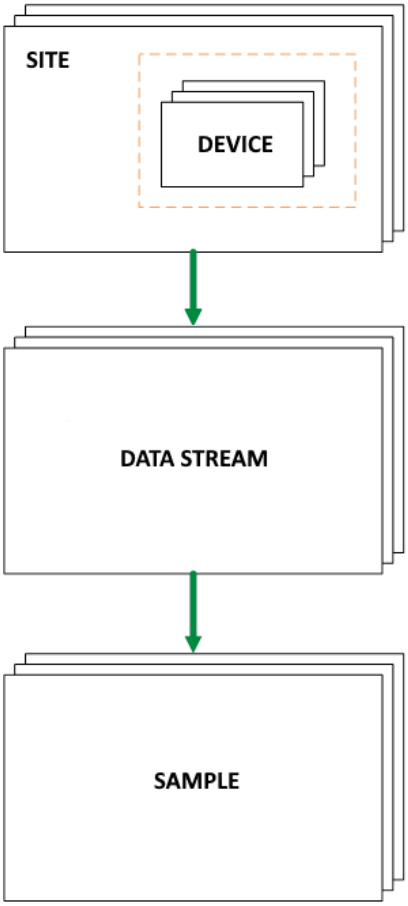

Figure 3. Ayyeka Data Model Diagram

Device Transmission Interval

Every device has three configurable transmission intervals: Normal, Event and Emergency. Normal is the default interval. Event and Emergency are two alternate intervals. Data is always transmitted at the Normal rate, unless a different rate is specified – either Event or Emergency – when stream samples fall within a certain configurable range of values (called thresholds).

Stream Thresholds

A major feature of the Ayyeka system is the ability to set multiple thresholds for any data stream, and specify which actions to take when stream samples fall within each threshold. A threshold is defined as a range of values. For example, for a stream that samples temperatures, you can define a Low threshold of 0-32 degrees, a Normal threshold of 32-74 degrees, and a High threshold of 78-104 degrees Fahrenheit.

For each threshold, you can define the following when stream samples fall within the threshold range:

- The device transmission interval – Whether to transmit data from this data stream at the Normal, Event or Emergency rate
- The device sampling interval – How often the device should sample this data stream.
- Sensor actions – Instruct the device to take one of the following actions:
  - Pause – Instructs the device to pause stream sampling.
  - Resume – Instructs the device to resume stream sampling.
  - Single – Instructs the device to take a single sample immediately and then resume its regular sampling schedule.
  - Pause Single – Instructs the device to take one sample immediately, and then pause sampling.

User Groups

A user's group defines which threshold alerts each user that is a member of it will receive. The full process includes the following actions:

1. On the one hand, create a user group, and assign users to the group.
2. On the other hand, set data stream thresholds and define which groups will receive alerts when stream samples fall within a threshold range.

Using the API

Authorization Using OAuth 2.0

Ayyeka provides the industry-standard OAuth 2.0 authorization protocol to allow users to authorize API client access to Ayyeka data via the REST API. This enables Ayyeka users to grant access to an API client without having to share their username and password. Users can view and revoke the API client they authorized in their account settings.

Ayyeka currently only supports the client credentials grant type.

For more information about OAuth 2.0, refer to http://tools.ietf.org/html/rfc6749.

Terminology

Term	Definition
Authorization Server:	Entity that protects data and validates credentials before authorizing an API Client to take any action on behalf of an end user.
API Client:	Entity that represents your application and allows use of OAuth2 for authentication.
Client Credentials:	API key and secret for the API Client.
Access Token:	Token provided by the Authorization Server to the client application to authorize access to data.

The process of gaining access to Ayyeka data via the REST API includes the following steps:

1. Register your application as an API client (within the Ayyeka UI from the API tab)
2. Get an access token
3. Authenticate with the REST API using the provided access token

Step 1: Creating an API Client

The first step is to create an API Client in the Ayyeka UI as follows:

1. In the Ayyeka UI, click on the API tab, then select the API Clients sub-tab.
2. Click on the Add API Client button.
3. Select REST API from the drop-down menu for Type. Type a comment, and then press submit.
4. An API Client Key and Secret will be generated for you to download. Note that this is the only time you can get the API Client Secret.
5. Store the API Key and Secret in your application in a secure place.

Step 2: Getting an Access Token

To make a REST API call, you must include request headers including the Authorization header with an OAuth 2.0 access token. To get an access token, pass the [ApiClientKey]:[ApiClientSecret] credentials to the Authorization Server in base64 format in the Authorization header in a get access token request.

The following figure illustrates the process:

```
 +---------+                                  +---------------+
 :         :                                  :               :
 :         :>-- A - Client Authentication --->: Authorization :
 : Client  :                                  :     Server    :
 :         :<-- B ---- Access Token ---------<:               :
 :         :                                  :               :
 +---------+                                  +---------------+
```

Authorization Server Auth URL: https://restapi.ayyeka.com/auth/token

Example request:

APIClient Key	APIClient Secret
149F9AC689CB4F16853433AA0AE44E5F	ePs4G352BUD9FYCvPXEoz2RC5UYx/K8BoVs+mkrVfpg=

```
POST /auth/token HTTP/1.1

Host: restapi.ayyeka.com

Content-Type: application/x-www-form-urlencoded

Authorization: Basic
MTQ5RjlBQzY4OUNCNEYxNjg1MzQzM0FBMEFFNDRFNUY6ZVBzNEczNTJCVUQ5RllDdlBYRW9
6MlJDNVVZeC9LOEJvVnMrbWtyVmZwZz0=

Cache-Control: no-cache



grant_type=client_credentials
```

Term	Definition
grant_type	This must be client_credentials.
client_id	Your application's API Client Key.
client_secret	Your application's API Client Secret

Example response:

```
HTTP/1.1 200 OK

Cache-Control: no-store

Content-Type: application/json; charset=utf-8

Date: Mon, 26 Jun 2017 14:36:08 GMT

Pragma: no-cache

Content-Length: 231

Connection: keep-alive

{"access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiIyIiwiZXhwIjoxNDk4N
DkxMzY4LCJpYXQiOjE0OTg0ODc3NjgsImlzcyI6IkNsb3VkIiwic3ViIjoiMjc...",

   "token_type":"JWT","expires_in":3600}
```

Step 3: Authentication with Access Token

Using the access token provided in the authorization response, the API client can now access the REST API on behalf of the authorizing user as follows:

- Use a header in the format Authorization: Bearer [token].
- Your application should check for 403 errors in case the user has revoked application access or the token has expired.

Example request:

```
GET /v1.0/site HTTP/1.1

Host: restapi.ayyeka.com

Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiIyIiwiZXhwIjoxNDk4NDkxMzY4LCJpYXQiOjE
0OTg0ODc3NjgsImlzcyI6IkNsb3VkIiwic3ViIjoiMjc...

Cache-Control: no-cache
```

Example response:

```
HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8



[{"id":1,"creationDate":"2013-05-31T01:11:41Z","displayName":"Test
Site","status":"Active","accountOrganizationId":2}]
```

API Reference

Ayyeka provides a RESTful API for exploring sites and streams and retrieving data samples. See *Ayyeka REST API*.

The following operations are supported.

- Device Methods:
  - *getDeviceBySerialNumber*

    Returns the device corresponding to the specified serial number
- Site methods:
  - *getAllSites*

    Returns all sites to which the client has access
  - *getDevicesBySite*

    Returns all devices from a site with a specified ID
  - *getSiteById*

    Returns information about the site with a specified ID
  - *getStreamsBySite*

    Returns all streams from a site with a specified ID
- Stream methods
  - *getLastSampleByStream*

    Returns the last sample from the stream specified by ID
  - *getSamplesByStream*

    Returns up to 100 samples from the stream specified by ID
  - *getStreamById*

    Returns a stream with the specified ID