**Overview of Azure IoT Suite**

The Azure internet of things (IoT) services offer a broad range of capabilities. These enterprise grade services enable you to:

* Collect data from devices
* Analyze data streams in-motion
* Store and query large data sets
* Visualize both real-time and historical data
* Integrate with back-office systems

To deliver these capabilities, Azure IoT Suite packages together multiple Azure services with custom extensions as *preconfigured solutions*. These preconfigured solutions are base implementations of common IoT solution patterns that help to reduce the time you take to deliver your IoT solutions. Using the [IoT software development kits](https://azure.microsoft.com/documentation/articles/iot-hub-sdks-summary/), you can customize and extend these solutions to meet your own requirements. You can also use these solutions as examples or templates when you are developing new IoT solutions.

The following video provides an introduction to Azure IoT Suite:

[[](https://azure.microsoft.com/en-us/documentation/videos/azurecon-2015-introducing-the-microsoft-azure-iot-suite/)09-29-201515 min, 20 sec](https://azure.microsoft.com/en-us/documentation/videos/azurecon-2015-introducing-the-microsoft-azure-iot-suite/)

Azure IoT services in Azure IoT Suite

The preconfigured solutions typically use the following services:

* Core to Azure IoT Suite is the [Azure IoT Hub](https://azure.microsoft.com/documentation/services/iot-hub/) service. This service provides the device-to-cloud and cloud-to-device messaging capabilities and acts as the gateway to the cloud and the other key IoT Suite services. The service enables you to receive messages from your devices at scale, and send commands to your devices.
* [Azure Stream Analytics](https://azure.microsoft.com/documentation/services/stream-analytics/) provides in-motion data analysis. IoT Suite leverages this service to process incoming telemetry, perform aggregation, and detect events. The preconfigured solutions also use stream analytics to process informational messages that contain data such as metadata or command responses from devices. The solutions use Stream Analytics to process the messages from your devices and deliver those messages to other services.
* [Azure Storage](https://azure.microsoft.com/documentation/services/storage/) and [Azure DocumentDB](https://azure.microsoft.com/documentation/services/documentdb/) provide the data storage capabilities. The preconfigured solutions use blob storage to store telemetry and to make it available for analysis. The solutions use DocumentDB to store device metadata and enable the device management capabilities of the solutions.
* [Azure Web Apps](https://azure.microsoft.com/documentation/services/app-service/web/) and [Microsoft Power BI](https://powerbi.microsoft.com/) provide the data visualization capabilities. The flexibility of Power BI enables you to quickly build your own interactive dashboards that use IoT Suite data.

For an overview of the architecture of a typical IoT solution, see [Microsoft Azure and the Internet of Things (IoT)](https://azure.microsoft.com/en-us/documentation/articles/iot-suite-what-is-azure-iot/).

## Azure IoT Suite

The Microsoft Azure IoT Suite is an enterprise-grade solution that enables you to get started quickly through a set of extensible preconfigured solutions that address common IoT scenarios, such as [remote monitoring](https://azure.microsoft.com/en-us/documentation/articles/iot-suite-getstarted-preconfigured-solutions/) and [predictive maintenance](https://azure.microsoft.com/en-us/documentation/articles/iot-suite-predictive-overview/). These solutions are implementations of the IoT solution architecture described previously.

The preconfigured solutions are complete, working, end-to-end solutions that include simulated devices to get you started, preconfigured Azure services such as [Azure IoT Hub](https://azure.microsoft.com/services/iot-hub/), [Azure Event Hubs](https://azure.microsoft.com/services/event-hubs/), [Azure Stream Analytics](https://azure.microsoft.com/services/stream-analytics/), [Azure Machine Learning](https://azure.microsoft.com/services/machine-learning/), and [Azure storage](https://azure.microsoft.com/services/storage/), and solution specific management consoles. The preconfigured solutions contain proven, production-ready code that you can customize and extend to implement your own specific IoT scenarios.

You may also be interested in the [Azure IoT Hub](https://azure.microsoft.com/services/iot-hub/) service that many of the preconfigured solutions use. [Azure IoT Hub](https://azure.microsoft.com/services/iot-hub/) provides the secure and reliable bi-directional communications between devices and the cloud used in the preconfigured solution architecture.

IoT Suite includes preconfigured solutions that enable you to quickly get started with and to explore the common IoT scenarios, such as *Remote monitoring* and *Predictive maintenance*, that Azure IoT Suite makes possible. You can deploy these solutions to your Azure subscription and then run a complete, end-to-end IoT scenario.

The Azure IoT Suite preconfigured solutions are implementations of common IoT solution patterns that you can deploy to Azure using your subscription. You can use the preconfigured solutions:

* As a starting point for your own IoT solutions.
* To learn about common patterns in IoT solution design and development.

Each preconfigured solution is a complete, end-to-end implementation using simulated devices to generate telemetry.

In addition to deploying and running the solutions in Azure, you can download the complete source code and then customize and extend the solution to meet your specific IoT requirements.

##### **Note:**

To deploy one of the preconfigured solutions, visit [Microsoft Azure IoT Suite](https://www.azureiotsuite.com/). The article[Get started with the IoT preconfigured solutions](https://azure.microsoft.com/en-us/documentation/articles/iot-suite-getstarted-preconfigured-solutions/) provides more informatin about how to deploy and run one of the solutions.

The following table shows how the solutions map to specific IoT features:

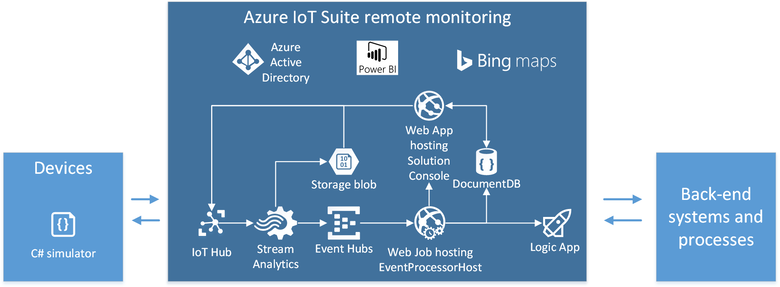
| Solution | Data Ingestion | Device Identity | Command and Control | Rules and Actions | Predictive Analytics |
| --- | --- | --- | --- | --- | --- |
| [Remote monitoring](https://azure.microsoft.com/en-us/documentation/articles/iot-suite-remote-monitoring-sample-walkthrough/) | Yes | Yes | Yes | Yes | - |
| [Predictive maintenance](https://azure.microsoft.com/en-us/documentation/articles/iot-suite-predictive-overview/) | Yes | Yes | Yes | Yes | Yes |

* Data ingestion: Ingress of data at scale to the cloud.
* Device identity: Manage unique identities of every connected device.
* Command and control: Send messages to a device from the cloud to cause the device to take some action.
* Rules and actions: The solution back end uses rules to act on specific device-to-cloud data.
* Predictive analytics: The solution back end applies analyzes device-to-cloud data to predict when specific actions should take place. For example, analyzing aircraft engine telemetry to determine when engine maintenance is due.

## Remote Monitoring preconfigured solution overview

We have chosen to discuss the remote monitoring preconfigured solution in this article because it illustrates many common design elements that the other solutions share.

The following diagram illustrates the key elements of the remote monitoring solution. The sections below provide more information about these elements.



## Devices

When you deploy the remote monitoring preconfigured solution, four simulated devices are pre-provisioned in the solution that simulate a cooling device. These simulated devices have a built in temperature and humidity model that emits telemetry. These simulated devices are included to illustrate the end-to-end flow of data through the solution, and to provide a convenient source of telemetry and a target for commands if you are a back-end developer using the solution as a starting point for a custom implementation.

When a device first connects to IoT Hub in the remote monitoring preconfigured solution, the device information message sent to the IoT hub enumerates the list of commands that the device can respond to. In the remote monitoring preconfigured solution, the commands are:

* Ping Device: The device responds to this command with an acknowledgement. This is useful for checking that the device is still active and listening.
* Start Telemetry: Instructs the device to start sending telemetry.
* Stop Telemetry: Instructs the device to stop sending telemetry.
* Change Set Point Temperature: Controls the simulated temperature telemetry values the device sends. This is useful for testing back-end logic.
* Diagnostic Telemetry: Controls if the device should send the external temperature as telemetry.
* Change Device State.: Sets the device state metadata property that the device reports. This is useful for testing back-end logic.

You can add more simulated devices to the solution that emit the same telemetry and respond to the same commands.

## IoT Hub

In this preconfigured solution, the IoT Hub instance corresponds to the Cloud Gateway in a typical [IoT solution architecture](https://azure.microsoft.com/en-us/documentation/articles/iot-suite-what-is-azure-iot/).

An IoT hub receives telemetry from the devices at a single endpoint. An IoT hub also maintains device specific endpoints where each devices can retrieve the commands that are sent to it.

The IoT hub makes the received telemetry available through the service-side telemetry read endpoint.

## Azure Stream Analytics

The preconfigured solution uses three [Azure Stream Analytics](https://azure.microsoft.com/documentation/services/stream-analytics/) (ASA) jobs to filter the telemetry stream from the devices:

* DeviceInfo job - outputs data to an Event hub that routes device registration specific messages, sent when a device first connects or in response to a **Change device state**command, to the solution device registry (a DocumentDB database).
* Telemetry job - sends all raw telemetry to Azure blob storage for cold storage and calculates telemetry aggregations that display in the solution dashboard.
* Rules job - filters the telemetry stream for values that exceed any rule thresholds and outputs the data to an Event hub. When a rule fires, the solution portal dashboard view displays this event as a new row in the alarm history table and triggers an action based on the settings defined on the Rules and Actions views in the solution portal.

In this preconfigured solution, the ASA jobs form part of to the **IoT solution back end** in a typical [IoT solution architecture](https://azure.microsoft.com/en-us/documentation/articles/iot-suite-what-is-azure-iot/).

## Event processor

In this preconfigured solution, the event processor forms part of the **IoT solution back end** in a typical [IoT solution architecture](https://azure.microsoft.com/en-us/documentation/articles/iot-suite-what-is-azure-iot/).

The **DeviceInfo** and **Rules** ASA jobs send their output to Event hubs for delivery to other back end services. The solution uses an [EventPocessorHost](https://azure.microsoft.com/en-us/documentation/articles/event-hubs-programming-guide/#event-processor-host) instance, running in a [WebJob](https://azure.microsoft.com/en-us/documentation/articles/web-sites-create-web-jobs/), to read the messages from these Event hubs. The **EventProcessorHost** uses the **DeviceInfo** data to update the device data in the DocumentDB database, and uses the **Rules** data to invoke the Logic app and update the alerts display in the solution portal.

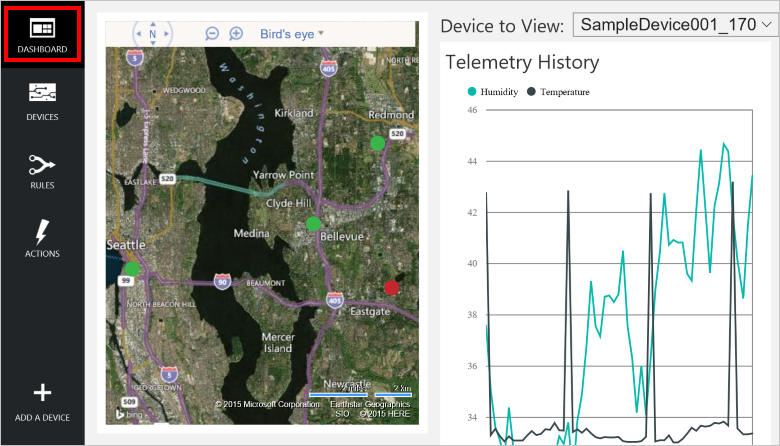
## Device identity registry and DocumentDB

Every IoT hub includes a [device identity registry](https://azure.microsoft.com/en-us/documentation/articles/iot-hub-devguide/#device-identity-registry) that stores device keys. IoT Hub uses this information authenticate devices - a device must be registered and have a valid key before it can connect to the hub.

This solution stores additional information about devices such as their state, the commands they support, and other metadata. The solution uses a DocumentDB database to store this solution-specific device data and the solution portal retrieves data from this DocumentDB database for display and editing.

The solution must also keep the information in the device identity registry synchronized with the contents of the DocumentDB database. The **EventProcessorHost** uses the data from**DeviceInfo** stream analytics job to manage the synchronization.

## Solution portal



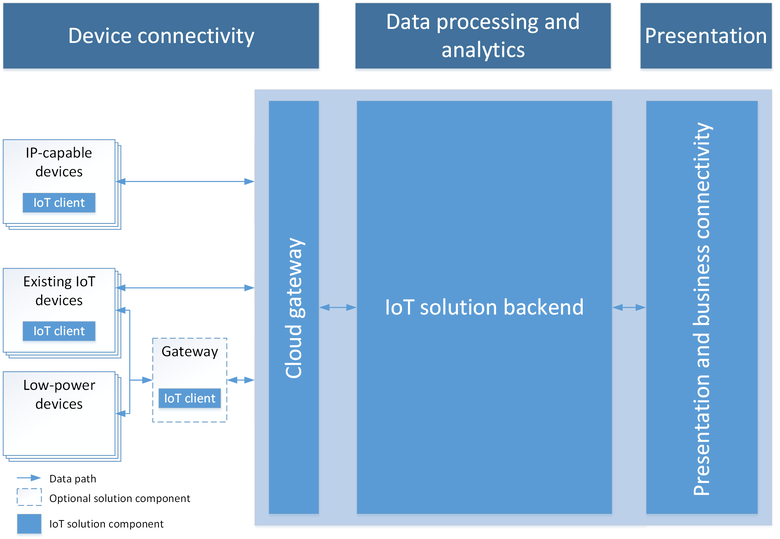
The solution portal is a web-based UI that is deployed to the cloud as part of the preconfigured solution. It enables you to:

* View telemetry and alarm history in a dashboard.
* Provision new devices.
* Manage and monitor devices.
* Send commands to specific devices.
* Manage rules and actions.

In this preconfigured solution, the solution portal forms part of the **IoT solution back end** and part of the **Processing and business connectivity** in a typical [IoT solution architecture](https://azure.microsoft.com/en-us/documentation/articles/iot-suite-what-is-azure-iot/).

## IoT solution architecture

The following diagram shows a typical IoT solution architecture. Note that it does not include the names of any specific Azure services, but describes the key elements in a generic IoT solution architecture. In this architecture, IoT devices collect data which they send to a cloud gateway. The cloud gateway makes the data available for processing by other back-end services from where data is delivered to other line-of-business applications or to human operators through a dashboard or other presentation device.



##### **Note:**

For an in-depth discussion of IoT architecture see the [Microsoft Azure IoT Reference Architecture](http://download.microsoft.com/download/A/4/D/A4DAD253-BC21-41D3-B9D9-87D2AE6F0719/Microsoft_Azure_IoT_Reference_Architecture.pdf).

### Device connectivity

In this IoT solution architecture, devices send telemetry, such as sensor readings from a pumping station, to a cloud endpoint for storage and processing. In a predictive maintenance scenario, the back end might use the stream of sensor data to determine when a specific pump requires maintenance. Devices can also receive and respond to cloud-to-device commands by reading messages from a cloud endpoint. For example, in the predictive maintenance scenario the solution back end might send commands to other pumps in the pumping station to begin re-routing flows just before maintenance is due to start to make sure the maintenance engineer can get started as soon as she arrives.

One of the biggest challenges facing IoT projects is how to reliably and securely connect devices to the solution back end to enable the device to send telemetry and retrieve commands. IoT devices have different characteristics as compared to other clients such as browsers and mobile apps. IoT devices:

* Are often embedded systems with no human operator.
* Can be deployed in remote locations, where physical access is very expensive.
* May only be reachable through the solution back end. There is no other way to interact with the device.
* May have limited power and processing resources.
* May have intermittent, slow, or expensive network connectivity.
* May need to use proprietary, custom, or industry specific application protocols.
* Can be created using a large set of popular hardware and software platforms.

In addition to the requirements above, any IoT solution must also deliver scale, security, and reliability. The resulting set of connectivity requirements is hard and time-consuming to implement using traditional technologies such as web containers and messaging brokers. Azure IoT Hub and the IoT Device SDKs make it easier to implement solutions that meet these requirements.

A device can communicate directly with a cloud gateway endpoint, or if the device cannot use any of the communications protocols that the cloud gateway supports, it can connect through an intermediate gateway, such as the [IoT Hub protocol gateway](https://azure.microsoft.com/en-us/documentation/articles/iot-hub-protocol-gateway/), that performs protocol translation. For example, from the Common Industrial Protocol (CIP) to AMQPS.

### Data processing and analytics

In the cloud, an IoT solution back end is where most of the data processing in the solution occurs, in particular filtering and aggregating telemetry and routing it to other services. The IoT solution back end:

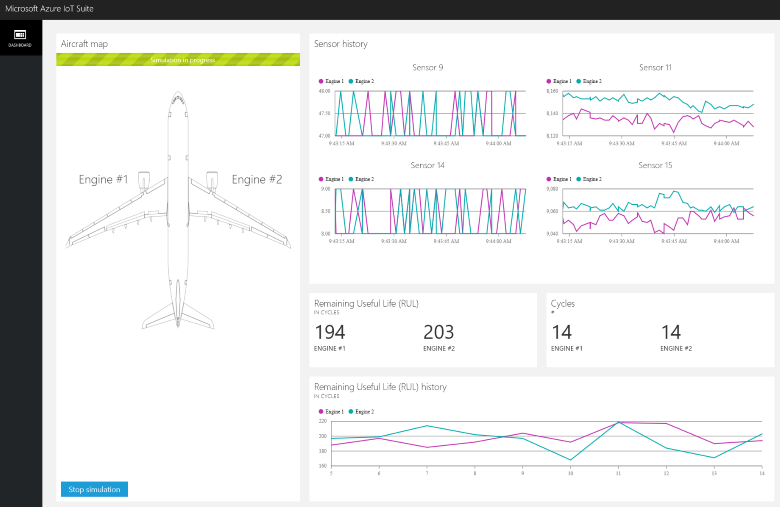
* Receives telemetry at scale from your devices and determines how to process and store that data.
* May enable you to send commands from the cloud to specific device.
* Provides device registration capabilities that enable you to provision devices and to control which devices are permitted to connect to your infrastructure.
* Enables you to track the state of your devices and monitor their activities.

In the predictive maintenance scenario, the solution back end stores historical telemetry data to use to identify patterns and analyzes telemetry as it arrives to spot the patterns that indicate maintenance is due on a specific pump.

IoT solutions can include automatic feedback loops. For example, an analytics module in the back end can identify from telemetry that the temperature of a specific device is above normal operating levels and then send a command to the device, instructing it to take corrective action.

### Presentation and business connectivity

The presentation and business connectivity layer allows end users to interact with the IoT solution and the devices. It enables users to view and analyze the data collected from their devices. These views can take the form of dashboards or BI reports that can display both historical data or near real-time data. For example, an operator can check on the status of particular pumping station and see any alerts raised by the system. This layer also allows integration of the IoT solution back end with existing line-of-business applications to tie into enterprise business processes or workflows. For example, the predictive maintenance solution can integrate with a scheduling system that books an engineer to visit a pumping station when the solution identifies a pump in need of maintenance.



**Security**

When designing a system, it is important to understand the potential threats to that system, and add appropriate defenses accordingly, as the system is designed and architected. It is particularly important to design the product from the start with security in mind because understanding how an attacker might be able to compromise a system helps make sure appropriate mitigations are in place from the beginning.

Security starts with a threat model

Microsoft has long used threat models for its products and has made the company’s threat modeling process publically available . The company experience demonstrates that the modelling has unexpected benefits beyond the immediate understanding of what threats are the most concerning. For example, it also creates an avenue for an open discussion with others outside the development team, which can lead to new ideas and improvements in the product.

The objective of threat modeling is to understand how an attacker might be able to compromise a system and then make sure appropriate mitigations are in place. Threat modeling forces the design team to consider mitigations as the system is designed rather than after a system is deployed. This fact is critically important, because retrofitting security defenses to a myriad of devices in the field is infeasible, error prone and will leave customers at risk.

Many development teams do an excellent job capturing the functional requirements for the system that benefit customers. However, identifying non-obvious ways that someone might misuse the system is more challenging. Threat modeling can help development teams understand what an attacker might do and why. Threat modeling is a structured process that creates a discussion about the security design decisions in the system, as well as changes to the design that are made along the way that impact security. While a threat model is simply a document, this documentation also represents an ideal way to ensure continuity of knowledge, retention of lessons learned, and help new team onboard rapidly. Finally, an outcome of threat modeling is to enable you to consider other aspects of security, such as what security commitments you wish to provide to your customers. These commitments in conjunction with threat modeling will inform and drive testing of your IoT solution.

When to threat model

[Threat modeling](http://www.microsoft.com/security/sdl/adopt/threatmodeling.aspx) offers the greatest value if it is incorporated into the design phase. When you are designing, you have the greatest flexibility to make changes to eliminate threats. Eliminating threats by design is the desired outcome. It is much easier than adding mitigations, testing them, and ensuring they remain current and moreover, such elimination is not always possible. It becomes harder to eliminate threats as a product becomes more mature, and in turn will ultimately require more work and a lot more hard tradeoffs than threat modeling early on in the development.

What to threat model

You should thread model the solution as a whole and also focus in the following areas:

* The security and privacy features
* The features whose failures are security relevant
* The features that touch a trust boundary

Who threat models

Threat modeling is a process like any other. It is a good idea to treat the threat model document like any other component of the solution and validate it. Many development teams do an excellent job capturing the functional requirements for the system that benefit customers. However, identifying non-obvious ways that someone might misuse the system is more challenging. Threat modeling can help development teams understand what an attacker might do and why.

How to threat model

The threat modeling process is composed of four steps; the steps are:

* Model the application
* Enumerate Threats
* Mitigate threats
* Validate the mitigations

The process steps

Three rules of thumb to keep in mind when building a threat model:

1. Create a diagram out of reference architecture.
2. Start breadth-first. Get an overview, and understand the system as a whole, before deep-diving. This helps ensure that you deep-dive in the right places.
3. Drive the process, don’t let the process drive you. If you find an issue in the modeling phase and want to explore it, go for it! Don’t feel you need to follow these steps slavishly.

Threats

The four core elements of a threat model are:

* Processes (web services, Win32 services, \*nix daemons, etc. Note that some complex entities (e.g. field gateways and sensors) can be abstracted as a process when a technical drill down in these areas is not possible.
* Data stores (anywhere data is stored, such as a configuration file or database)
* Data flow (where data moves between other elements in the application)
* External Entities (anything that interacts with the system, but is not under the control of the application, examples include users and satellite feeds)

All elements in the architectural diagram are subject to various threats; we will use the STRIDE mnemonic. Read [Threat Modeling Again, STRIDE](https://blogs.msdn.microsoft.com/larryosterman/2007/09/04/threat-modeling-again-stride/) to know more about the STRIDE elements.

Different elements of the application diagram are subject to certain STRIDE threats:

* Processes are subject to STRIDE
* Data flows are subject to TID
* Data stores are subject to TID, and sometimes R, if the data stores are log files.
* External entities are subject to SRD

Security in IoT

Connected special-purpose devices have a significant number of potential interaction surface areas and interaction patterns, all of which must be considered to provide a framework for securing digital access to those devices. The term “digital access” is used here to distinguish from any operations that are carried out through direct device interaction where access security is provided through physical access control. For example, putting the device into a room with a lock on the door. While physical access cannot be denied using software and hardware, measures can be taken to prevent physical access from leading to system interference.

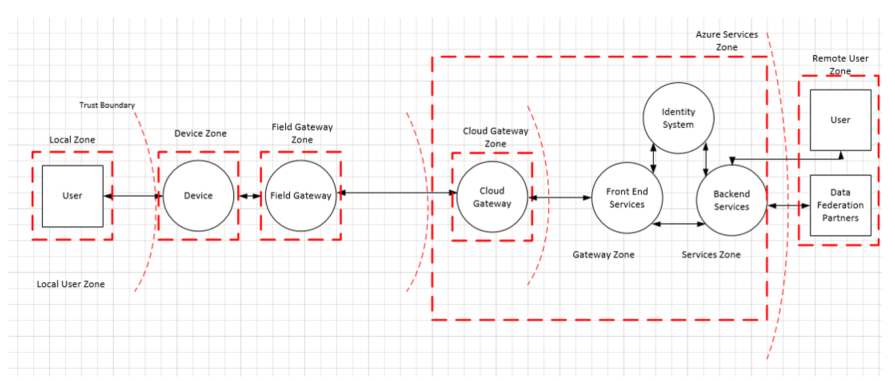
As we explore the interaction patterns, we will look at “device control” and “device data” with the same level of attention. “Device control” can be classified as any information that is provided to a device by any party with the goal of changing or influencing its behavior towards its state or the state of its environment. “Device data” can be classified as any information that a device emits to any other party about its state and the observed state of its environment.

In order to optimize security best practices, it is recommend that a typical IoT architecture be divided into several component/zones as part of the threat modeling exercise. These zones are described fully throughout this section and include:

* Device,
* Field Gateway,
* Cloud gateways, and
* Services.

Zones are broad way to segment a solution; each zone often has its own data and authentication and authorization requirements. Zones can also be used to isolation damage and restrict the impact of low trust zones on higher trust zones.

Each zone is separated by a Trust Boundary, which is noted as the dotted red line in the diagram below. It represents a transition of data/information from one source to another. During this transition, the data/information could be subject to Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege (STRIDE).



The components depicted within each boundary are also subjected to STRIDE, enabling a full 360 threat modeling view of the solution. The sections below elaborate on each of the components and specific security concerns and solutions that should be put into place.

The sections that follows will discuss standard components typically found in these zones.

The Device Zone

The device environment is the immediate physical space around the device where physical access and/or “local network” peer-to-peer digital access to the device is feasible. A “local network” is assumed to be a network that is distinct and insulated from – but potentially bridged to – the public Internet, and includes any short-range wireless radio technology that permits peer-to-peer communication of devices. It does *not* include any network virtualization technology creating the illusion of such a local network and it does also not include public operator networks that require any two devices to communicate across public network space if they were to enter a peer-to-peer communication relationship.

The Field Gateway Zone

Field gateway is a device/appliance or some general-purpose server computer software that acts as communication enabler and, potentially, as a device control system and device data processing hub. The field gateway zone includes the field gateway itself and all devices that are attached to it. As the name implies, field gateways act outside dedicated data processing facilities, are usually location bound, are potentially subject to physical intrusion, and will have limited operational redundancy. All to say that a field gateway is commonly a thing one can touch and sabotage while knowing what its function is.

A field gateway is different from a mere traffic router in that it has had an active role in managing access and information flow, meaning it is an application addressed entity and network connection or session terminal. An NAT device or firewall, in contrast, do not qualify as field gateways since they are not explicit connection or session terminals, but rather a route (or block) connections or sessions made through them. The field gateway has two distinct surface areas. One faces the devices that are attached to it and represents the inside of the zone, and the other faces all external parties and is the edge of the zone.

The cloud gateway zone

Cloud gateway is a system that enables remote communication from and to devices or field gateways from several different sites across public network space, typically towards a cloud-based control and data analysis system, a federation of such systems. In some cases, a cloud gateway may immediately facilitate access to special-purpose devices from terminals such as tablets or phones. In the context discussed here, “cloud” is meant to refer to a dedicated data processing system that is not bound to the same site as the attached devices or field gateways. Also in a Cloud Zone, operational measures prevent targeted physical access and is not necessarily exposed to a “public cloud” infrastructure.

A cloud gateway may potentially be mapped into a network virtualization overlay to insulate the cloud gateway and all of its attached devices or field gateways from any other network traffic. The cloud gateway itself is neither a device control system nor a processing or storage facility for device data; those facilities interface with the cloud gateway. The cloud gateway zone includes the cloud gateway itself along with all field gateways and devices directly or indirectly attached to it. The edge of the zone is a distinct surface area where all external parties communicate through.

The services zone

A “service” is defined for this context as any software component or module that is interfacing with devices through a field- or cloud gateway for data collection and analysis, as well as for command and control. Services are mediators. They act under their identity towards gateways and other subsystems, store and analyze data, autonomously issue commands to devices based on data insights or schedules and expose information and control capabilities to authorized end-users.

Information-devices vs. special-purpose devices

PCs, phones, and tablets are primarily interactive information devices. Phones and tablets are explicitly optimized around maximizing battery lifetime. They preferably turn off partially when not immediately interacting with a person, or when not providing services like playing music or guiding their owner to a particular location. From a systems perspective, these information technology devices are mainly acting as proxies towards people. They are “people actuators” suggesting actions and “people sensors” collecting input.

Special-purpose devices, from simple temperature sensors to complex factory production lines with thousands of components inside them, are different. These devices are much more scoped in purpose and even if they provide some user interface, they are largely scoped to interfacing with or be integrated into assets in the physical world. They measure and report environmental circumstances, turn valves, control servos, sound alarms, switch lights, and do many other tasks. They help to do work for which an information device is either too generic, too expensive, too big, or too brittle. The concrete purpose immediately dictates their technical design as well the available monetary budget for their production and scheduled lifetime operation. The combination of these two key factors constrains the available operational energy budget, physical footprint, and thus available storage, compute, and security capabilities.

If something “goes wrong” with automated or remote controllable devices, for example, physical defects or control logic defects to willful unauthorized intrusion and manipulation. The production lots may be destroyed, buildings may be looted or burned down, and people may be injured or even die. This is, of course, a whole different class of damage than someone maxing out a stolen credit card's limit. The security bar for devices that make things move, and also for sensor data that eventually results in commands that cause things to move, must be higher than in any e-commerce or banking scenario.

Device control and device data interactions

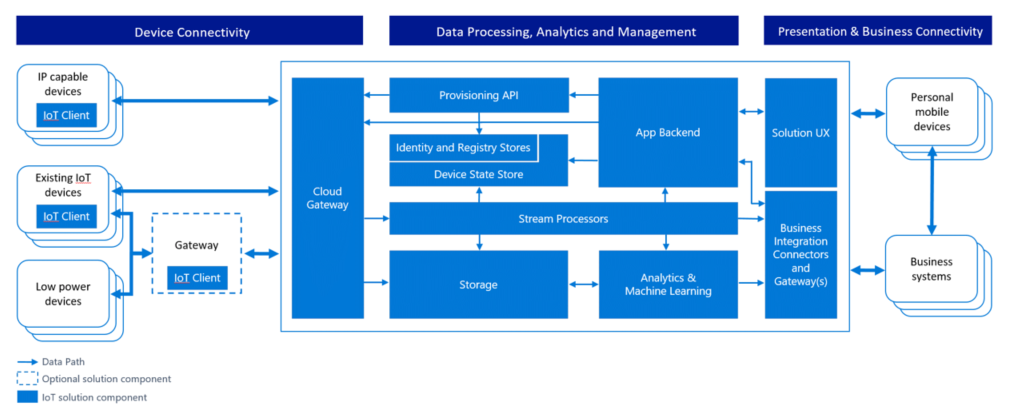
Connected special-purpose devices have a significant number of potential interaction surface areas and interaction patterns, all of which must be considered to provide a framework for securing digital access to those devices. The term “digital access” is used here to distinguish from any operations that are carried out through direct device interaction where access security is provided through physical access control. For example, putting the device into a room with a lock on the door. While physical access cannot be denied using software and hardware, measures can be taken to prevent physical access from leading to system interference.

As we explore the interaction patterns, we will look at “device control” and “device data” with the same level of attention while threat modeling. “Device control” can be classified as any information that is provided to a device by any party with the goal of changing or influencing its behavior towards its state or the state of its environment. “Device data” can be classified as any information that a device emits to any other party about its state and the observed state of its environment.

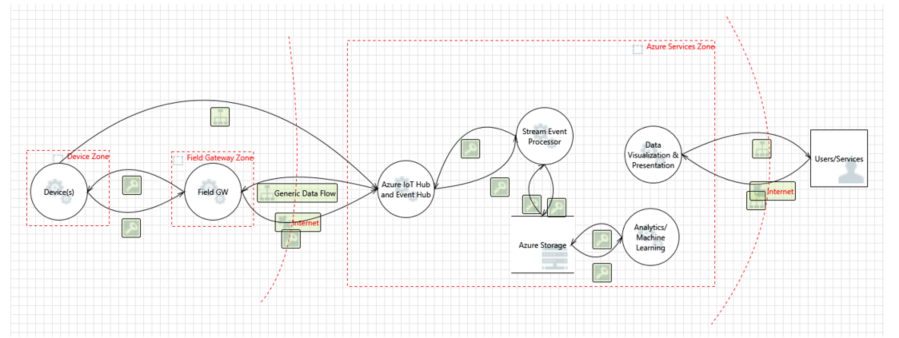
Threat modeling the Azure IoT reference architecture

Microsoft uses the framework outlined above to do threat modelling for Azure IoT. In the section below we therefore use the concrete example of Azure IoT Reference Architecture to demonstrate how to think about threat modelling for IoT and how to address the threats identified. In our case we identified four main areas of focus:

* Devices and Data Sources,
* Data Transport,
* Device and Event Processing, and
* Presentation



The diagram below provides a simplified view of Microsoft’s IoT Architecture using a Data Flow Diagram model that is used by the Microsoft Threat Modeling Tool:



It is important to note that the architecture separates the device and gateway capabilities. This allows the user to leverage gateway devices that are more secure: they are capable of communicating with the cloud gateway using secure protocols, which typically requires greater processing overhead that a native device - such as a thermostat - could provide on its own. In the Azure services zone, we assume that the Cloud Gateway is represented by the Azure IoT Hub service.

Device and data sources/data transport

This section explores the architecture outlined above through the lens of threat modeling and gives an overview of how we are addressing some of the inherent concerns. We will focus on the core elements of a threat model:

* Processes (those under our control and external items)
* Communication (also called data flows)
* Storage (also called data stores)

Processes

In each of the categories outlined in the Azure IoT architecture, we try to mitigate a number of different threats across the different stages data/information exists in: process, communication, and storage. Below we give an overview of the most common ones for the “process” category, followed by an overview of how these could be best mitigated:

**Spoofing (S)**: An attacker may extract cryptographic key material from a device, either at the software or hardware level, and subsequently access the system with a different physical or virtual device under the identity of the device the key material has been taken from. A good illustration is remote controls that can turn any TV and that are popular prankster tools.

**Denial of Service (D)**: A device can be rendered incapable of functioning or communicating by interfering with radio frequencies or cutting wires. For example, a surveillance camera that had its power or network connection intentionally knocked out will not report data, at all.

**Tampering (T)**: An attacker may partially or wholly replace the software running on the device, potentially allowing the replaced software to leverage the genuine identity of the device if the key material or the cryptographic facilities holding key materials were available to the illicit program. For example an attacker may leverage extracted key material to intercept and suppress data from the device on the communication path and replace it with false data that is authenticated with the stolen key material.

**Information Disclosure (I)**: If the device is running manipulated software, such manipulated software could potentially leak data to unauthorized parties. For example, an attacker may leverage extracted key material to inject itself into the communication path between the device and a controller or field gateway or cloud gateway to siphon off information.

**Elevation of Privilege (E)**: A device that does specific function can be forced to do something else. For example, a valve that is programmed to open half way can be tricked to open all the way.

| **Component** | **Threat** | **Mitigation** | **Risk** | **Implementation** |
| --- | --- | --- | --- | --- |
| Device | S | Assigning identity to the device and authenticating the device | Replacing device or part of the device with some other device. How do we know we are talking to the right device? | Authenticating the device, using Transport Layer Security (TLS) or IPSec. Infrastructure should support using pre-shared key (PSK) on those devices that cannot handle full asymmetric cryptography. Leverage Azure AD, OAuth (<http://www.rfc-editor.org/in-notes/internet-drafts/draft-tschofenig-ace-oauth-iot-00.txt>) |
|  | TRID | Apply tamperproof mechanisms to the device for example by making it very hard to impossible to extract keys and other cryptographic material from the device. | The risk is if someone is tampering the device (physical interference). How are we sure, that device has not tampered with. | The most effective mitigation is a trusted platform module (TPM) capability that allows storing keys in special on-chip circuitry from which the keys cannot be read, but can only be used for cryptographic operations that use the key but never disclose the key. Memory encryption of the device. Key management for the device. Signing the code. |
|  | E | Having access control of the device. Authorization scheme. | If the device allows for individual actions to be performed based on commands from an outside source, or even compromised sensors, it will allow the attack to perform operations not otherwise accessible. | Having authorization scheme for the device |
| Field Gateway | S | Authenticating the Field gateway to Cloud Gateway (cert based, PSK, Claim based,..) | If someone can spoof Field Gateway, then it can present itself as any device. | TLS RSA/PSK, IPSe, [RFC 4279](https://tools.ietf.org/html/rfc4279). All the same key storage and attestation concerns of devices in general – best case is use TPM. 6LowPAN extension for IPSec to support Wireless Sensor Networks (WSN). |
|  | TRID | Protect the Field Gateway against tampering (TPM?) | Spoofing attacks that trick the cloud gateway thinking it is talking to field gateway could result in information disclosure and data tampering | Memory encryption, TPM’s, authentication. |
|  | E | Access control mechanism for Field Gateway |  |  |

Here are some examples of threats in this category:

Spoofing: An attacker may extract cryptographic key material from a device, either at the software or hardware level, and subsequently access the system with a different physical or virtual device under the identity of the device the key material has been taken from.

**Denial of Service**: A device can be rendered incapable of functioning or communicating by interfering with radio frequencies or cutting wires. For example, a surveillance camera that had its power or network connection intentionally knocked out will not report data, at all.

**Tampering**: An attacker may partially or wholly replace the software running on the device, potentially allowing the replaced software to leverage the genuine identity of the device if the key material or the cryptographic facilities holding key materials were available to the illicit program.

**Tampering**: A surveillance camera that’s showing a visible-spectrum picture of an empty hallway could be aimed at a photograph of such a hallway. A smoke or fire sensor could be reporting someone holding a lighter under it. In either case, the device may be technically fully trustworthy towards the system, but it will report manipulated information.

**Tampering**: An attacker may leverage extracted key material to intercept and suppress data from the device on the communication path and replace it with false data that is authenticated with the stolen key material.

**Tampering**: An attacker may partially or completely replace the software running on the device, potentially allowing the replaced software to leverage the genuine identity of the device if the key material or the cryptographic facilities holding key materials were available to the illicit program.

**Information Disclosure**: If the device is running manipulated software, such manipulated software could potentially leak data to unauthorized parties.

**Information Disclosure**: An attacker may leverage extracted key material to inject itself into the communication path between the device and a controller or field gateway or cloud gateway to siphon off information.

**Denial of Service**: The device can be turned off or turned into a mode where communication is not possible (which is intentional in many industrial machines).

**Tampering**: The device can be reconfigured to operate in a state unknown to the control system (outside of known calibration parameters) and thus provide data that can be misinterpreted Elevation of Privilege: A device that does specific function can be forced to do something else. For example, a valve that is programmed to open half way can be tricked to open all the way.

**Denial of Service**: The device can be turned into a state where communication is not possible.

**Tampering**: The device can be reconfigured to operate in a state unknown to the control system (outside of known calibration parameters) and thus provide data that can be misinterpreted.

**Spoofing/Tampering/Repudiation**: If not secured (which is rarely the case with consumer remote controls) an attacker can manipulate the state of a device anonymously. A good illustration is remote controls that can turn any TV and that are popular prankster tools.

Communication

Threats around communication path between devices, devices and field gateways and device and cloud gateway. The table below has some guidance around open sockets on the device/VPN:

| **Component** | **Threat** | **Mitigation** | **Risk** | **Implementation** |
| --- | --- | --- | --- | --- |
| Device IoT Hub | TID | (D)TLS (PSK/RSA) to encrypt the traffic | Eavesdropping or interfering the communication between the device and the gateway | Security on the protocol level. With custom protocols, we need to figure out how to protect them. In most cases, the communication takes place from the device to the IoT Hub (device initiates the connection). |
| Device Device | TID | (D)TLS (PSK/RSA) to encrypt the traffic. | Reading data in transit between devices. Tampering with the data. Overloading the device with new connections | Security on the protocol level (HTTP(S)/AMQP/MQTT/CoAP. With custom protocols, we need to figure out how to protect them. The mitigation for the DoS threat is to peer devices through a cloud or field gateway and have them only act as clients towards the network. The peering may result in a direct connection between the peers after having been brokered by the gateway |
| External Entity Device | TID | Strong pairing of the external entity to the device | Eavesdropping the connection to the device. Interfering the communication with the device | Securely pairing the external entity to the device NFC/Bluetooth LE. Controlling the operational panel of the device (Physical) |
| Field Gateway Cloud Gateway | TID | TLS (PSK/RSA) to encrypt the traffic. | Eavesdropping or interfering the communication between the device and the gateway | Security on the protocol level (HTTP(S)/AMQP/MQTT/CoAP). With custom protocols, we need to figure out how to protect them. |
| Device Cloud Gateway | TID | TLS (PSK/RSA) to encrypt the traffic. | Eavesdropping or interfering the communication between the device and the gateway | Security on the protocol level (HTTP(S)/AMQP/MQTT/CoAP). With custom protocols, we need to figure out how to protect them. |

Here are some examples of threats in this category:

**Denial of Service**: Constrained devices are generally under DoS threat when they actively listen for inbound connections or unsolicited datagrams on a network, because an attacker can open many connections in parallel and not service them or service them very slowly, or the device can be flooded with unsolicited traffic. In both cases, the device can effectively be rendered inoperable on the network.

**Spoofing, Information Disclosure**: Constrained devices and special-purpose devices often have one-for-all security facilities like password or PIN protection, or they wholly rely on trusting the network, meaning they will grant access to information when a device is on the same network, and that network is often only protected by a shared key. That means that when the shared secret to device or network is disclosed, it is possible to control the device or observe data emitted from the device.

**Spoofing**: an attacker may intercept or partially override the broadcast and spoof the originator (man in the middle)

**Tampering**: an attacker may intercept or partially override the broadcast and send false information Information Disclosure: an attacker may eavesdrop on a broadcast and obtain information without authorization Denial of Service: an attacker may jam the broadcast signal and deny information distribution

Storage

Every device and field gateway has some form of storage (temporary for queuing the data, os image storage).

| **Component** | **Threat** | **Mitigation** | **Risk** | **Implementation** |
| --- | --- | --- | --- | --- |
| Device storage | TRID | Storage encryption, signing the logs | Reading data from the storage (PII data), tampering with telemetry data. Tampering with queued or cached command control data. Tampering with configuration or firmware update packages while cached o queued locally can lead to OS and/or system components being compromised | Encryption, message authentication code (MAC) or digital signature. Where possible, strong access control through resource access control lists (ACLs) or permissions. |
| Device OS image | TRID |  | Tampering with OS /replacing the OS components, | Read-only OS partition, signed OS image, Encryption |
| Field Gateway storage (queuing the data) | TRID | Storage encryption, signing the logs | Reading data from the storage (PII data), tampering with telemetry data, tampering with queued or cached command control data. Tampering with configuration or firmware update packages (destined for devices or field gateway) while cached o queued locally can lead to OS and/or system components being compromised | BitLocker |
| Field Gateway OS image | TRID |  | Tampering with OS /replacing the OS components | Read-only OS partition, signed OS image, Encryption |

Device and event processing/cloud gateway zone

A cloud gateway is system that enables remote communication from and to devices or field gateways from several different sites across public network space, typically towards a cloud-based control and data analysis system, a federation of such systems. In some cases, a cloud gateway may immediately facilitate access to special-purpose devices from terminals such as tablets or phones. In the context discussed here, “cloud” is meant to refer to a dedicated data processing system that is not bound to the same site as the attached devices or field gateways, and where operational measures prevent targeted physical access but is not necessarily to a “public cloud” infrastructure. A cloud gateway may potentially be mapped into a network virtualization overlay to insulate the cloud gateway and all of its attached devices or field gateways from any other network traffic. The cloud gateway itself is neither a device control system nor a processing or storage facility for device data; those facilities interface with the cloud gateway. The cloud gateway zone includes the cloud gateway itself along with all field gateways and devices directly or indirectly attached to it.

Cloud gateway is mostly custom built piece of software running as a service with exposed endpoints to which field gateway and devices connect. As such it must be designed with security in mind. Please follow [SDL](http://www.microsoft.com/sdl) process for designing and building this service.

Services zone

A control system (or controller) is a software solution that interfaces with a device, or a field gateway, or cloud gateway for the purpose of controlling one or multiple devices and/or to collect and/or store and/or analyze device data for presentation, or subsequent control purposes. Control systems are the only entities in the scope of this discussion that may immediately facilitate interaction with people. The exception are intermediate physical control surfaces on devices, like a switch that allows a person to turn the device off or change other properties, and for which there is no functional equivalent that can be accessed digitally.

Intermediate physical control surfaces are those where any sort of governing logic constrains the function of the physical control surface such that an equivalent function can be initiated remotely or input conflicts with remote input can be avoided – such intermediated control surfaces are conceptually attached to a local control system that leverages the same underlying functionality as any other remote control system that the device may be attached to in parallel. Top threats to the cloud computing can be read at [Cloud Security Alliance (CSA)](https://cloudsecurityalliance.org/research/top-threats/) page.