CHAPTER 3

Azure IoT Hub

In the world today there are millions of devices of different types generating an enormous amount of data. A large number of these devices are part of a larger IoT solution in which the devices are sending their data to the cloud for storage, processing, and analysis. At a high level, IoT solutions can be broken down into core essentials of *device connectivity* and *data processing and analysis*, as shown in Figure 3-1.

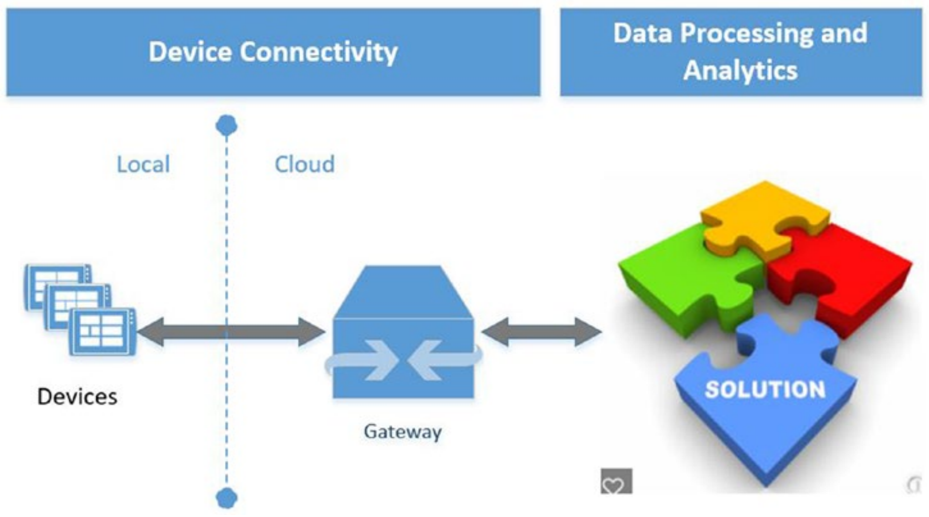


Figure 3-1. *IoT solution core essentials*

Device connectivity is simply devices that generate and collect data which is then sent to a cloud gateway. The cloud gateway acts as a mediator that gathers the incoming data and makes it available for further processing by other services and processes of the IoT solution. Yet, within the IoT solution architecture there are real and distinct challenges that exist. These challenges come in the form of how to create a secure and dependable connection between the devices and the back-end solution.

Think about how devices are being used today. Devices have been connected to cows, cars, and spaceships. Your phone, watch, step counters, and fitness trackers are data-generating devices. So, when dealing with device connectivity, the challenge is to figure out the best and most efficient method and approach for enabling and providing not only securely connected devices but also reliably connected devices.

S. Klein, *IoT Solutions in Microsoft's Azure IoT Suite*, DOI 10.1007/978-1-4842-2143-3_3

IoT solutions are not your typical client app. Think about it for a minute. As mentioned above, these IoT devices aren't sitting on your desktop and you don't hit them with your browser. No, these devices are out in pastures, on the street, or orbiting the earth. They are in farms and factories, appliances and gadgets.

As such, special attention needs to be paid to their unique, perplexing characteristics, such as

- Slow or unreliable network connectivity
- Limited power resources
- Lack of physical access to the device
- Possible use of proprietary or custom application protocols
- Human device interaction

While this list is in no way complete, it should give you an idea of the real challenges faced by IoT solutions today. In addition, there is another key characteristic of devices that cannot be overlooked. It should be noted that the arrow between the devices and the gateway (as well as the gateway and the IoT Solution) in Figure 3-1 is bidirectional. This is due to the fact that IoT solutions require secure, *bidirectional* communication between devices and cloud gateways.

Thus, not only are devices sending data (device-to-cloud communication), but they can also receive and process messages and information (cloud-to-device communication) from a cloud endpoint. Imagine a scenario where the IoT solution might send a message to the device telling it to change configuration values. For example, a message could be sent to the device to tell it to change the rate at which it pulls data or change the upper or lower temperature alert limits. For example, iPads for years have been able to signal back and wipe themselves if they are stolen. In another example, a company in the UK is limiting water based on telemetry from the sensors on industrial water meters.

What is needed is a service that overcomes these challenges, not only providing proper security and reliability but scalability when needed. Introducing Azure IoT Hub.

In the previous chapter, the devices generated temperature and humidity data but the data was never stored or analyzed. How data is stored and analyzed will be covered in later chapters; this chapter will focus on the needs and requirements of IoT solutions that are pertinent to communicating with devices.

What Is Azure IoT Hub?

To overcome the innate challenges of IoT solutions discussed earlier, Microsoft introduced Azure IoT Hub, a fully managed communications service that provides highly secure, dependable, and scalable messaging between IoT devices and IoT solutions. Azure IoT Hub can be viewed as a high-scale gateway that acts as an enabler and manager of all bidirectional communication to and from devices, as shown in Figure 3-2.

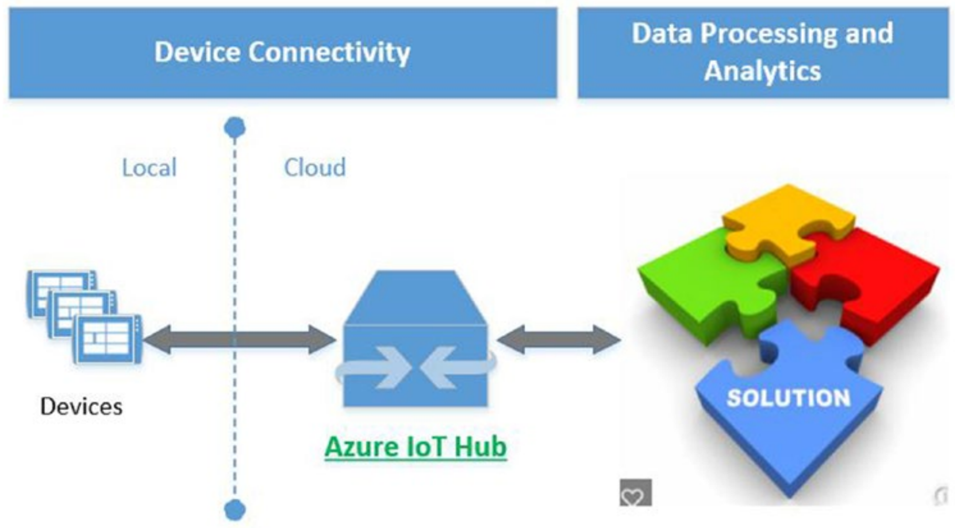


Figure 3-2. *The role of IoT Hub*

At its core, Azure IoT Hub is designed to overcome the device connection limitations and challenges inherent to IoT solutions discussed earlier, including

- Reliable and dependable device-to-cloud and cloud-to-device messaging
- Real-time device registry
- Secure communication via per-device security credentials and access control
- Extensive device connectivity monitoring
- Event monitoring
- Libraries and SDKs available for most languages

Let's explore these benefits in a little more detail.

Why Use Azure IoT Hub?

While there are already existing Azure services that provide device-to-cloud messaging, the benefits of Azure IoT Hub provide the key, essential necessities needed in IoT solutions. The architectural goals and principles that went in to designing and building IoT Hub were founded on four vital areas:

- Support for both hardware and software scenarios, such as the wide collection of devices, environments, and scenarios
- Security should be the foundation of the service across all aspects, including data protection, and device and user identity
- Support for millions of simultaneous connected devices
- Composability to allow for the extension of various components

With these principles and goals in mind, the benefits of Azure IoT hub can be summarized into the following:

- **Scalability**: The ability to support millions of simultaneous connected devices as well as millions of events per second. IoT Hub automatically scales as you add devices.
- **Per-Device Authentication and Security**: Complete, fine-grained control over which devices can access your IoT solution as well as ensuring that any cloud-to-device commands are sent to the correct device.
- **Device Monitoring**: Identify device connectivity issues through detailed operation logs. These logs contain device identity management operations and device connectivity events.
- **Extensibility**: IoT Hub can be extended to provide support for custom protocols. Extensibility through the use of first-party and third-party technologies.
- Support for various languages and platforms through IoT device SDKs and device libraries.

With these benefits as a backdrop to this chapter, an architectural discussion into data flow and communication will be helpful in creating and configuring the IoT Hub later in this chapter and for use in the next chapter.

Architectural Overview

A key element, and one of the challenges discussed earlier, is how IoT devices can connect to IoT solutions. For Azure IoT Hub, devices can connect either directly or indirectly. Figure 3-3 shows the logical communication flow from the different types of devices into the IoT Hub.

IP-capable devices can directly connect using IoT Hub-supported protocols like HTTP, MQTT, AMQPS, and TLS. Devices that do not support these protocols can still connect to Azure IoT Hub via a custom cloud protocol gateway. Those devices that cannot access the Internet directly, such as devices using industry-specific protocols that work on internal networks, can still benefit from IoT Hub by using what is called a field gateway.

AMQP stands for Advanced Message Queueing Protocol and is an open standard application layer protocol finish. MQTT is a machine-to-machine connectivity protocol designed as a lightweight publish/subscribe messaging transport, which makes it very suitable for use with IoT.

A protocol gateway for Azure IoT is simply an open-source framework that provides bidirectional communication between devices. As an open-source framework it supports custom gateways and protocol adaptions, including MQTT.

A field gateway acts as a communication enabler, and is typically an appliance or general purpose software, or Internet-capable device that relays messages from smaller non-Internet-capable devices. They differ from traffic routers in that they manage information and access flow.

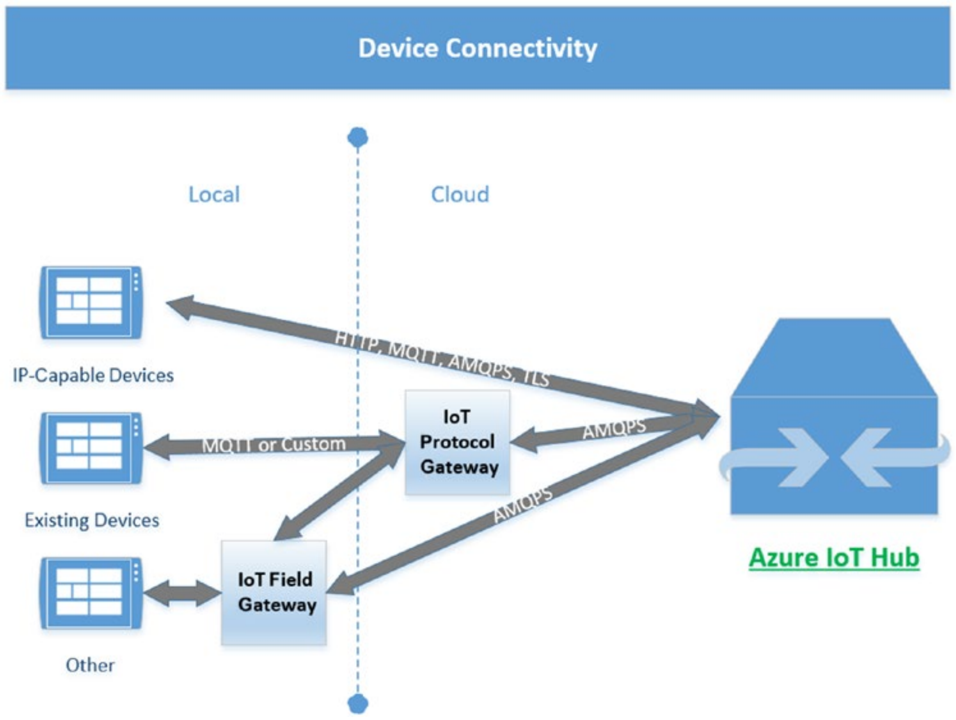


Figure 3-3. *IoT Hub device connectivity*

The architecture described and implemented by Azure IoT Hub enables two specific communication patterns, which have been mentioned, but it's worth spending a bit more time on them.

- **Device-to-cloud**: As mentioned, device-to-cloud communication is refers to data being sent from the device to the cloud. For example, from a Raspberry Pi to Azure IoT Hub.
- **Cloud-to-device**: On the flip side, IoT solutions can use IoT Hub to send messages to individual devices.

Two concepts that frequently come up when talking about device-to-cloud communication are *hot path* and *cold path* processes. Understanding these concepts will help determine the optimum settings for device-to-cloud communications. At a high level, hot path refers to data that needs immediate or near-immediate processing and analysis. For example, as data comes in to IoT Hub, it is immediately picked up by Azure Stream Analytics for real-time analysis. Cold path refers to data stored to be processed later. For example, you might be gathering data that you want to store in a relational database for future processing or analysis.

Creating an IoT Hub

Now that you have a good foundation and understanding of IoT Hub, this section will walk you through creating and configuring an Azure IoT Hub. Here you will create and configure the IoT Hub. In the next chapter, you will hook up the IoT Hub to your Raspberry Pi.

As mentioned in the introduction, all the examples in this book, beginning with this one and throughout the rest of the book, will require a Microsoft Azure subscription. You can create a free, 30-day trial Azure account by visiting `https://azure.microsoft.com/en-us/free/`. If you have a Microsoft MSDN subscription, you can activate Microsoft Azure. Details can be found on your individual MSDN account site.

OK, let's get started. Open your favorite browser and navigate to `http://portal.azure.com/`. When prompted, log in. Once in the portal, click New, select Internet of Things, and then select Azure IoT Hub, as shown in Figure 3-4.

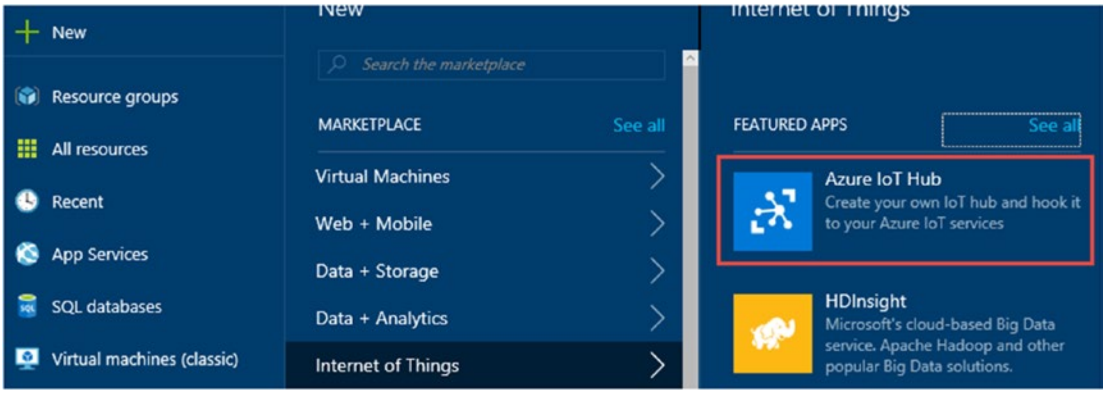


Figure 3-4. *Creating an Azure IoT Hub from the Azure Portal*

The IoT Hub blade will appear, similar to the one in Figure 3-5. In the blade, enter or select the following:

- **Name**: The name of your IoT Hub. Since I am connecting my Raspberry Pi 3 to the hub, I named my hub MyPi3Hub.
- **Pricing and Scale Tier**: Click that section of the blade and select the Free F1 tier.
- **Resource Group**: Either select an existing Resource Group or create a new one.
- **Location**: Select the appropriate region in which to host the IoT Hub.

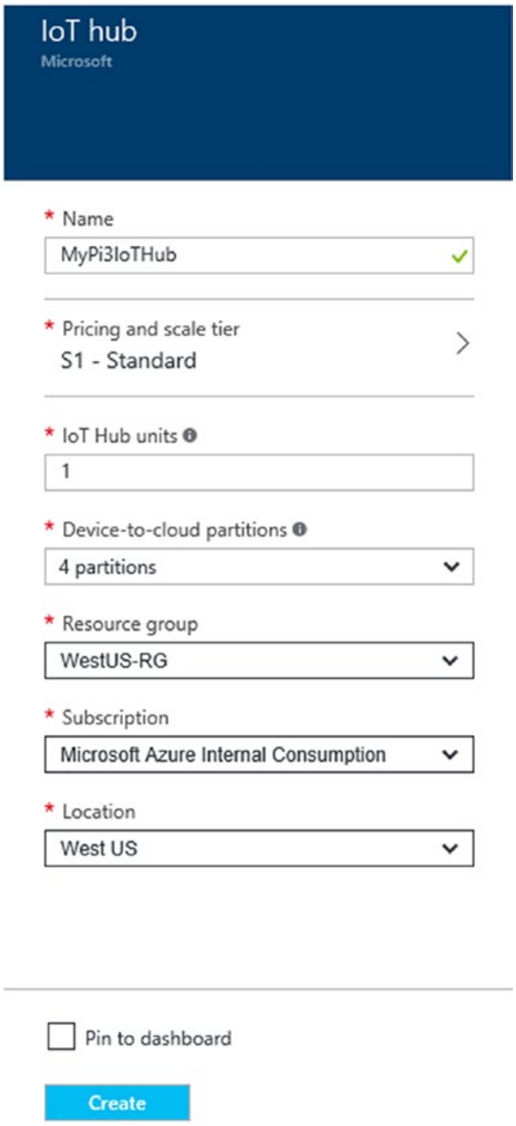
By default, the Pricing and Scale Tier will default to a Standard S1 tier. Clicking in that section of the blade will show that there are three tiers; Free, S1, and S2. The difference between the tiers is the number of messages sent to and from IoT Hub per unit each day. For the purposes of this demo and the book, the Free tier will suffice. Only one Free tier can be created per subscription.

Both the IoT Hub units and device-to-cloud partitions will have default values (one for IoT Hub units and four for device-to-cloud partitions). Leave those values as is, but I'll describe what they are.

IoT Hub units define the performance of your IoT Hub. For example, the free tier provides 8,000 messages per unit, per day. However, with the free tier, you are only allowed a maximum of one unit. The standard S1 and S2 tiers provide up to 200 units. Thus, if you need more performance, increase your number of units within your IoT Hub.

When creating an IoT Hub, your performance sweet spot will be determined by the number of devices connecting to IoT Hub along with how many messages will be processed by IoT Hub. Doing that math will give you an idea of what tier you need, along with the number of units within that tier. You are essentially billed by the number of units in your IoT Hub.

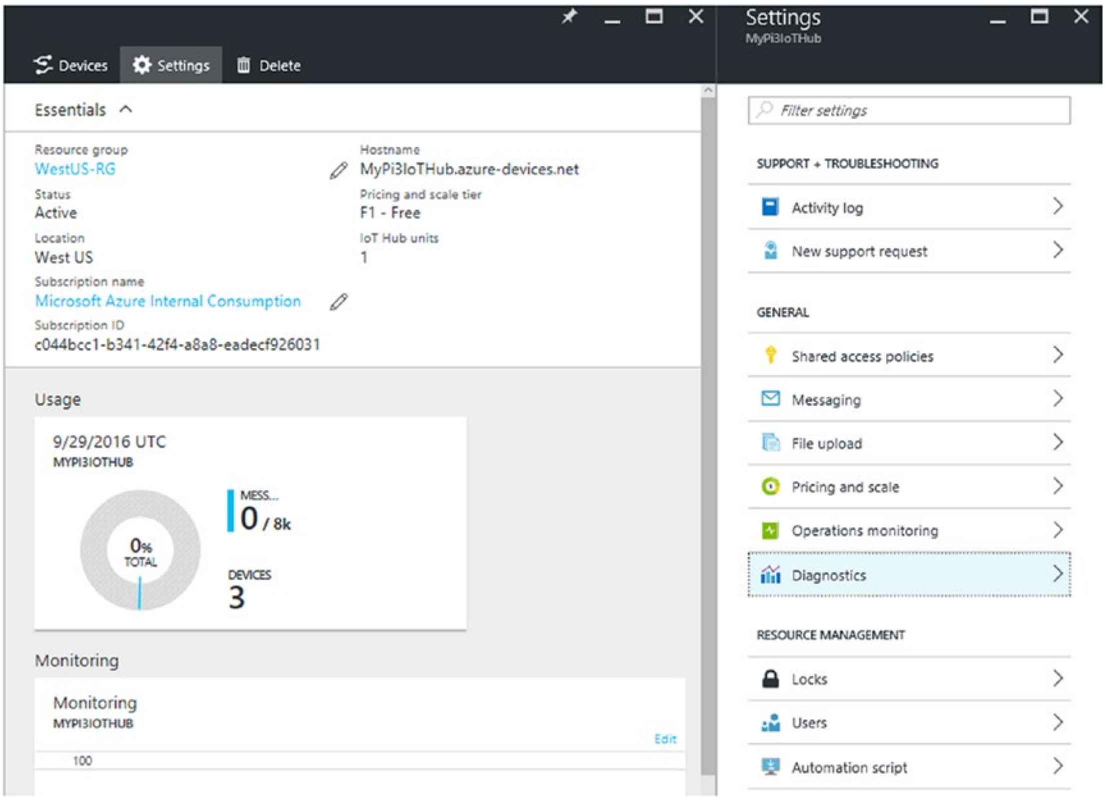
IoT Hub provides message streaming through a partitioned consumer pattern, where a partition is simply an ordered sequence of events. As new messages come in, they are added to the end of the sequence. An IoT Hub can have multiple partitions, each operating independent of other partitions and each containing its own sequence of data, thus growing at different rates.


IoT hub
Microsoft

* Name	MyPi3IoTHub
* Pricing and scale tier	S1 - Standard
* IoT Hub units	1
* Device-to-cloud partitions	4 partitions
* Resource group	WestUS-RG
* Subscription	Microsoft Azure Internal Consumption
* Location	West US

Figure 3-5. *IoT Hub creation settings*

Click the "Pin to dashboard" checkbox and then click Create. The creation of the IoT Hub will take a couple of minutes. Once it is created, you will see a new icon on your dashboard. Click the new IoT Hub icon on your dashboard to open up both the Essentials blade and the Settings blade for the newly created IoT Hub, similar to Figure 3-6.



Resource group
WestUS-RG

Status
Active

Location
West US

Subscription name
Microsoft Azure Internal Consumption

Subscription ID
c044bcc1-b341-42f4-a8a8-eadecf926031

Hostname
MyPi3IoTHub.azure-devices.net

Pricing and scale tier
F1 - Free

IoT Hub units
1

9/29/2016 UTC
MYPI3IOTHUB
0% TOTAL
MESS... 0 / 8k
DEVICES 3

Monitoring
MYPI3IOTHUB
100

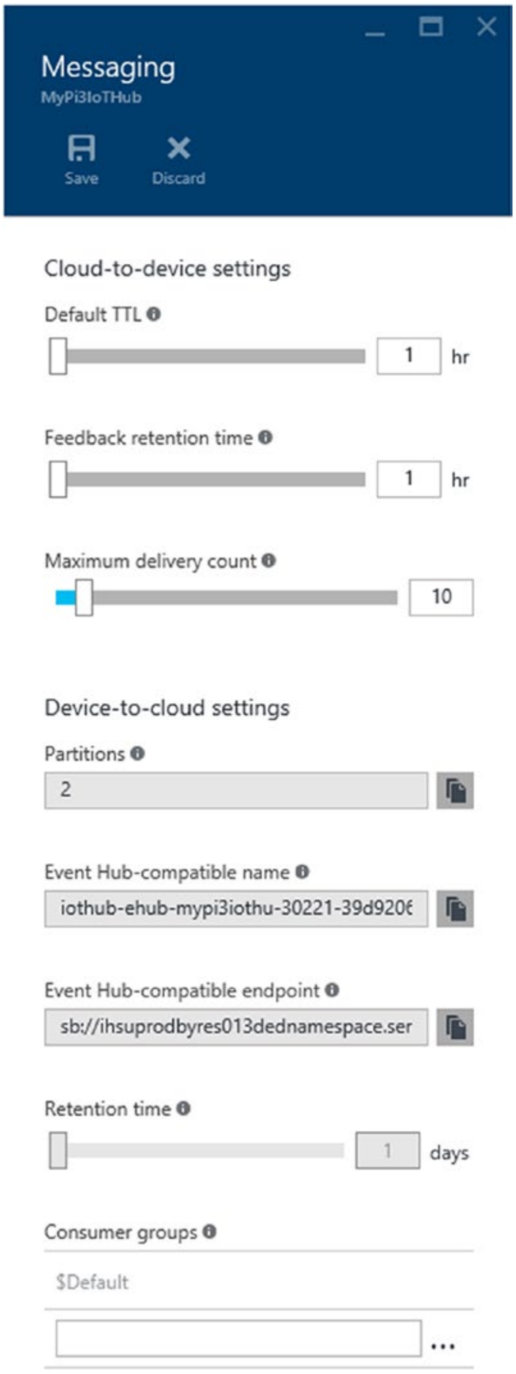
Figure 3-6. *IoT Hub settings*

The left Essentials blade provides a summary of the IoT Hub as well as usage information such as the number of messages processed by the IoT Hub and the number of devices registered. Also included in this is monitoring information which shows all event and connectivity information being collected.

The right Settings blade is where further configuration of items such as messaging, diagnostics, and operations monitoring takes place. The following section will discuss these settings.

Messaging Settings

The Messaging blade contains configuration settings pertaining to cloud-to-device and device-to-cloud message communication, as shown in Figure 3-7.



Messaging
MyPi3IoTHub

Cloud-to-device settings

Default TTL: 1 hr

Feedback retention time: 1 hr

Maximum delivery count: 10

Device-to-cloud settings

Partitions: 2

Event Hub-compatible name: iothub-ehub-mypi3iothu-30221-39d920€

Event Hub-compatible endpoint: sb://ihsuprodbyres013dednamespace.ser

Retention time: 1 days

Consumer groups: $Default

Figure 3-7. *IoT Hub messaging settings*

Most of the settings come with default configuration values but I'll explain these settings.

Cloud-To-Device Settings

- **Default TTL (Time-to-Live)**: Specifies the amount of time (hours) a message is available for the device to consume before it is expired by IoT Hub. Valid values are from 1 to 48.
- **Feedback Retention Time**: Specifies how long in hours the IoT Hub will maintain the feedback for expiration or delivery of cloud-to-device messages. Valid values are from 1 to 48.
- **Maximum Delivery Count**: Specifies the number of times the IoT Hub will attempt to deliver a cloud-to-device message to a device. Valid values are from 1 to 100.

Device-To-Cloud Settings

- **Partitions**: Specifies the number of partitions for your device-to-cloud events.
- **Event Hub-Compatible Name**: Specifies the Event Hub name when using SDKs or other integrations that expect to read from Event Hub.
- **Event Hub-Compatible Endpoint**: Specifies the Event Hub endpoint when using SDKs or other integrations that expect to read from Event Hub.
- **Retention Time**: Specifies how long in days this IoT Hub will maintain device-to-cloud events.
- **Consumer Groups**: Used by applications to pull data from the IoT Hub.

Any time an IoT Hub is created, an Event Hub is also created internally. It is common to use both an Azure IoT Hub and Event Hub in the same IoT solution. For example, the IoT Hub provides the device-to-cloud communication with the Event Hub picking up the real-time processing later in the data processing stages. Thus, it is useful to have access to the Event Hub in some situations.

Consumer groups have been mentioned a few times in this chapter already, and will be mentioned a few more times throughout the book, so a few words on *consumers* and *consumer groups* are in order. A *consumer* is simply any entity (application or process) that reads data from hub (Event Hub or IoT Hub). A *consumer group* is a view of a Hub (Event Hub or IoT Hub) as a whole, including state and position. Consumer groups provide consuming applications, such as Azure Stream Analytics, to have their own independent and separate view into the event stream and read the stream at their own pace. Essentially, consumer groups are used by applications to pull data from the Hub.

Event Hub will be covered in a later chapter, but for now suffice it to say that both are event processing services that enable event and telemetry ingress to the cloud, and both allow you to create 20 consumer groups. However, the main differences are the following:

- IoT Hub provides device-to-cloud and cloud-to-device messaging, whereas Event Hub only supports event ingress (device-to-cloud).
- Both support AMQP and AMQP over WebSockets and HTTP/1, but IoT Hub also works with the IoT Hub Protocol Gateway, which supports custom protocols.
- Both use per-device identity control for secure communication.
- IoT Hub simultaneously supports millions of connected devices, where Event Hub supports a more limited number of connections.

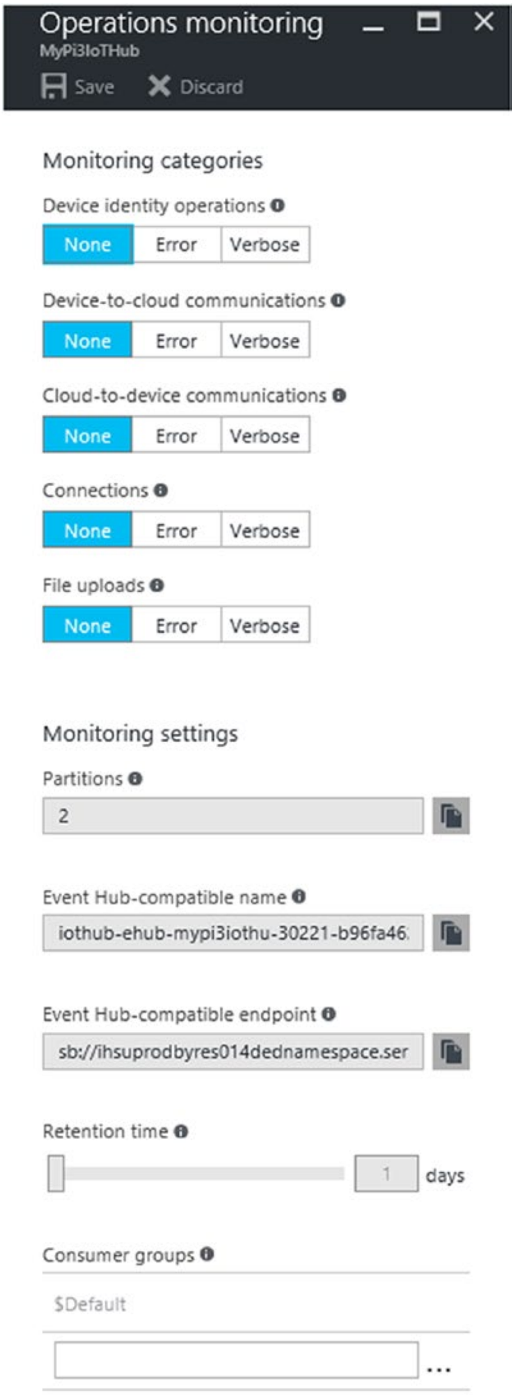
A bit more needs to be said on the custom protocols because this is quite a big topic, one that probably warrants a whole chapter or two by itself. However, for the purposes of this chapter and to build on the custom protocols idea, the Microsoft Azure IoT Protocol Gateway needs some attention. The Microsoft Azure IoT Protocol Gateway is a framework that provides a programming model for building custom protocol adapters for a variety of protocols.

The Microsoft Azure IoT Protocol Gateway provides a number of configuration settings to define the behavior of the protocol, including support for Quality of Service (QoS), subscriptions and topics, message ordering, retry logic, authorization, encryption and decryption, and more. Usually there is no need to adjust the settings but you can if there is a need to adjust the protocols behavior.

You can read more about the Microsoft Azure IoT Protocol Gateway and download it at `https://github.com/Azure/azure-iot-protocol-gateway`.

Operations Monitoring Settings

The Operations Monitoring blade contains configuration settings pertaining to the type of information and events monitored by IoT Hub, as shown in Figure 3-8.



Monitoring categories

Device identity operations: [✓ None] Error Verbose

Device-to-cloud communications: [✓ None] Error Verbose

Cloud-to-device communications: [✓ None] Error Verbose

Connections: [✓ None] Error Verbose

File uploads: [✓ None] Error Verbose

Monitoring settings

Partitions: 2

Event Hub-compatible name: iothub-ehub-mypi3iothu-30221-b96fa46.

Event Hub-compatible endpoint: sb://ihsuprodbyres014dednamespace.ser

Retention time: 1 days

Consumer groups: $Default

Figure 3-8. *IoT Hub operations monitoring settings*

Monitoring Categories

- **Device Identity Operations**: Logs errors related to operations on the device identity registry.
- **Device-to-Cloud Communications**: Logs errors related to device-to-cloud messaging.
- **Cloud-to-Device Communications**: Logs errors related to cloud-to-device messaging.
- **Connections**: Logs errors when a device connects or disconnects from the IoT Hub.

Each IoT Hub has a device identity registry that is available to use for creating per-device resources in the IoT Hub service. For example, it might be used as a queue to store up cloud-to-device messages to be sent to the device later.

When a user attempts to create, update, or delete an entry in the IoT Hub identity registry, the device identity operations track this information. This information comes in handy when you want to monitor certain scenarios such as device provisioning.

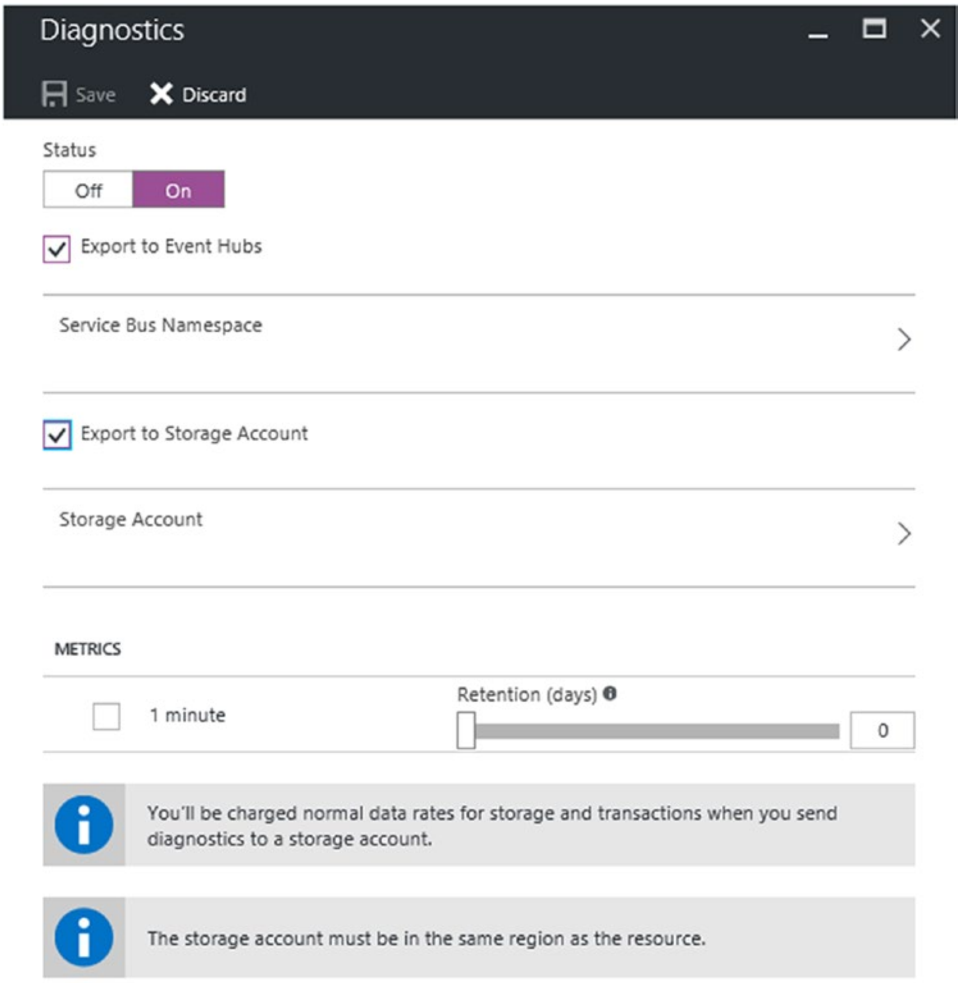
Monitoring Settings

- **Partitions**: Specifies the number of partitions for your operations monitoring events.
- **Event Hub-Compatible Name**: Specifies the Event Hub name when using SDKs or other integrations that expect to read from Event Hub.
- **Event Hub-Compatible Endpoint**: Specifies the Event Hub endpoint when using SDKs or other integrations that expect to read from Event Hub.
- **Retention Time**: Specifies how long in days this IoT Hub will maintain operation monitoring events.
- **Consumer Groups**: Used by applications to pull data from the IoT Hub.

Notice that the Event Hub settings are similar to the information on the Messaging blade. This is simply because one is for messaging and the other is for monitoring. Similar details are needed.

Diagnostics Settings

The Diagnostics blade contains configuration settings pertaining to diagnostics for your Azure IoT Hub. By default, diagnostics is turned off, which disables monitoring charts and alerts for your resource. Turning diagnostics on will allow you to configure some diagnostics settings, as shown in Figure 3-9.



Status
Off [✓ On]

[✓] Export to Event Hubs

Service Bus Namespace

[✓] Export to Storage Account

Storage Account

METRICS

☐	1 minute	Retention (days) 0

You'll be charged normal data rates for storage and transactions when you send diagnostics to a storage account.

The storage account must be in the same region as the resource.

Figure 3-9. *IoT Hub diagnostic settings*

The diagnostics help provide better insight on the overall state of the resources in IoT Hub, such as the health of the services and the devices connected to your IoT Hub. From this information you can glean root-cause issues and proactively see what is happening within IoT Hub.

The first thing to do is to select what to do with the diagnostics data. Your options are to archive it to an Azure storage account, stream it to an Azure event hub, or send it to Log Analytics. From there, you'll need to specify the storage account, event, hub, or log analytics account. You can choose all three if you'd like. You are not limited to selecting a single option.

Other Settings

Wrapping up this section, there are a few more blades to cover.

Shared Access Policies

The Shared Access Policies blade allows you to define a set of permissions to endpoints. These policies can be applied to services, devices, and applications to define a fine-grained access control. When a device connects to IoT Hub, IoT Hub will authenticate the endpoints by verifying a token to both the defined shared access policies as well as the device identity registry security credentials.

New shared access policies are created by simply providing a name to the policy and then specifying the permissions: registry read, registry write, service connect, and device connect.

Pricing and Scale

The Pricing and Scale blade is similar and common to all services in Azure. It simply allows you to pick the performance level of Azure IoT Hub. Currently there are three tiers: Free, Standard S1, and Standard S2. The difference between then is simply how many messages you plan to send to the corresponding IoT Hub.

A couple of items to pay attention to: first, only one free tier can be created per IoT Hub. Second, IoT Hub does not let you transition between the free and standard tier. Meaning, if you have a free tier and want to upgrade it to a standard tier, this is not possible. The blade will inform you of that.

Summary

The goal of this chapter was to provide insight into the challenges of connecting devices to IoT solutions and how IoT Hub overcomes these challenges, including security and performance. This chapter provided an overview of Azure IoT Hub and then looked at why Azure IoT Hub is a viable and solid solution for solving the challenges and problems that IoT solutions face today.

Lastly, this chapter covered the creation and configuration of an Azure IoT Hub, including the different options and settings and what you need to know when creating and configuring your Azure IoT Hub.

However, there is more to do! So in the next chapter you will add your device to Azure IoT Hub and start sending messages.