Reader WebUI User Guide

Version 8.0

http://www.impinj.com

Contents

1 Introduction

The primary focus of this document is the use of web UI in the management of events and device presets. The web UI can be accessed via the web browser and is useful for quick setup or troubleshooting. Web browsers we support are Chrome 96.x, Firefox 95.x, and Safari 15.x. This document will provide a brief overview of the interface and explain how to:

- Access the Impinj web UI
- Navigate the site
- Define RFID presets
- Use industry-standard tools and protocols to consume reader events.

Rather than an exhaustive description of each field, this document provides an introduction to the web UI, describes how to perform some basic operations and demonstrates how to configure Inventory operations using common parameters.

Note: To access specific information, click the help icon attached to the relevant field.

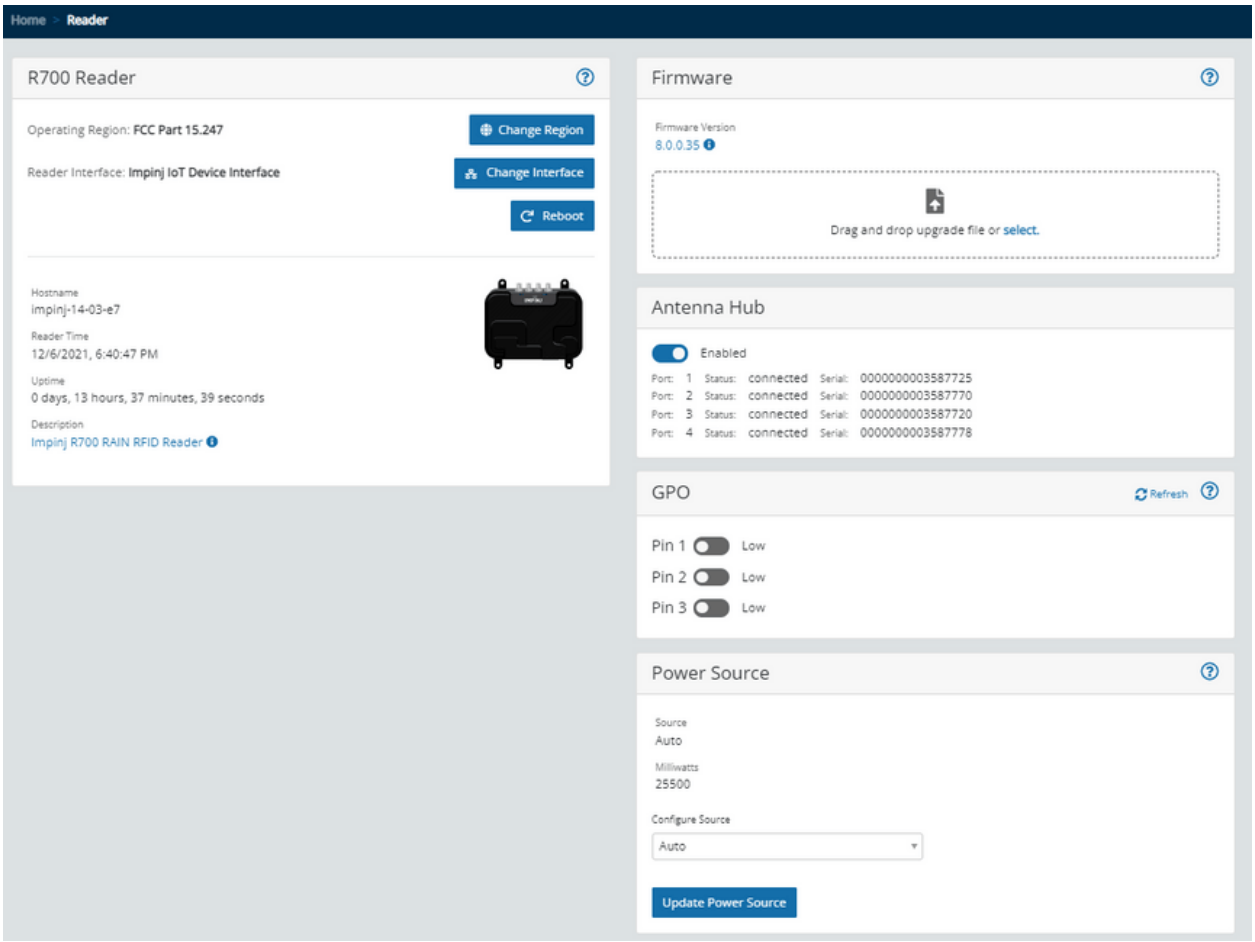
2 Reader Management

The **Reader** page is the Impinj IoT Interface web UI landing page. It is the first screen loaded after you log in. From this screen you can:

- Update firmware
- Change the interface
- Reboot
- Change region (if on a reader that supports multiple regions)
- Enable or disable the antenna hub
- Configure GPO
- Configure the PowerSource

As shown in the image below, the screen is generally divided into five discrete panels:

- **Reader Information**: To reboot, change the region, or change interface.
- **Reader Firmware**: To upgrade reader firmware, or access firmware version details.
- **Antenna Hub**: To enable or disable the antenna hub.
- **GPO**: To set the GPO pin number and state.
- **Power Source**: To set the power source.



Important: The firmware is configured so that only those features that have been enabled for a particular reader will appear on the screen. For example, if the region has been preconfigured and will not change, the **Change Region** button does not appear.

2.1 Update Reader Firmware

You can update the device firmware from the **Reader Firmware** panel of the Reader page. This panel displays the current firmware version. Click on the version number for additional firmware version details such as build revision, build date and time, and build plan.

To upgrade the reader firmware:

1. Select a .upgx or .raucb file either by clicking on the select link or by dragging the file into the boxed region.
2. Click the **Reboot button**.

Note: Installation is automatic when the upload completes, but the new firmware is not active until the reader is rebooted.

2.2 Enable or Disable Antenna Hub

This panel is self-explanatory. Click the button to enable or disable the antenna hub. Reboot the reader for changes to take effect.

Note: If the antenna hub is not attached, the toggle button will immediately revert to disabled.

2.3 Configure GPO

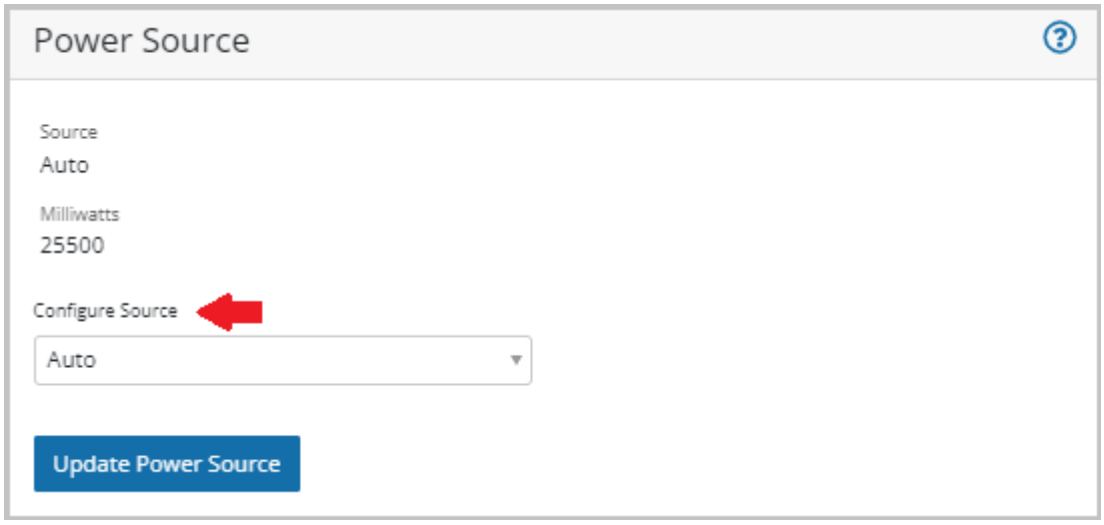
The Impinj R700 reader permits GPO control. Click the toggle button to set the relevant GPO pin. The toggle turns blue to indicate that the GPO has been powered on (High) or off (Low).

Note: The GPO panel only appears when the reader interface is in IoT mode. It does not appear when the reader is in LLRP mode.

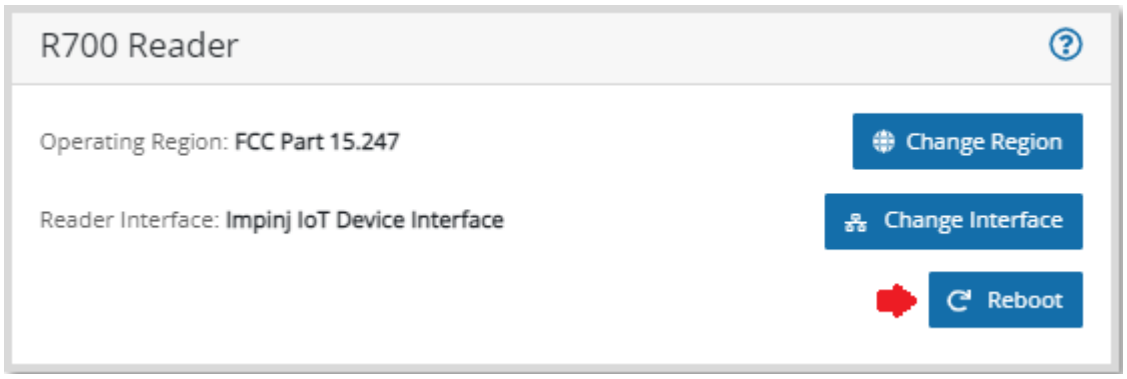
2.4 Configure the Power Source

The power reader power source can be set from this panel. To set the power source:

1. Select the source from the **Configure Source** drop down list and click the **Update Power Source** button.



2. Click the **Reboot**, shown in the image below.



Changes will take effect on reboot.

Important: Automatic selection of PoE+ depends on LLDP support at the power source. If your switch or injector adapter does not support LLDP but does provide PoE+ power levels, you must manually configure the expected power levels using the RShell command:

```
> config system power source PoE+
```

For more specific information about configuring the power source, see the **Impinj R700 RAIN RFID Reader Installation and Operations Guide**.

3 Manage Profile Presets

Presets enable the user to customize the behavior of reader operations. For the R700 readers, inventory is currently the only profile in use. The web UI provides a default preset with common preconfigured settings that the user can run to quickly access data from the reader. The default preset cannot be edited but can be cloned. The clone can then be edited. A new preset is generated either through cloning an existing preset or creating an original preset by clicking the **Create New Inventory Preset** button.

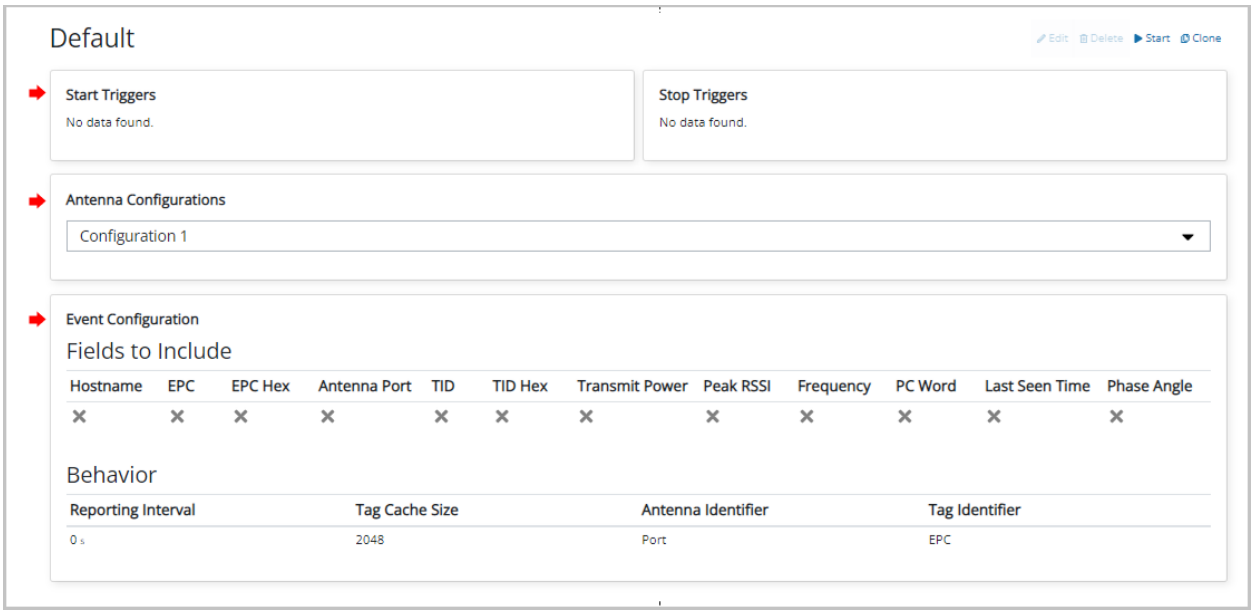
It is often more convenient to clone and edit the default preset, as it has already been configured with the necessary and/or common settings specific to each reader. To clone a preset:

1. Click the clone button as shown in the image below.
2. Enter a unique, alpha-numeric ID. Note that the ID cannot include spaces but can include periods, underscores, and dashes.
3. Click the **Clone** button.

3.1 Configure Inventory Presets

Inventory presets are divided into three areas, as shown in the image below:

- Antenna Configurations
- Event Reporting Configuration
- Start and Stop Triggers



3.1.1 Antenna Configurations

Antenna configurations define how operations behave on a particular antenna port. The IoT interface enables the user to configure each antenna individually.

Note: An antenna configuration is required for each antenna used during the operation, designated by the port number.

Four fields are required:

- **Antenna Port**: Designated port number for antenna in use.
- **Transmit Power**: Sets the output power for the reader. Note that power is specified in units of centi decibel milliwats, or cdBm. For example, 30 dBm is entered as 3000.
- **Session**: Session in which to run inventory. Please refer to the **Impinj R700 RAIN RFID Reader Installation and Users Guide** for specific information about sessions. Also very useful is the Fundamentals of RAIN RFID course found on the Impinj Training Portal.

- **Population Estimate**: The estimated number of tags being inventoried, specified to the power of two. It is recommended that the number specified is relatively close to the average number of tags the device will see at any given time. If there is variability in the number of tags that may be seen, always choose the lowest value in the range. It is more efficient for the device to inventory a tag population exceeding the expected value than if the value is set too high.

Although not required, **Antenna Name** is useful for assigning the antenna port a name, which is then displayed in all Tag Inventory reports. This is particularly handy when there are several antennas connected to the same device. The name can be descriptive, such as "dockdoor 1" to describe a particular area the antenna covers.

Other optional fields include:

- **Receive Sensitivity (dBm)**: The receive sensitivity of the antenna in dBm.
- **Tag Access Password Hex**: Enter a hexadecimal password here as necessary for when an operation requires tag access command execution.
- **FastId**: Select enabled or disabled. Fast ID instructs compatible tag chips to backscatter the EPC and TID together during an inventory.
- **Search Mode**: Specifies how the reader targets tags based on their state and how the state of a tag changes after it is read. Understanding EPC Gen2 Search Modes and Sessions is an excellent source of more information about search modes, as is What Reader Mode Session and Search Mode should I use for my application.
- **RF Mode**: Select a Reader Mode to specify the rules used for communication between the reader and the tag. Impinj R700 Reader Modes (RF Modes) from our Support site provides a good overview. What Reader Mode, Session, and Search Mode should I use for my application? is another useful article.
- **Power Sweeping**: Check to enable inventory rounds to be performed at increasingly power levels.
- **Tag Authentication**: Check to enable a tag authentication challenge to be sent to every inventoried tag.

Antenna Configuration also contains two buttons:

- **Add Filter** button: Click this button to add a filter. The user can add up to two filters. See Filters, below, for more information.

- **Tag Memory** button: Click this button to open a tag memory configuration page, that enables the user to read the various memory banks of a single tag. The user can create up to four tag memory reads.

3.1.2 Filters

The filter configuration page opens with six fields. The first field is **Tag Filter Verification**, a custom parameter that controls behavior of the backup tag filter functionality, which can be used as a secondary

filter for any tags that ignore the Gen2 filtering. This field is not Filter specific, affecting one or both filters if created. Select Active to enable filtering.

The other five fields are specific to each filter:

- **Memory Bank**: Select from EPC, TID, or User memory partitions. User is the memory partition containing the user memory, if applicable. This field defaults to EPC.
- **Action**: Select to Include or Exclude tags that match the filter. This action defaults to Include.
- **Bit Offset**: Required. Enter a number to determine the number of bits in memory at which to apply the filter.
- **Mask**: Enter the hexadecimal value to filter tags by.
- **Mask Length**: Enter the bit length of the mask. If the mask's length is not divisible by four, specify in bits how much of that mask should be used. The mask is left-justified.

When the user adds a second filter, the **Filter Link** field appears. Select whether to match either filter, or to match both.

3.1.3 Tag Memory Reads

The Tag Memory Reads configuration page opens with three fields, as follows:

- **Memory Bank**: Select the name of the memory bank to read from, either EPC, TID, User, or Reserved.
- **Word Offset**: Enter the offset, in 16-bit words, where reading begins.
- **Word Count**: Enter the number of 16-bit words to read.

3.1.4 Event Configuration

This configuration defines what information is included in the event report, and controls how and when Tag Inventory events are reported. It is set globally, so any differences in behavior between antenna reports are due to differences in the configuration of the antenna configuration.

The reader hostname is included if the **Include Reader Hostname** checkbox is checked. It is unchecked by default.

The **Tag Inventory - Included Fields** section allows the user to choose which fields are included in an event report. The user can include as many or as few fields as needed. Note that the TID and EPC each have two versions; plain EPC and TID are in Base64 format, whereas the hex option provides the data in traditional hexademical format.

The **Tag Inventory - Behavior** section controls how and when Tag Inventory events are recorded. **Reporting Interval** is optional and by default is set to 0, which disables this feature. The **Tag Cache Size** can be set to a minimum of 1024 and a maximum of 8192. It is set to 2048 by default.

3.1.5 Start and Stop Triggers

Triggers are configured to start and stop the inventory preset and are controlled by signals from the General Purpose Input (GPI) port number.

Click the **Create New Start Trigger** or **Create New Stop Trigger** to create a new trigger. Enter the port number in the **GPI** text box. The user can create as many triggers as there are ports available. Inputting port numbers beyond those available to the reader will have no effect.

The **Transition** drop down list allows the user to choose if an increase or decrease in voltage will activate the trigger. **High to Low**, indicating a decrease, is selected by default.

4 Event Reporting

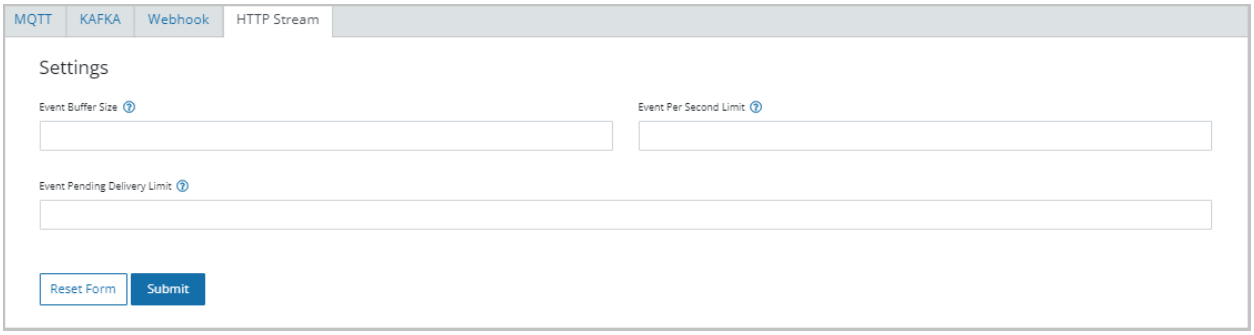
Event Reporting enables information to be retrieved from the Reader. The Impinj Rest API provides four methods for retrieving information, **HTTP Streaming**, **MQTT**, **Kafka**, and **Webhooks**.

4.1 HTTP Streaming

This is the easiest of the methods, as no third party applications are required and data can be accessed via the browser or by calling the curl command-line utility. Because the Impinj Reader hosts the http server, clients can access the stream directly.

Note that only five clients can access the stream at one time, so if many clients require access at the same time, a third-party application such as MQTT or Kafka might be the better option.

If the IOT Interface is enabled, HTTP Stream output is always available and does not need to be enabled or disabled. The HTTP Stream page does allow you to modify the behavior of the stream, as shown in the image below.



Note that these settings apply to all your connected clients.

Event Buffer Size:

Determines the maximum number of events waiting for delivery that the device can hold in memory. In this case, waiting for delivery either means that the device is waiting on time or data limits that have been configured in the preset, or that the device as been disconnected from the network. In either case, the reader stores the latest events up to the amount specified and discards older events.

Event Per Second Limit:

Dictates how many events per second that the device sends across the stream. Note that this does not limit the number of events stored in memory, so if Event Per Second limit is set to be slower than **Reporting Interval(s)** in the preset, the buffer will fill up and create overflow events. To avoid losing data, ensure that the generation rate of events does not exceed the transfer limit set by this field.

Event Pending Delivery Limit:

Determines the maximum amount of time in minutes that an event will be stored in the event buffer while waiting for transfer. If the event cannot be tranferred through the stream within that time limit, the event is deleted.

4.2 MQTT

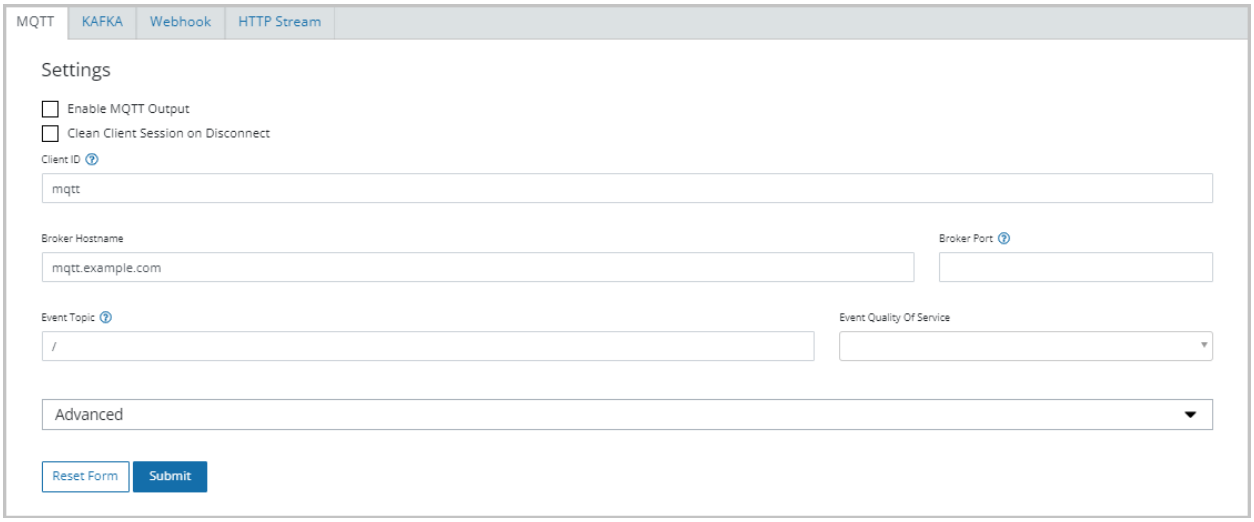
MQTT stands for Message Queueing Telemetry Transport. It is a client-server, publish-subscribe protocol that is lightweight and fairly straightforward. It is created for use on hardware with limited memory and/or processing power and is used when bandwidth or connection stability can be an issue, as is common with IOT-based applications. Impinj includes MQTT as one of its communication protocols because of its reliability and because it is already a common solution in the IOT space.

MQTT differs from HTTP Stream in that MQTT accepts an unlimited number of clients. Note that MQTT allows any client to be a publisher or a subscriber of messages, but the current version of Impinj REST interface only exposes the MQTT publisher, so is only used by the Reader to output event data. For that reason this document only covers how to properly connect Impinj Readers to MQTT brokers.

Unlike HTTP Stream, MQTT can be enabled or disabled, by selecting or deselecting the **Enable MQTT Output** checkbox at the top of the MQTT **Settings** page.

With MQTT, a client can connect to the broker as a clean session or as a persistent session. A clean session means that the broker deletes the client's subscriptions when it disconnects and must recreate those subscriptions when it reconnects. With a persistent session, the broker stores all non-QoS0 (Quality of Service level 0) messages that are published while the client is disconnected and resubscribes automatically upon connection. The default is for the session to persist. If you want a clean session, click the **Clean Client Session on Disconnect**.

The figure below illustrates the basic fields required to connect and run an MQTT broker.



- **Client ID**: A string used to uniquely identify the Reader to the MQTT broker and allow clients to be durable across several disconnects and reconnects.

- **Broker Hostname**: IP Address or Hostname of the MQTT broker.
- **Event Topic**: The base topic to which events will be published. MQTT requires that each message be published with an assigned topic that is used to organize and route the message.
- **Broker Port**: Defaults to 1883 if not specified.
- **Event Quality of Service**: The agreement between the sender and the receiver that defines the behavior of handshaking to ensure delivery. MQTT offers three levels:
  - **0**: No guarantee of delivery. The sender sends out a message with no expectation of receipt from the receiver. Requires lowest overhead.
  - **1**: Message will be delivered at least once, possibly more. The sender looks for a response message from the receiver. If the sender does not see an acknowledgement, will send the original message again. The message adds some overhead.
  - **2**: Guarantees that the message is received only once. Accomplished by the delivery of several acknowledgement messages between sender and receiver. This requires the most overhead and is used only in situations where bandwidth is less important than precise delivery.

As well as these required fields, the IOT web UI offers optional fields for more advanced configuration. These fields are not required but enable increased functionality. To access them, click in the **Advanced** field. The panel will drop open, as shown in the image below, divided into the following sections:

- **Authentication**: Set a Username and Password for access to the broker.

- **Will**: Configure a message to notify other clients when a client disconnects ungracefully. For more information, see Last Will and Testament - MQTT Essentials: Part 9.
- **Event Configuration**: Configure Event Buffer Size, Event Per Second Limit, and Event Pending Delivery Limit. Click the question mark for descriptions of these fields. Note that **EventPerSecondLimit** dictates how many events per second the device sends but does not limit the number of events stored in memory. If set slower than the speed of event generation, the buffer fills up, creating overflow events. To avoid this, ensure that the generation rate of events does not exceed the transfer limit set by the **EventPerSecondLimit** field.
- **Connection**: Set the **Keep Alive Interval** field. The **Keep Alive Interval** acts as a heartbeat between the client and the broker. The broker assumes the client is connected as long as it continues to receive this signal.

Note: Although this document only provides a very basic overview of its capabilities, MQTT is a robust protocol with a full set of features and options. For more comprehensive documentation, see the HiveMQ site.

4.3 Kafka

Like MQTT, Kafka is a client-server publish-subscribe protocol. Both require external brokers to manage data flow, are highly scalable, and can handle a high number of clients if the broker systems are powerful enough to handle the load.

A major difference between Kafka and MQTT is that MQTT is designed to be simple, lightweight and easy to use, whereas Kafka is built to be powerful, flexible, and highly performant. This means that a minimal Kafka infrastructure setup takes more preparation than MQTT and has a longer learning curve to fully understand.

Like MQTT, Kafka allows any client to be a publisher or a subscriber of messages but, as with MQTT, the current version of Impinj REST IoT Interface only exposes the Kafka publisher, so this document will focus on concepts that must be understood from the perspective of a producer in order to configure the Impinj IoT device.

Kafka is intended to be a distributed data framework that emphasizes redundancy and performance through parallelization. Events are stored sequentially to logs and each log is dedicated to a specific topic that is stored on disc, accessible by any number of clients for a configured amount of time. Clients read the topic log in sequence, keeping track of their own place in the log. If a client gets disconnected, it remembers its place in the log and continues reading from that place when it reconnects.

Topics are divided into a number of partitions, and replication is implemented at the partition level. Each record in a partition is assigned and identified by its unique offset. Partitions allow topics to be parallelized by splitting the data across multiple brokers.

Manual organization of these brokers and partitions would rapidly become unwieldy, so Kafka utilizes a number of automatic load-balancing algorithms to simplify management. However, one key concept that must be understood in order to configure the impinj IoT Device Interface is event ordering.

4.3.1 Event Ordering

Each broker can divide a single topic across multiple, often several, partitions. The default behavior of the broker is to distribute all topic events across all allowed partitions for that topic. This can disrupt the order of events because an ordered relationship of events across partitions cannot be guaranteed.

The most important concept to understand is that *event ordering is only guaranteed within a single partition of a topic.*

In some cases this doesn't matter, but often the situation requires that events are read in the order they were produced. To guarantee ordering, the producer provides a key, written with the event, and all events with that key are put into the same partition, preserving order.

4.3.2 Configuring Kafka

Although you can interact with Kafka through either the REST API or web UI, this document focuses primarily on how to use the Impinj web UI to configure the reader, as the client, and the broker.

Like MQTT, Kafka can be enabled or disabled by selecting or deselecting the **Enable KAFA Output** checkbox at the top of the MQTT **Settings** page.

The page is divided into two sections. The first five fields at the top of the page enable you to configure the reader as the client. Beneath those fields, under **Bootstrap Server Configurations** are the fields used to configure the broker.

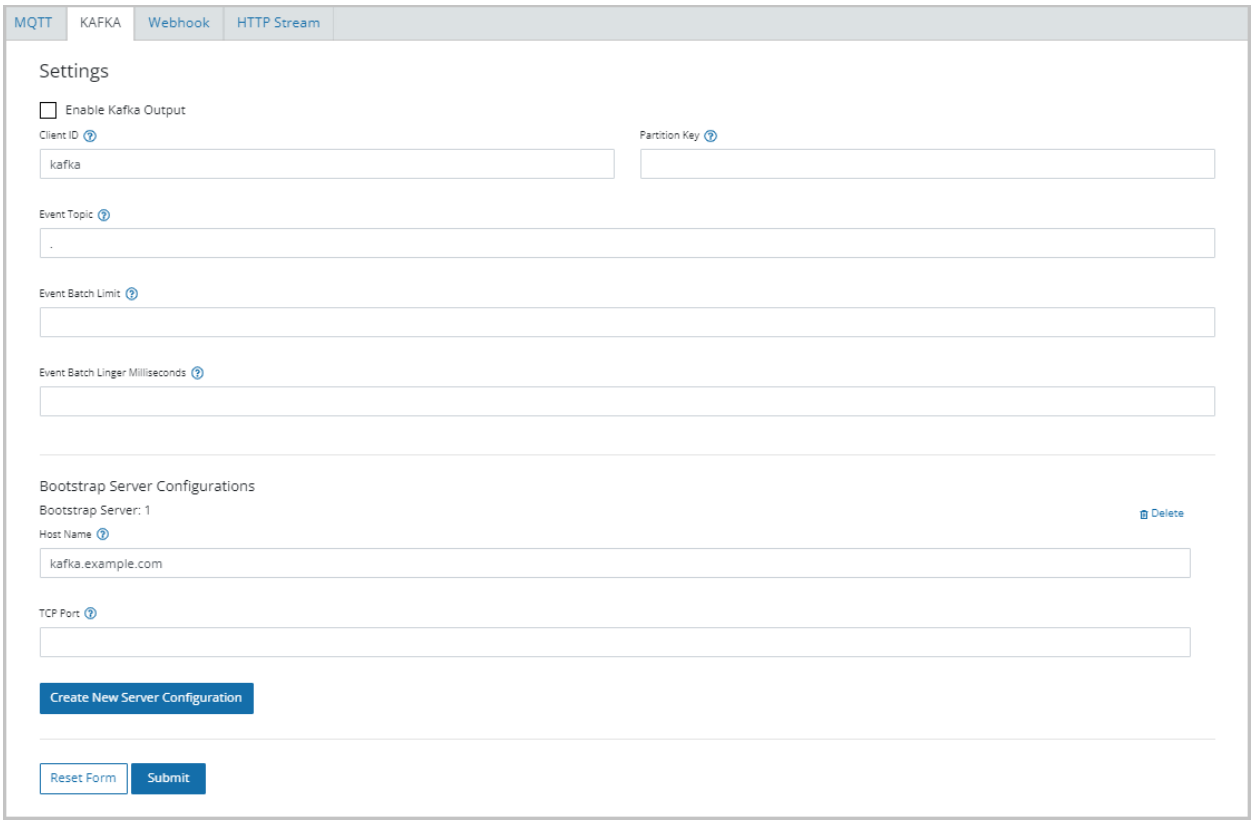
As you can see in the image below, three fields are completed when you first expose the **Kafka** tab under **Event Reporting**:

- **Client ID**
- **Event Topic**
- **Host Name**

These fields are required and should be self-explanatory for developers familiar with client-server publish-subscribe protocols, but for more information, click the **Help** icon to the right of each field title.

The following fields are optional:

- **Partition Key**: Defaults to "" if left blank. Specifying the partition key will force events from this device to be written to the same partition within a broker rather than spread across multiple partitions. This guarantees ordered event reads but may not always be necessary.
- **Event Batch Limit**: An integer specifying maximum number of events/messages batched within one message set. This field is used to group individual events into batches in order to optimize performance. If the number of events per second is very high, sending each transmission individually creates overhead that can overwhelm the system. Sending the data in batches improves performance, at the cost of potentially increased latency.
- **Event Batch Linger Milliseconds**: An integer specifying the amount of time, in milliseconds, to wait to send batch messages when **Event Batch Limit** has not yet been reached. A value of 0 will send each message as it is available.
- **TCP Port**: Under **Bootstrap Server Configurations**, the TCP port used to connect to the Kafka broker. Defaults to 9092 if left blank.



Click **Submit** to commit your changes. A notification appears, confirming that the changes have been committed. If the broker is connected, the gray dot in the notification bar at the top of the screen turns to green and another notification appears, confirming that the broker is connected.

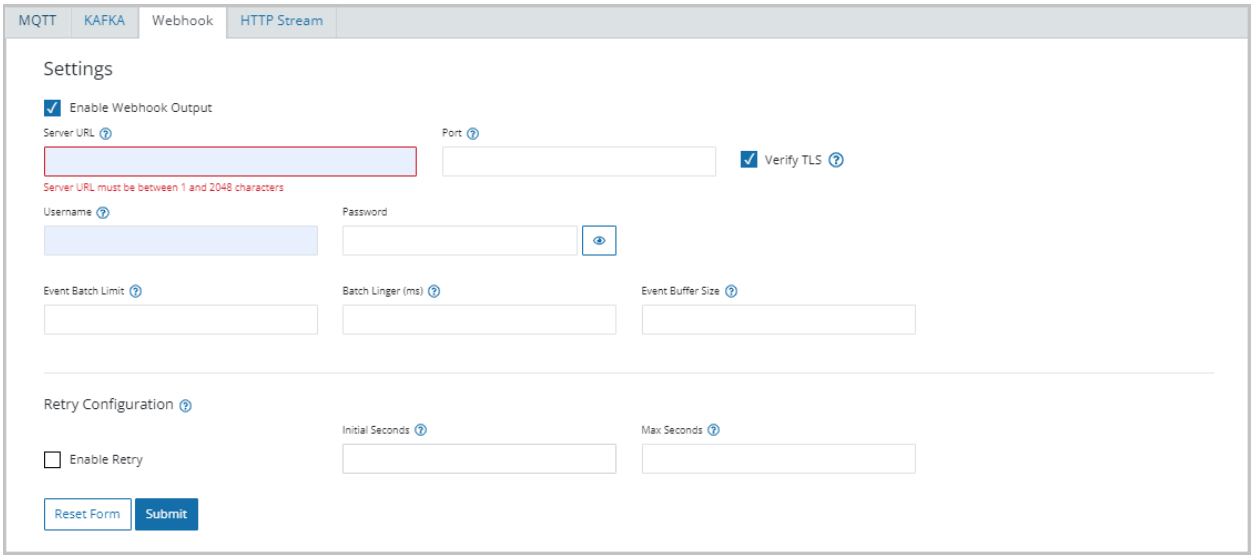
4.4 Webhooks

Webhooks are user-defined HTTP callbacks that receive events in batches from the reader. Impinj includes Webhooks as a method for processing events because its near universal use among developers makes it easy to adopt and support. It also enables users to upgrade to R700 without losing the functionality previously provided by Speedway Connect.

To configure Webhooks, click **Event Reporting** from the navigation bar, then click the **Webhook** tab.

Like MQTT and Kafka, Webhooks can be enabled or disabled by checking or unchecking the **Enable Webhook Output** checkbox.

As shown in the image below, the **Server URL** input box is the only field required.



The following fields are optional:

- **Port**: The port on the receiving server. If left blank, defaults to 80 for http server URLs and 443 for those that are https.
- **Verify TLS**: If checked, verifies that the certificate supplied by the web server is vouched for by a valid certificate authority (CA). Used with https server URLs.
- **Username**: Used for authentication. Only basic authentication is supported.
- **Password**: Self-explanatory. Click the eye icon to the right of this field to toggle text display.
- **Event Batch Limit**: The maximum number of events sent in a batch. Defaults to 10,000 if left blank. The minimum is one. The Maximum is 300,000. A higher batch limit results in fewer but larger batches, which saves on network traffic and server request count at the expense of latency.
- **Batch Linger**: The number of milliseconds before a message must be sent, whether or not the eventBatchLimit requirement has been met. If no events are queued, an empty message is sent. For example, an installation with 100 readers and eventBatchLingerMilliseconds=1000 will issue at least 100 requests per second irregardless of whether the messages contain any events.
- **Event Buffer Size**:The minimum number of events that will be queued on the reader waiting for delivery. When the queue exceeds this size, events will periodically be purged and an "OverflowEvent" will be inserted into the queue in place of the purged events. Defaults to 100,000 if left blank. The minimum is 1,000. The maximum is 300,000.
- **Enable Retry**: If an error occurs while posting the webhook, and retry is enabled, the failed request will be stored and periodically retried. No additional requests will be attempted by the webhook until the failed request is successfully made.
- **Initial Seconds**: The initial time to wait before retrying the webhook. Defaults to one, the minimum, if left blank. The maximum is 300.
- **Max Seconds**: The retry period will be increased exponentially until this value is reached.

  Note: In the event of Network or HTTP server timeouts, the reader-side timeout request is 10,000 milliseconds and is not configurable. This timeout is in addition to the retry delay.

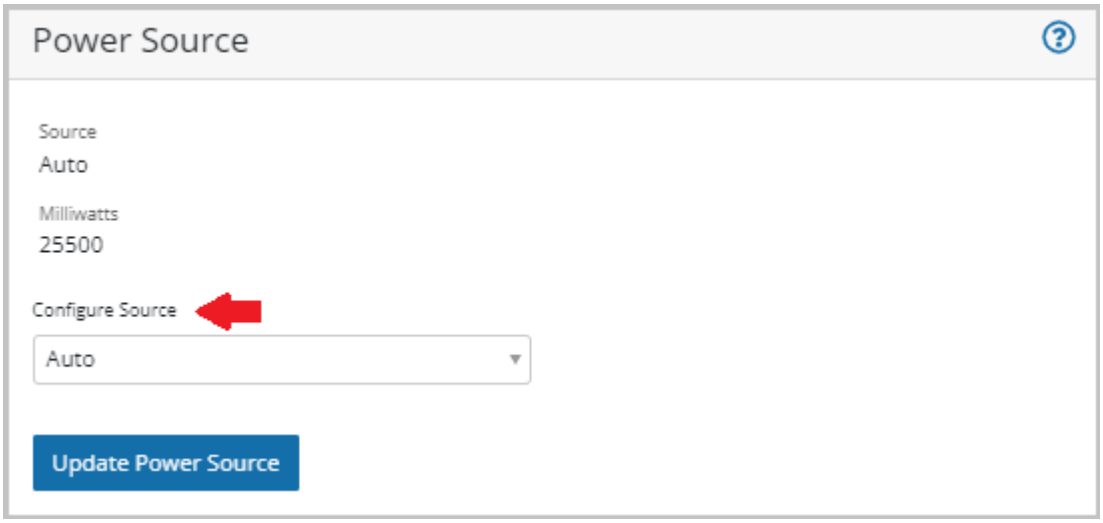
Click the **Submit** button to save.

For more information about Webhooks see What is a webhook: How they work and how to set them up.

5 Configure the Network

The network connection supplies both data connectivity and power.

The power source is configured from the **Power Source** Panel on the **Reader** page.



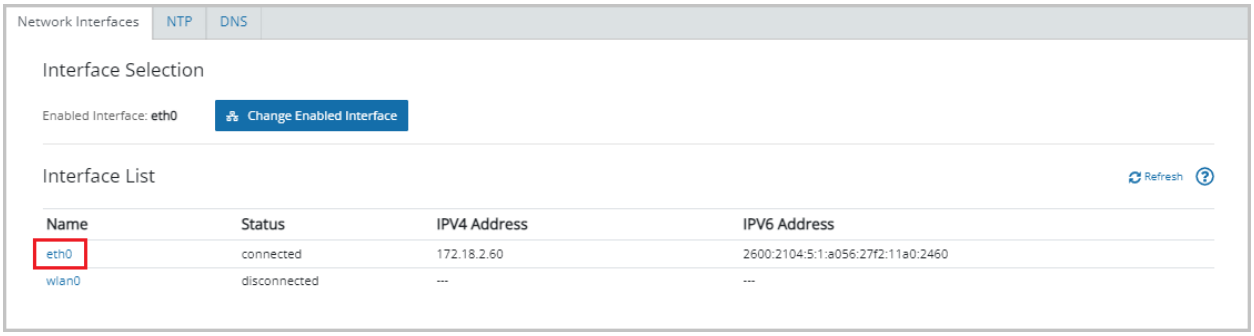
Note: The selected power source will persist across reboots but needs to be reconfigured after a full system reset.

To configure the network, click on **Network** from the navigation bar. This page has three tabs:

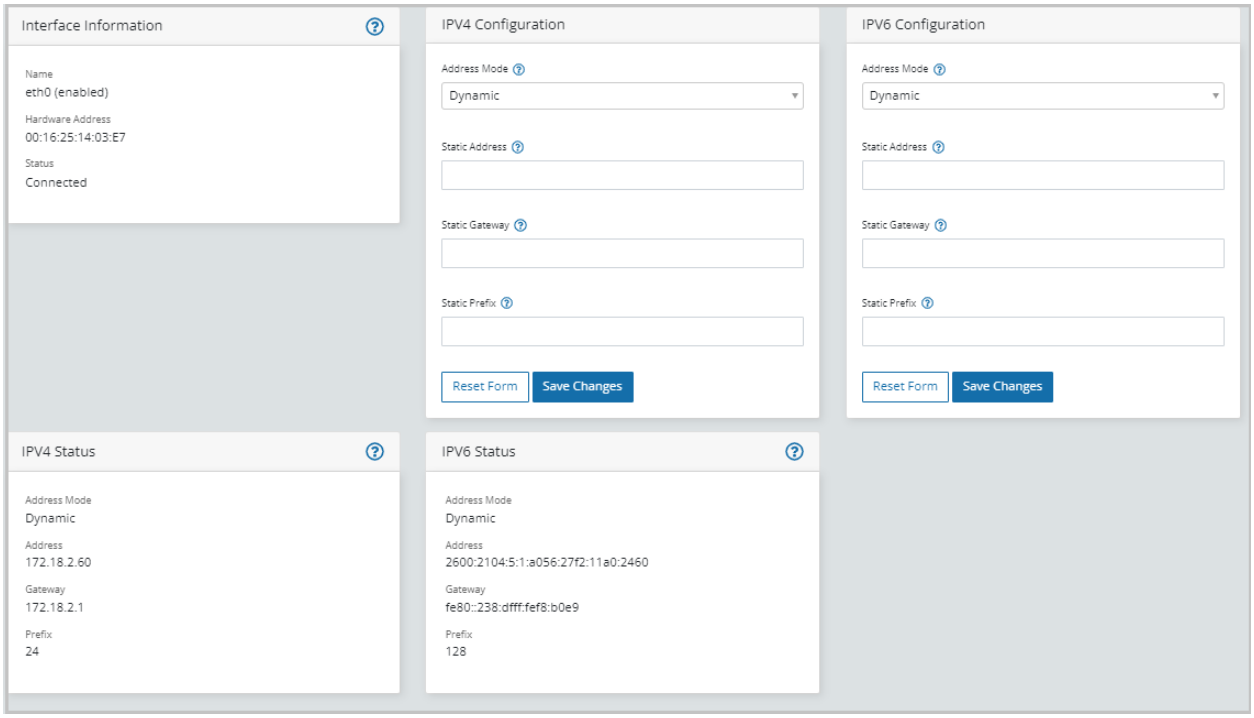
- **Network Interfaces**
- **NTP**
- **DNS**

5.1 Configure Network Interfaces

The Impinj R700 reader supports both IPv4 and IPv6 protocols. To access your Network Interface, click on the name of the interface in the **Interface List**.



A page similar to that in the image below appears.

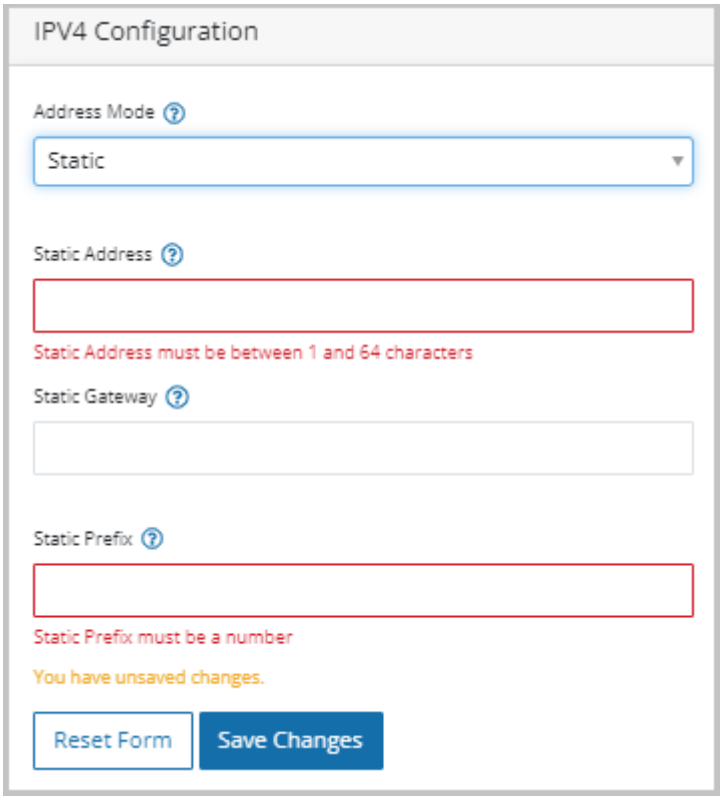


Network Interfaces

From this page you can determine the status of your network and set or modify your IPV4 and IPV6 Address Mode.

If **Dynamic** is selected, the interface will use DHCP to automatically obtain an IP Address. Static settings will be saved on the reader but not used.

When **Static** is selected, at least one **Static Address** and **Static Prefix** must be input. If left blank, those fields turn red and the changes will not be saved.

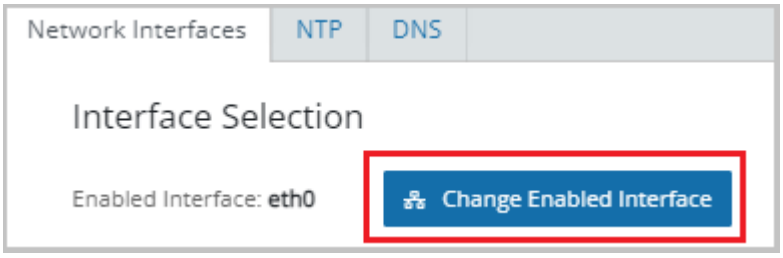


Required Static Address Fields

Static Gateway is optional but if no value is set, the gateway of the reader will not be configured and the reader can only be accessed within its local network.

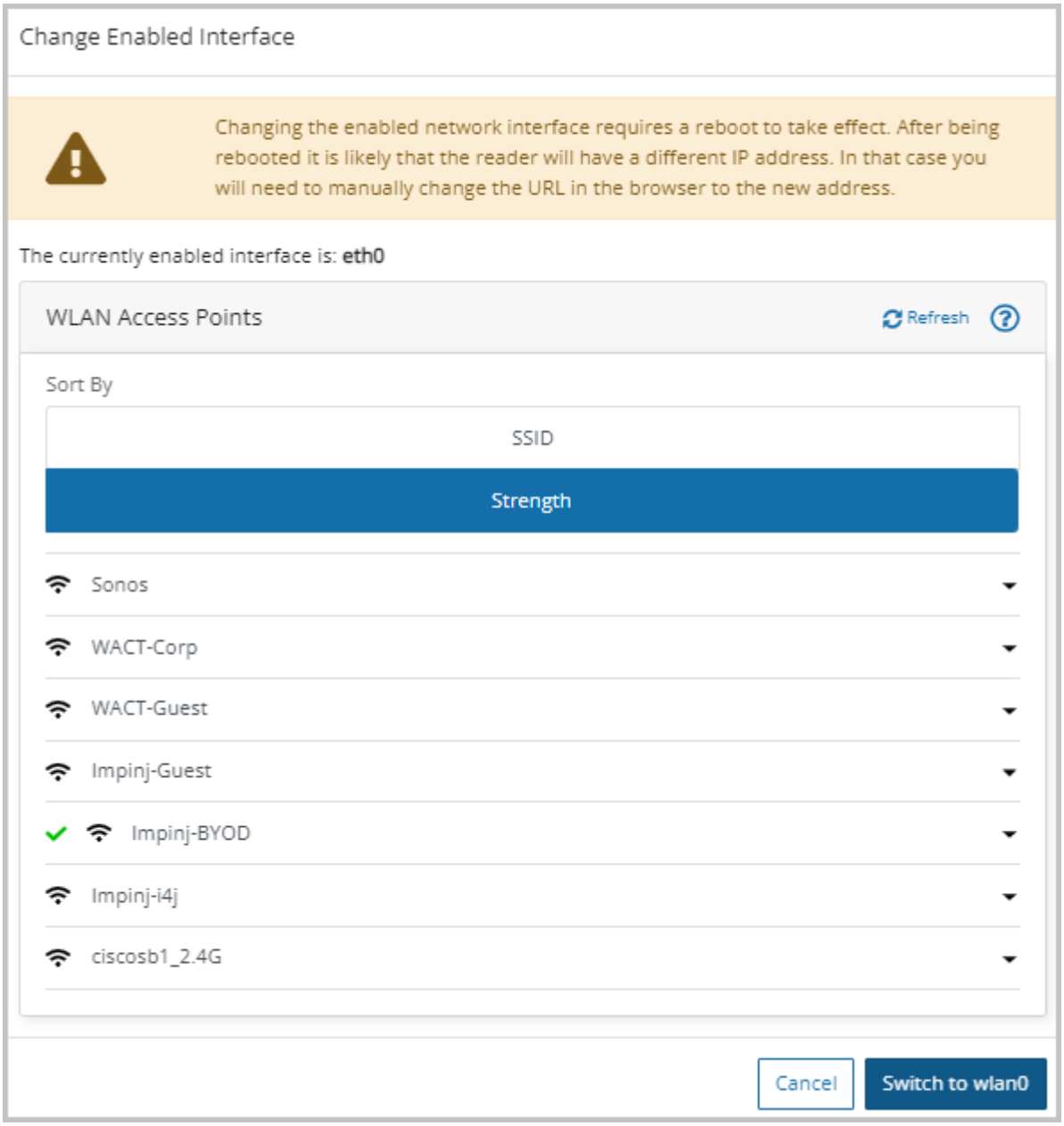
5.2 Change Enabled Interface

To change the interface that is enabled, click the **Change Enabled Interface** button from the Interface Selection page.



If switching from wlan to ethernet, from the the Change Enabled Interface dialog, click the **Switch to eth0** button, then click **reboot**.

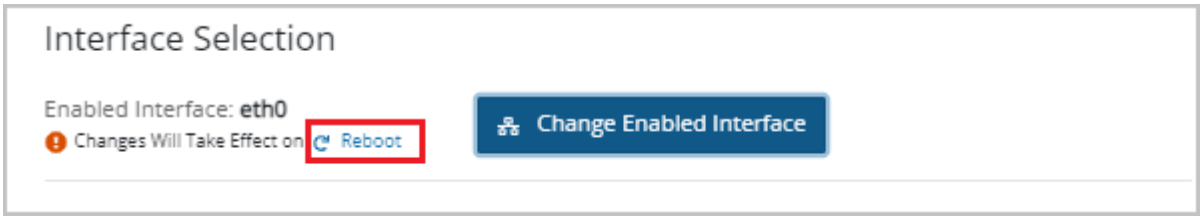
If switching to wlan, the Change Enabled Interface dialog opens with a list of available servers. These can be sorted by SSID or by Strength.



Change Enabled Interface

To connect:

1. Click on the desired server.
2. Click the **Connect** button.
3. Enter the password.
4. Click **Connect** again.
5. Click **Reboot**.



Note: Changing the enabled network interface requires a reboot to take effect. After being rebooted it is likely that the reader will have a different IP address. Depending on the network setup, DNS can take up to twenty minutes or more to map the hostname to the new IP address.

5.3 Add a Static NTP Server

The reader can support up to a maximum of six static NTP servers. To add a static NTP server:

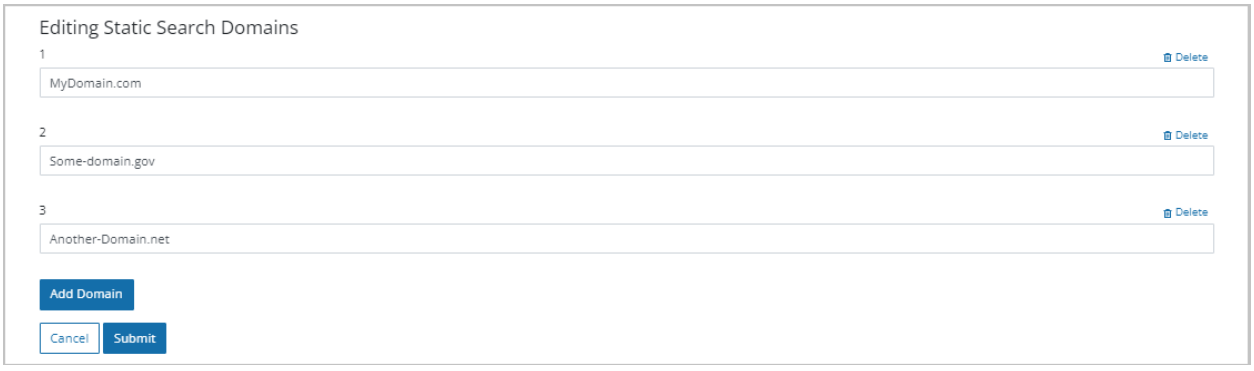
1. Click the **Enabled** toggle button to disable NTP.
2. Click the **Add Static NTP Server** button.
3. In the field that appears, enter the server address or hostname of the NTP server you want to add. This can be either an IPv4 address in dot-decimal format or an IPv6 address compliant with RFC-5952.
4. Click **Submit**.
5. Click the **Enabled** toggle button again to re-enable NTP.

The NTP server will appear on the list as shown in the image below. It may be deleted by clicking the **Delete** button.

5.4 Add a Static Domain or Server

Servers and search domains can be added under the **DNS** tab. The procedure is essentially the same for each. To add a static Search Domain:

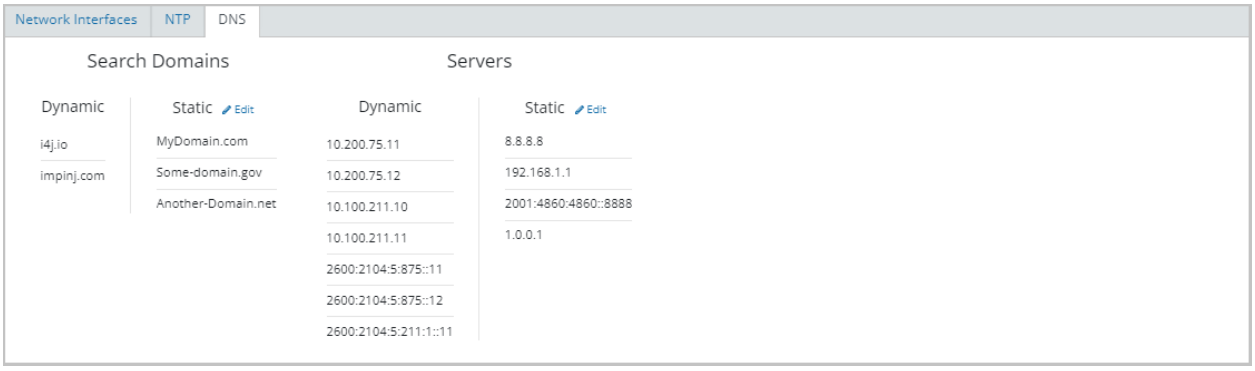
1. Click the **Edit** button under Search Domains to open the Editing Static Search Domains dialog.
2. Click the **Add Domain** button.
3. Enter the domain name in the input field.
4. If necessary, click **Add Domain** to add another domain. Add as many as required, up to a limit of 12 search domains. Click **Delete** to remove a domain.
5. Click **Submit**.



To add a static Server:

1. Click the **Edit** button under Servers to open the Editing Static Servers dialog.
2. Click the **Add Server** button.
3. Enter the IP address in the input field.
4. If necessary, click **Add Server** to add another server. Add as many as required, up to a limit of 12 servers. Click **Delete** to remove a server.
5. Click **Submit**.

The added static Search Domains and static Servers will appear on the DNS page as shown in the image below.



6 Notices

Impinj gives no representation or warranty, express or implied, for accuracy or reliability of information in this document. Impinj reserves the right to change its products and services and this information at any time without notice.

EXCEPT AS PROVIDED IN IMPINJ'S TERMS AND CONDITIONS OF SALE (OR AS OTHERWISE AGREED IN A VALID WRITTEN INDIVIDUAL AGREEMENT WITH IMPINJ), IMPINJ ASSUMES NO LIABILITY WHATSOEVER AND IMPINJ DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATED TO SALE AND/OR USE OF IMPINJ PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT.

NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY PATENT, COPYRIGHT, MASK WORK RIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT IS GRANTED BY THIS DOCUMENT.

Impinj assumes no liability for applications assistance or customer product design. Customers should provide adequate design and operating safeguards to minimize risks.

Impinj products are not designed, warranted or authorized for use in any product or application where a malfunction may reasonably be expected to cause personal injury or death or property or environmental damage ("hazardous uses") or for use in automotive environments. Customers must indemnify Impinj against any damages arising out of the use of Impinj products in any hazardous or automotive uses.

Impinj, GrandPrix™, Indy®:, Monza®, Octane™, QT®, Speedway®, STP™, True3D™, xArray®, and xSpan® are trademarks or registered trademarks of Impinj, Inc. All other product or service names are trademarks of their respective companies.

These products may be covered by one or more U.S. patents. See impinj.com/patents for details.

For more information, contact support@impinj.com