

InfinID Technologies, Inc.

**AssetWorx API
Quickstart Guide**

June 3rd, 2020

Table of Contents

1.0 AssetWorx Overview	3
1.1 Software Overview	3
1.2 AssetWorx API Overview	3
1.3 Quickstart Guide Steps	3
2.1 API Overview	4
3.0 Setup	5
3.1 Software Setup	5
3.2 Hardware Setup	5
4.0 Basic API Calls	6
4.1 Common Endpoints	6
5. Issuing V-Tag Commands	7
5.1 Set Threshold	8
5.2 Set Accel Sensor	9
5.3 Get Position	10
5.4 Set Position	11
5.5 Reset Tag	12
5.6 Dwell Time	13
5.7 Activate	14
5.8 Buzz	15
6. AssetWorx Client Layer	16
6.1 Client Layer Structure	16
6.2 Instantiating client instances	16

1.0 AssetWorx Overview

1.1 Software Overview

AssetWorx is an RFID based asset management application. It works with barcodes, passive RFID, as well as InfinID Technologies' mesh network V-Tag technology.

1.2 AssetWorx API Overview

The AssetWorx API has been designed to make it easy to tie into the AssetWorx data via a well structured RESTFUL API.

1.3 Quickstart Guide Steps

- API Overview – Please see chapter 2
- Setting up software and hardware – Please see chapter 3
- Basic API calls – Please see chapter 4

2.1 API Overview

The AssetWorx API is a RESTFUL API. This means that interfacing with the API is done by structuring HTTP verbs such as POST, PUT, GET, and DELETE.

The complete API Swagger documentation can be found online:

<https://swaggerhub.com/apis/InfinID-Technologies/AssetWorx>

To get started with testing out the APIs, you can use the Swagger definitions to generate a client of your choosing.

AssetWorx provides its own authentication server, but to help facilitate Single Sign-On, AssetWorx can be configured to accept tokens from an external authentication server. Authenticating with the AssetWorx Authentication Server or an External Authentication Server will provide an encrypted access token as well as an expiration time in minutes. The token should be passed with every subsequent API call to the AssetWorx resource server.

Please see the [Swagger Definitions](#) for more details on getting a token.

3.0 Setup

3.1 Software Setup

The first step is to install the AssetWorx software applications that are needed. The web application will always be needed but to integrate with RFID hardware such as passive readers and V-Tag Gateways, the Alarm Monitoring Service will also need to be installed. Please refer to the [AssetWorx Software Installation Instructions](#) for more details.

3.2 Hardware Setup

After software has been setup, the next step is to install any hardware that is needed. Hardware devices usually consist of one of the following:

- **Passive RFID Portals** – These are portals which only read passive RFID tags. They are useful for setting up choke points throughout the enterprise.
- **Handheld Scanners** – Handheld scanners are useful for doing periodic inventory of assets that have been marked with passive RFID tags.
- **V-Tag Gateways** – V-Tag Gateways facilitate communication with InfinID Technology mesh network V-Tags.

4.0 Basic API Calls

The API is organized into a collection of resources that expose a common set of endpoints as well as additional resource specific endpoints.

4.1 Common Endpoints

Every Resource exposes endpoints for inserting, updating, deleting as well as querying data:

GET	/api/asset/{id}	Get by ID	🔒
GET	/api/asset/MostRecent/	Get most recent	🔒
POST	/api/asset/	Post Asset	🔒
PUT	/api/asset/	Update Asset	🔒
DELETE	/api/asset/	Delete Asset	🔒
GET	/api/assets/	Get all	🔒
GET	/api/assets/After/{id}	Get any after id	🔒
GET	/api/assets/Paged/	Get paged entities	🔒
GET	/api/assets/Count/	Get count entities	🔒

Some resources expose additional endpoints which expose resource specific actions. To see the complete list of API calls available, please refer to the [Swagger Definitions](#) .

5. Issuing V-Tag Commands

The following tag commands are available:

- Set Threshold
- Set Accel Sensor
- Get Position
- Set Position
- Reset Tag
- Set Dwell Time
- Activate
- Buzz

Tag Command Timeouts: If there is no response from the tag within 20 minutes then the command will fail with a failure message of “Timeout.” This may be due to an invalid tag number, a depleted battery in the tag, or the tag being out of range of communications.

5.1 Set Threshold

Purpose: A tag may be configured with thresholds for temperature, acceleration, or battery level. If the threshold is exceeded, an immediate alarm is generated.

To disable a threshold, set the duration to 0 in the set threshold command.

If a threshold value is exceeded and an alarm reported, no new alarm indication will be issued until the sensor measurement has dropped back below the threshold value for a period of 300 seconds.

By default, there is a low battery threshold of 2.7V configured for each tag when it arrives from the factory. Tags generally will fail when the battery voltage drops below 2.5V.

/api/queuedcommand/setThreshold/

POST /api/queuedcommand/setThreshold/

SetThreshold

Summary

Posts a SetThreshold command

Description

Posts a SetThreshold command to define min or max threshold parameters for a V-Tag

Parameters

Name	Located in	Description	Required	Schema
tagID	query	V-Tag ID	Yes	↔ string
thresholdType	query	Temperature Upper, Temperature Lower, Acceleration Upper (g), Battery	Yes	↔ string
threshold	query	threshold value	Yes	↔ number
duration	query	Duration in milliseconds	Yes	↔ integer

Responses

Code	Description	Schema
200	successful operation	↔ integer
401	If not authorized	

Security

Security Schema	Scopes
Bearer	

Try this operation

5.2 Set Accel Sensor

Purpose: The acceleration sensor is checked 12 times per second to check whether the tag is moving. By configuring the tag to ignore the acceleration sensor, you can add a month or two to the battery life. Note that the tag movement notifications are used by the location estimation algorithms as outlined in Chapter 3, so you should not disable the acceleration sensor if you need location estimates.

/api/queuedcommand/setAccelSensor/



POST /api/queuedcommand/setAccelSensor/

SetAccelSensor

Summary

Posts a SetAccelSensor command

Description

Posts a SetAccelSensor command to return the current X,Y, and Z values.

Parameters

Name	Located in	Description	Required	Schema
tagID	query	V-Tag ID	Yes	↔ string
enabled	query	true if enabled false if disabled	Yes	↔ string

Responses

Code	Description	Schema
200	successful operation	↔ integer
401	If not authorized	

Security

Security Schema	Scopes
Bearer	

Try this operation

5.3 Get Position

Purpose: Query a tag for its position (X, Y, Z). After returning, the asset record will automatically be updated.

/api/queuedcommand/getPosition/



POST /api/queuedcommand/getPosition/

GetPosition

Summary

Posts a GetPosition command

Description

Posts a GetPosition command to return the current X,Y, and Z values.

Parameters

Name	Located in	Description	Required	Schema
tagID	query	V-Tag ID	Yes	↔ string

Responses

Code	Description	Schema
200	successful operation	↔ integer
401	If not authorized	

Security

Security Schema	Scopes
Bearer	

Try this operation

5.4 Set Position

Purpose: Set the position for a fixed tag.

Setting the tag position will only work for a fixed reference tag. All other tags determine their X,Y, and Z automatically and will normally have a position type of estimated, unless they have not discovered enough neighbors to allow them to perform location calculations, in which case their position will be unknown.

X	-10,000 to +10,000 (meters)
Y	-10,000 to +10,000 (meters)
Z	-127 to +127 (floor number)

Table 7-1 Coordinate Ranges for X, Y, Z

/api/queuedcommand/setPosition/

POST /api/queuedcommand/setPosition/ setPosition

Summary

Posts a SetPosition command

Description

Posts a SetPosition command for a fixed V-Tag

Parameters

Name	Located in	Description	Required	Schema
tagID	query	V-Tag ID	Yes	↔ string
x	query	X position in meters	Yes	↔ number
y	query	Y position in meters	Yes	↔ number
z	query	Z level	Yes	↔ integer

Responses

Code	Description	Schema
200	successful operation	↔ integer
401	If not authorized	

Security

Security Schema	Scopes
Bearer	

Try this operation

5.5 Reset Tag

Purpose: Simulates an acceleration event for the tag and will cause the tag to issue minute movement reports. This is especially useful for testing a tag when it is already deployed in the field.

/api/queuedcommand/resetTag/

POST /api/queuedcommand/resetTag/

ResetTag

Summary

Posts a ResetTag command.

Description

Posts an ResetTag command which causes a tag to start rediscovering its neighbors and sending minute movement reports.

Parameters

Name	Located in	Description	Required	Schema
tagID	query	V-Tag ID	Yes	↔ string

Responses

Code	Description	Schema
200	successful operation	↔ integer
401	If not authorized	

Security

Security Schema	Scopes
Bearer	

Try this operation

5.6 Dwell Time

Purpose: When enabled, Dwell Time reverses the nature of movement reports for a tag and only starts reporting when it has held still for X minutes. Set to 0 to disable Dwell Time.

/api/queuedcommand/setDwellTime/

POST /api/queuedcommand/setDwellTime/

SetDwellTime

Summary

Posts a SetDwellTime command.

Description

Posts a SetDwellTime command. Dwell Time reverses the movement reports for a tag and only starts reporting when it has held still for X minutes. Set to 0 to disable.

Parameters

Name	Located in	Description	Required	Schema
tagID	query	V-Tag ID	Yes	↔ string
dwellTime	query	time in minutes	Yes	↔ integer

Responses

Code	Description	Schema
200	successful operation	↔ integer
401	If not authorized	

Security

Security Schema	Scopes
Bearer	

Try this operation

5.7 Activate

Purpose: Posts an Activate command which will cause a V-Tag to start sending tracking messages once a second for twenty minutes. These tracking messages can be read by the V-Tag Finder unit to zero in on a tags location.

/api/queuedcommand/activate/

POST /api/queuedcommand/activate/

Activate

Summary

Posts an Activate command.

Description

Posts an Activate command which will cause a V-Tag to start sending tracking messages once a second. These tracking messages can be read by the V-Tag Finder unit to zero in on a tags location.

Parameters

Name	Located in	Description	Required	Schema
tagID	query	V-Tag ID	Yes	↔ string

Responses

Code	Description	Schema
200	successful operation	↔ integer
401	If not authorized	

Security

Security Schema	Scopes
Bearer	

Try this operation

5.8 Buzz

Purpose: Posts an Buzz command which will cause a V-Tag to start beeping or flashing a light depending on the V-Tag hardware configuration

POST `/api/queuedcommand/buzz/` Posts a Buzz command.

Posts an Buzz command which causes a tag to start beeping or flashing a light depending on the V-Tag hardware configuration

Parameters Try it out

Name	Description
tagID <small>* required</small>	V-Tag ID
string (query)	<input type="text" value="tagID - V-Tag ID"/>

Responses Response content type **application/json**

Code	Description
200	successful operation
	Example Value Model
	<div>0</div>
401	If not authorized

6. AssetWorx Client Layer

As an alternative to creating your own code to call the web services, AssetWorx also provides a client layer which any .Net application can call. The client layer provides methods which will automatically structure the web service calls and monitor authentication token timeouts, exceptions, etc.

6.1 Client Layer Structure

To get started, locate the AssetWorx.Client.dll file in the API documentation folder and add a reference to it in your Visual Studio project. From the reference, you will find a client class for every API resource in AssetWorx. Every client implements the base repository interface:

```
public interface IRepository<T>
{
    Task<long> Insert(T t, string madeBy);
    Task<bool> Insert(T[] entities);
    Task Update(T t, string madeBy);
    Task Delete(long[] ids, string madeBy);
    Task<List<T>> GetPaged(string search, int page,
        int pageSize, string orderColumn, string orderDirection);
    Task<List<T>> GetAll(string search, string orderColumn, string orderDirection);
    Task<int> GetCount(string search);
    Task<T> Get(long id);
    Task<List<T>> GetAfter(long id);
    Task<T> GetMostRecent();
}
```

6.2 Instantiating client instances

Every client class constructor expects a ServiceOptions instance which defines the BaseURL, Username/Password, Client ID, Client Secret, Refresh Token. Different parameters are required depending on the OpenID Connect Grant Type that is desired. Each constructor also expects an AuthToken Singleton which the class will use to store the authentication token and timeout values that are received during authentication. Users of the AssetWorx client layer don't have to worry about authentication as these details are automatically handled by the client:

```
public class AssetClient : BaseClient<Asset>, IAssetRepository
{
    public AssetClient(ServiceOptions serviceOptions,
        AuthToken authToken) : base(serviceOptions, authToken){
    }
}
```