Developer Note

CELLULAR-BASED READER & GATEWAY APPLICATIONS

INSTRUCTIONAL GUIDANCE

Version 1.0

www.impinj.com

The following article aims to help systems integrators implement business solutions connecting Impinj RAIN RFID technology through a cellular network. In the absence of a fixed network connection, Impinj RAIN RFID can deliver item data in real time through a cellular network. This article will help users correctly configure their cellular routers as well as two different reader setups.

TABLE OF CONTENTS

OVERVIEW

Cellular networks enable RAIN RFID applications to deliver near real-time data without the need of a fixed network infrastructure.

Some RAIN RFID cellular use cases include:

- A Speedway reader installed inside delivery trucks sending real-time inventory data to a centralized database.
- An xPortal installed in a warehouse without a network infrastructure, sending notifications of Items entering or leaving the facility, to a cloud based application.
- A Speedway installed outside on a farm, tracking animals sending data about livestock to a centralized database.

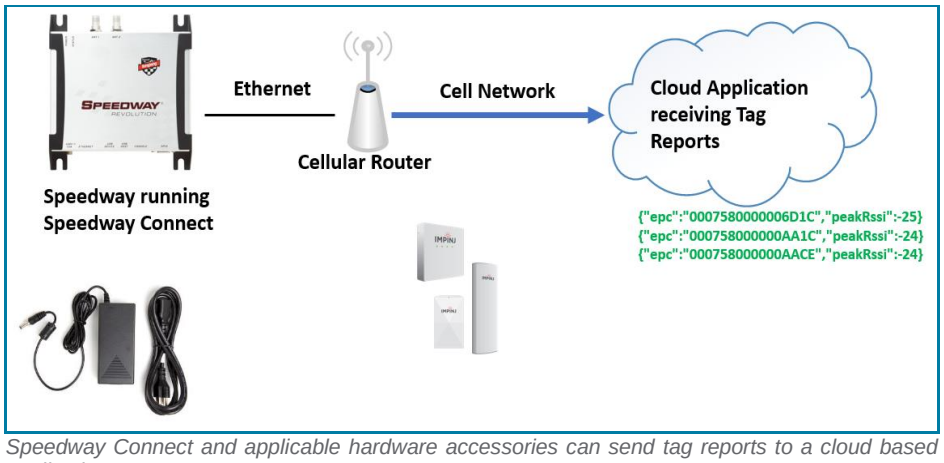
This article discusses a couple of approaches that have been successfully employed to implement a cellular wireless solution with Impinj Reader products.

APPLICATION EXAMPLES

Speedway Connect

Speedway Connect is an on-reader application developed by Impinj that has an HTTP post feature, well suited for cellular/cloud applications, which sends tag reports from the reader over the cellular network to a cloud based application. This architecture makes use of common protocols such as TCP/IP, HTTP, and JSON.

Figure 1: Cellular Network Connection with Speeway Connect – Architecture



Speedway Connect and applicable hardware accessories can send tag reports to a cloud based application.

Hardware and Software Requirements

This configuration requires a minimum set of hardware since Speedway Connect runs resident on readers and gateways.

- Speedway R120/220/420, xPortal, xArray, or xSpan
- Speedway Connect with License Key
- Impinj's Universal Power Supply
- Cellular Router (4G preferred)
- Ethernet Cable

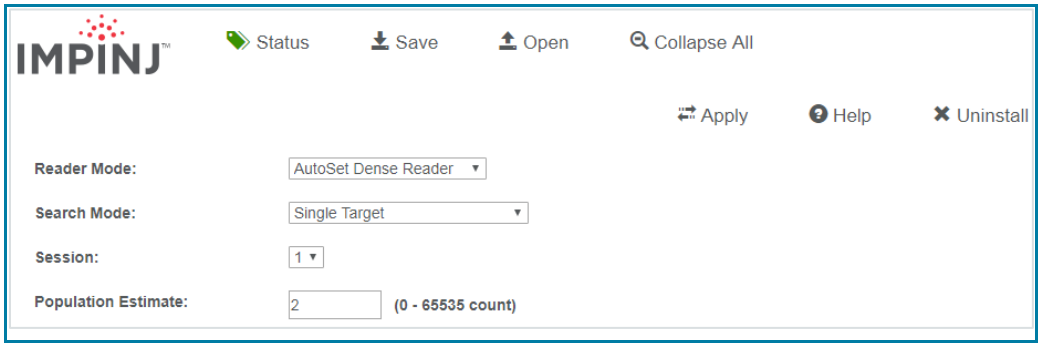
Configuration Details

This description assumes a basic knowledge of Speedway Connect. For more information go to: Speedway Connect.

When configuring Speedway Connect for this example, the following configuration steps were performed:

1. Downloaded and Installed Speedway Connect software on the reader.
2. Setup the reader's GEN 2 parameters. Specifically, we set Search Mode=Single Target and Session=1, so the reader will continuously inventory all tags in the field. When using Session 1, tags after they are inventoried will remain in B state for about 1 second allowing more difficult to read tags to also be read. See Understanding EPC Gen2 Search Modes and Sessions.

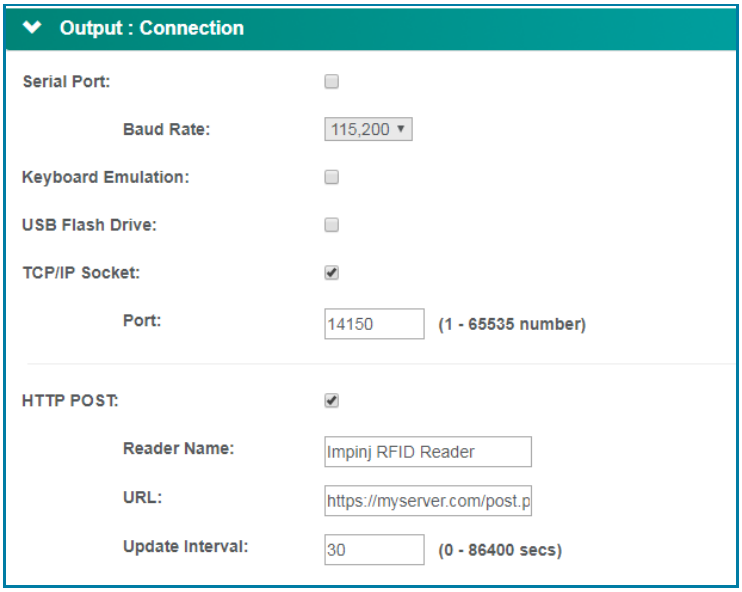
Figure 2: Speedway Connect – GEN 2 Parameters



At the top of the web page where the reader's GEN 2 settings are defined.

3. Under **Output : Connection**, TCP/IP socket was enabled in case needed for debugging purposes. Connecting to TCP/IP socket is a convenient method for verifying that Speedway Connect is reading tags. Also, HTTP Post was enabled so if tags are read by Speedway Connect will all be posted over HTTP. (Note that www.myserver.com in the URL field should be replaced with your own server.) The *Update Interval* was set to send HTTP posts every 30 seconds. You can choose a shorter *Update Interval* if your application requires.

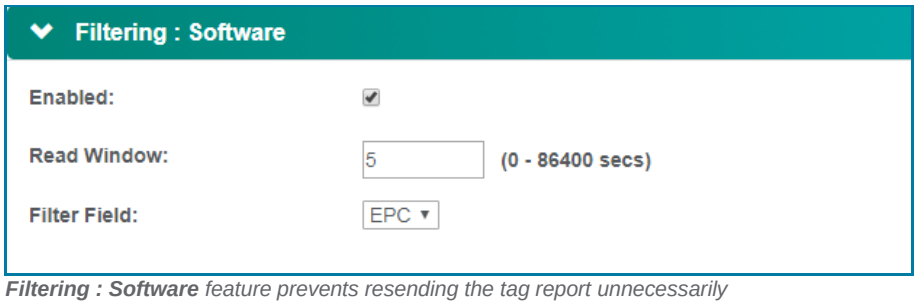
Figure 3: Speedway Connect – Output Connection Setup



TCP/IP Socket and HTTP Post are both enabled.

4. **Note:** Pay close attention to how often and how many tag reports are sent over the cellular network because cellular carriers charge based on data usage. The **Filtering : Software** feature is an effective way to limit the amount data sent over the cellular network. Below the "Read Window" of 5 seconds indicates that tag reports will only be posted when a tag is first seen in the field of view or in the case that a tag leaves the field of view for more than 5 seconds before it returns. The software filtering feature dramatically reduces the network traffic since without it, the tag reports would be resent every 30 seconds per the Update Interval. In other words, if the tag stayed in the field of view for 10 minutes it would be sent 20 times (2 times per minute * 10 minutes = 20 posts). With the Software filter enabled, the tag report is sent just once.

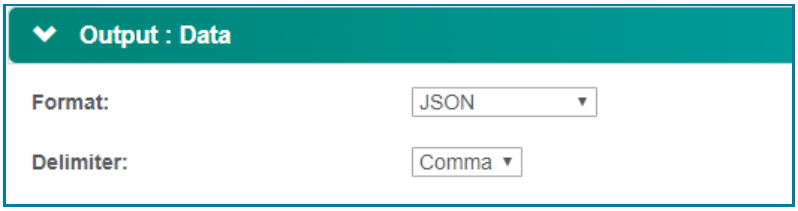
Figure 4: Speedway Connect – Filtering : Software



***Filtering : Software** feature prevents resending the tag report unnecessarily*

5. Under **Output: Data:** the format JSON was selected.

Figure 5: Speedway Connect – Output : Data



***Output : Data** is used to specify the JSON format*

6. Once Speedway Connect was successfully posting tag reports data to your server, we looked at the server to verify that the data was received.

Figure 6: Received Tag Reports

```
PHP_SELF = /post.php
REQUEST_TIME_FLOAT = 1495494079.16
REQUEST_TIME = 1495494079

No Post Params.
Empty post body.

Upload contains PUT data:
{"reader_name":"Impinj RFID Reader","mac_address":"00:16:25:11:B7:40","tag_reads":
[{"antennaPort":12,"epc":"E28011606000020497CE54E3","peakRssi":-69,"isHeartBeat":false},
{"antennaPort":8,"epc":"E2801160600002052AD3083D","peakRssi":-72,"isHeartBeat":false},
{"antennaPort":10,"epc":"E2801160600002052AD3083D","peakRssi":-72,"isHeartBeat":false},
{"antennaPort":10,"epc":"E28011606000020497CE54E3","peakRssi":-67,"isHeartBeat":false},
{"antennaPort":8,"epc":"E2801160600002052AD3083D","peakRssi":-72,"isHeartBeat":false},
{"antennaPort":8,"epc":"E2801160600002052AD3083D","peakRssi":-71,"isHeartBeat":false},
{"antennaPort":12,"epc":"E2801160600002052AD3083D","peakRssi":-71,"isHeartBeat":false},
{"antennaPort":6,"epc":"E28011606000020497CE54E3","peakRssi":-67,"isHeartBeat":false}]}
```

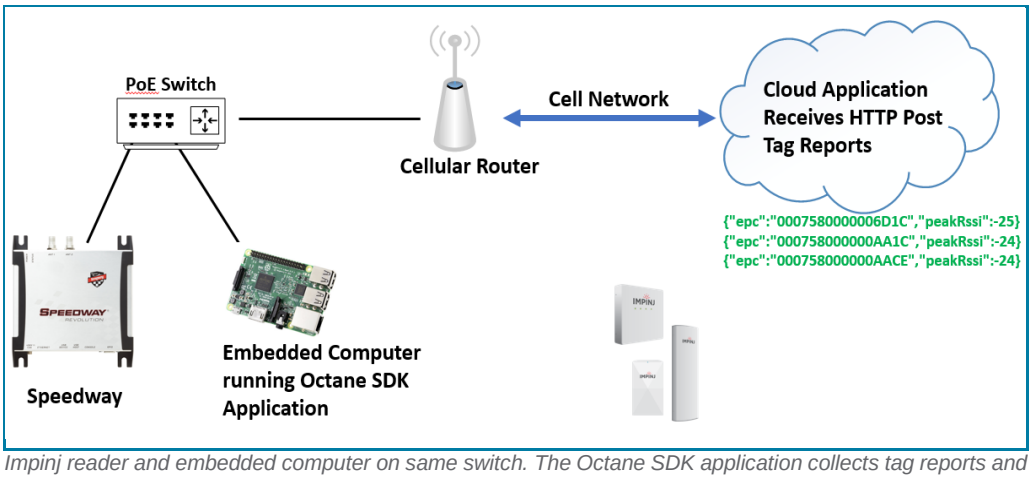
Tag reports are received on the server in a JSON format

See Cellular Router Configuration section for tips on cellular router Setup.

Embedded Computer Using Octane SDK

Another option for creating a cellular based solution is to write an Octane SDK application that sends tag reports to a centralized application on a cloud server. An example of such an application shown below, runs on an embedded computer installed on the same switch as the Impinj reader. This Octane SDK application will receive tag reports from the reader, package them up and send them to a cloud based application. While creating an Octane SDK application does require writing additional code, it provides full control on how the tag report data is aggregated before sending it out over the cellular connection thereby reducing the amount of unnecessary data transmission.

Figure 7: Cellular Network Connection using Embedded Computer – Architecture



Impinj reader and embedded computer on same switch. The Octane SDK application collects tag reports and posts them over HTTP to the cloud application

Hardware and Software Requirements

- Speedway R120/220/420, xPortal, xArray, or xSpan
- Octane SDK downloadable from support.impinj.com or NuGet.org
- Cellular router (4G preferred)
- Embedded computer (Linux or Windows)
- 3 Ethernet Cables

In this architecture, notice how the PoE switch both powers the Speedway and provides network connectivity to the devices.

Configuration Details

When configuring the Octane SDK solution, the following configuration steps were performed:

1. A basic Octane SDK .NET application was created that:
   - Connects to the Speedway reader
   - Customizes settings, starting with default settings
   - Sets up antenna port 1
   - Sets up the tag report method
   - Lastly, starts the reader
2. Notice the simple technique used to minimize traffic over the cell network: Set Search mode=Single Target, and set Session=2. This combination of search mode and session will generate tag reports (typically only once per tag) when the tag is in the field of view. In Session 2, tags will remain in the

B State until they leave the field of view of the Antenna. See Understanding EPC Gen2 Search Modes and Sessions.

3. Also, notice the use of hostname "speedwayr-11-22-33.local". Since this small 2 devices + cellular router network does not have a DNS server, using the .local suffix allows the application running on the embedded computer to connect to the Speedway.

Figure 8: Configuring the Settings Object

```
reader.Connect("speedwayr-11-22-33.local");   // access using .local address
Settings settings = reader.QueryDefaultSettings();
settings.TagPopulationEstimate = 8;
settings.Session = 2;
settings.SearchMode = SearchMode.SingleTarget;
settings.ReaderMode = ReaderMode.AutoSetDenseReader;
settings.Antennas.DisableAll();
settings.Antennas.GetAntenna(1).IsEnabled = true;
settings.Antennas.GetAntenna(1).MaxTxPower = false;
settings.Antennas.GetAntenna(1).TxPowerInDbm = 25;
settings.Antennas.GetAntenna(1).MaxRxSensitivity = true;
settings.Report.IncludeAntennaPortNumber = true;
reader.ApplySettings(settings);
reader.TagsReported += OnTagsReported;   // Tag Report hander
reader.Start(); (1).IsEnabled = true;
```

Code to setup reader to inventory tags on antenna 1.

4. Now that the inventory has been configured, let's look at the OnTagsReported method that is called each time a tag is read. This method posts a tag report defined by the TagData, see below, to the URI http://myserver.com/post.php.

Figure 9: OnTagsReported Method

```
void OnTagsReported(ImpinjReader sender, TagReport report)
{
    foreach (Tag tag in report)
    {
        Action<TagData> tagAction = (TagData td) =>
{
    Console.WriteLine("Write {0} {1}", td.EPC, td.Antenna);
            using (var client = new HttpClient())
            {
                var uri = new Uri("http://myserver.com/post.php");
                var json = new JavaScriptSerializer().Serialize(td);
 var stringContent = new StringContent(json, Encoding.UTF8,
"application/json");
                var response = client.PutAsync(uri, stringContent).Result;
                Console.WriteLine("status=" + response.StatusCode);
            }
        };
        TagData tagdata = new TagData(tag.Epc.ToHexString(),
tag.AntennaPortNumber);
        tagAction.Invoke(tagdata);  // run asynchronously
    }  // foreach
} // OnTagsReported
```

The OnTagsReported method that posts to a URI

5. We need to specify using a class called TagData that defines what is posted in the tag report.

Figure 10: TagData Object

```
class TagData
{
    public TagData(string e, int ant)
    {
        this.EPC = e;
        this.Antenna = ant;
    }
    public string EPC { get; set; }
    public int Antenna { get; set; }
}
```

TagData class defines the tag report fields to transmit

6. Once the data was sent over to the server, this is the JSON data that was captured.

Figure 11: TagData Object

```
Upload contains PUT data:
{"EPC":"00075800000000000006D1C","Antenna":1}
```

Data in JSON format that was captured on the server.

See Cellular Router Configuration section for tips on cellular router Setup.

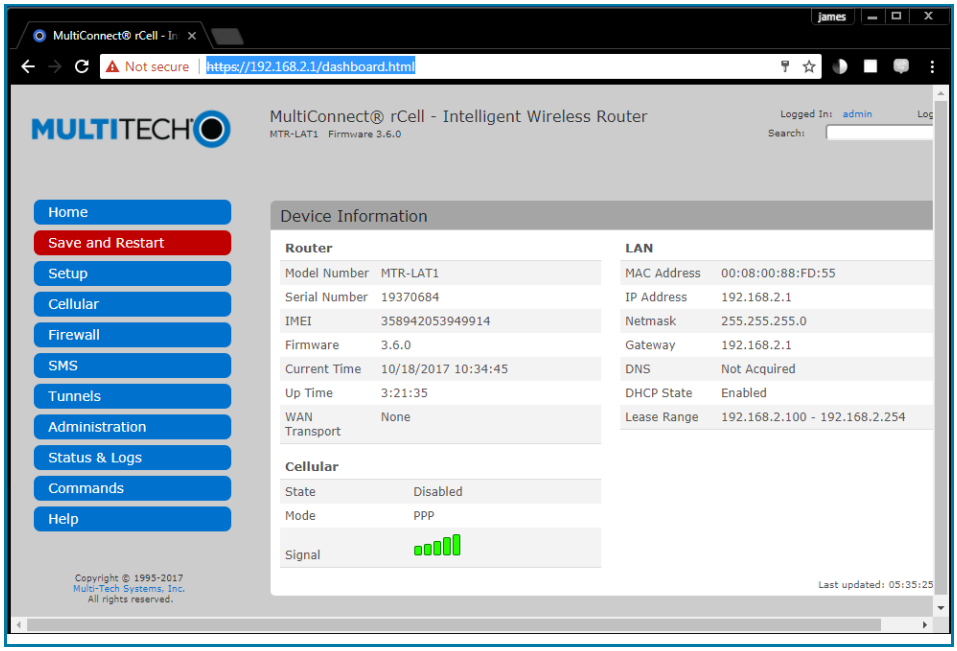
CELLULAR ROUTER CONFIGURATION

To setup your cellular router, refer to your router's documentation.

The following steps were taken from our MultiConnect® RCell 100 4G cellular router setup:

1. Installed an activated SIM card into the cellular router.
2. Connected a PC to the cellular router with an ethernet cable
3. Configured the cellular router from its administrative web page. This page shows device information including IP address and signal strength.

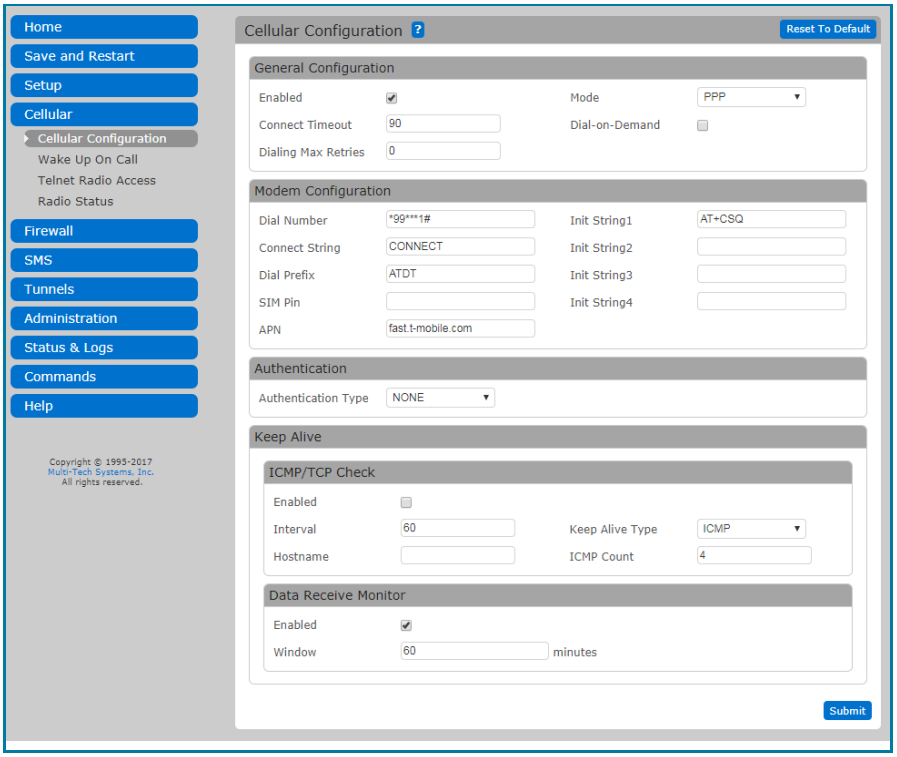
Figure 12: Cellular Router Configuration – Web UI Dashboard



Cell router Web UI configuration. (MultiConnect rCell)

4. The only configuration modification required was to specify the APN (Access Point Name). The T-Mobile® network uses the APN=*fast.t-mobile.com*.

Figure 13: Cellular Router Configuration – APN Specification



Cellular router Web UI Cellular Configuration page. (MultiConnect rCell)

BEST PRACTICES

- Always minimize the amount of data transmitted over the cellular network to avoid unnecessary charges and select the best data plan offered by your cellular carrier. Cellular providers typically provide a software that allows you to see how much data your SIM is consuming. Look at these reports especially right after deployment instead of waiting for the bill at the end of the month to avoid unexpected charges due to high data consumption.
- Make sure that the location of each installation has adequate cellular coverage. Check the RSSI value or signal strength from your router's web page.
- Use 4G based cellular routers in the United States since the 3G networks are being phased out.
- If there are cellular connection issues, often due to a weak signal preventing data communication, note the following:

Table 1: Resending HTTP Posts Following Cellular Connection Issue

USING SPEEDWAY CONNECT	USING OCTANE SDK
Automatically resends HTTP posts after failure due to connection loss	Developer should implement code that resends HTTP post when HTTP post failures are detected

EXTERNAL REFERENCE

http://support.impinj.com

Speedway Connect

Octane SDK

Understanding EPC Gen2 Search Modes and Sessions

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, MASKWORK RIGHT, OR OTHER INTELLECTUALPROPERTY 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"), including but not limited to military applications; life-support systems; aircraft control, navigation or communication; air-traffic management; or in the design, construction, operation, or maintenance of a nuclear facility. Customers must indemnify Impinj against any damages arising out of the use of Impinj products in any hazardous uses

Impinj, and Impinj products and features are trademarks or registered trademarks of Impinj, Inc. For a complete list of Impinj Trademarks, visit www.impinj.com/trademarks. All other product or service names may be trademarks of their respective companies.

The products referenced in this document may be covered by one or more U.S. patents. See www.impinj.com/patents for details.