CHAPTER 5

Azure Stream Analytics

At this point in the process, the data from the devices is sitting in Azure IoT Hub waiting for the next step of the journey. In this case, "sitting" means that the data has just arrived in Azure IoT Hub and is waiting to be picked up by another service for processing. Chapter 3 walked through the process of creating and configuring an Azure IoT Hub to receive the messages sent from the devices. And, depending on how the IoT Hub was configured, the data currently sitting in IoT Hub could have a very short lifespan.

The configuration of the IoT Hub in Chapter 3 chose the free tier, which means that the retention time for the messages is only a day. A single day. No more. Picking a standard S1 or standard S2 tier gives the flexibility to up the retention time to a maximum of seven days. A whole week. Honestly, whether it's a single day or seven days, that is not a bad thing. The data shouldn't be sitting around at this point anyway. The fact that IoT Hub keeps the data around for up to seven days means that it is guaranteeing reliable processing while also helping to absorb peak loads.

However, this point goes back to the *hot path*/*cold path* discussion. Really, you should have already determined what you intend to do with the data. Do you need real-time or near real-time analysis of the data, or do you plan on storing the data and keeping the data around for future analysis?

The thing is, regardless of the answer, IoT Hub is only a temporary holding place for incoming data. The next step is to pick up that data as soon as it comes in and do something with it, whether it's immediate analysis or storing it long term. In addition, regardless of the path (hot or cold), the data needs to be picked up and processed, so what is needed is a real-time processing engine that can pick up the data from IoT Hub, drop it where you want it, and in between maybe do some data transformations. And it would be nice if this engine was also highly scalable.

Luckily there is such an engine, and it is called Azure Stream Analytics. This chapter will take a detailed look at this service, what it can be used for and the scenarios in which it applies, and how to create and configure a Stream Analytics service.

What Is Azure Stream Analytics?

Simply put, Azure Stream Analytics (ASA) is a real-time event processing engine that provides analytical capabilities on streaming data from devices, applications, sensors, and more. It is a fully managed, highly scalable Azure service focused on making access to deep data insights powerful, inexpensive, and easy to use.

In the big picture, Azure Stream Analytics picks up ingested data from one of several sources, does some level of data transformation, and outputs the result to a different set of destinations. The flow that this book is taking is shown in Figure 5-1, which, as you saw in the previous chapters, is sending data from the devices directly to Azure IoT Hub. As discussed previously, Azure provides the option to use Cloud or Field gateways into IoT Hub for legacy (custom protocols) or low-powered devices, enabling them to take advantage of the same workflow and analysis benefits.

S. Klein, *IoT Solutions in Microsoft's Azure IoT Suite*, DOI 10.1007/978-1-4842-2143-3_5

Later chapters will fill in and complete the picture, but for now, Figure 5-1 shows where Stream Analytics fits into the current process.

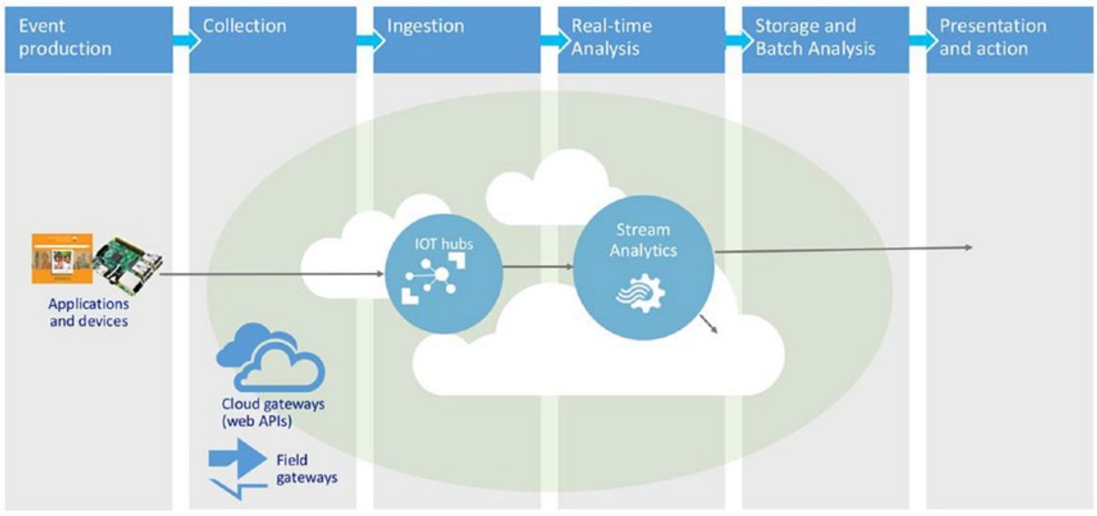


Figure 5-1. *Data flow*

Given what you know up to this point, think about the amount of data that can flow into Azure IoT Hub and get picked up by Azure Stream Analytics to be processed and acted upon in real time. Imagine the different scenarios where this can be useful: anomaly or fraud detection, real-time stock trading, or even high-stakes betting and gambling. What all of these scenarios have in common is the need to be scalable, reliable, and cost effective in every aspect of the process.

Azure Stream Analytics isn’t just limited to the scenarios just mentioned. Stream Analytics fits anywhere businesses need the ability to gain real-time insights into incoming data. At the beginning of this chapter, the topic of hot path and cold path data processing was mentioned. Hot path equates to real-time processing whereas cold path equates more to micro-batch processing. Real-time processing denotes the processing of an infinite stream of input data, with the time between data ingestion and ready results being small, typically measured in seconds.

Micro-batch processing is the method by which a group of transactions are collected over a period of time and processed as a batch. In Azure Stream Analytics, micro-batch processing is performed by a Stream Analytics job in which an input and output are defined and a SQL query is used to process the incoming data. An example of this will be shown in the next chapter as you continue the example from Chapter 4.

With the data in Azure IoT Hub, the next step in the journey is to create and configure a Stream Analytics job. However, before I get to that, let’s take a step back and drill down into the key capabilities and benefits of Stream Analytics, which will help put into perspective the scenarios where Stream Analytics fits as well as the best way to configure the Stream Analytics job.

Key Benefits and Capabilities

The capabilities and benefits of Azure Stream Analytics share many of the character traits as Azure IoT Hub and other services which have been discussed and which will be discussed in chapters to come.

- **Scale**: With hundreds and thousands of devices sending vast amounts of data into the cloud, Azure Stream Analytics is designed and built to handle the millions of events per second coming in, up to 1GB/second. What makes this possible is the partitioning capabilities of Event Hubs.
- **Reliability**: Data loss prevention and business continuity, two key facets of Stream Analytics, are provided via built-in recovery features and the ability to maintain state. As such, Stream Analytics is able to archive events and reapply processing to ensure repeatable results.
- **Connectivity**: Stream Analytics allows connectivity to many different sources and destinations, including Event Hubs and IoT Hubs for stream ingestion, as well as Azure Storage (Tables and Blobs), Azure SQL Database, Azure Data Lake, and even Power BI for output and results. A full list will be discussed in Chapter 6.
- **Ease of Use**: Not only is creating inputs and outputs extremely simple, but Stream Analytics also makes data transformations simple through the support of a variant of the SQL language called the Stream Analytics Query Language. Available right in the Azure portal, this language supports IntelliSense and auto-complete directly in the query editor.
- **Cost Effectiveness**: As with other Azure cloud services, Stream Analytics is designed to provide a high-value, real-time analytics solution for a low cost. Built around a pay-as-you-go model, the cost is based on the number of units and the amount of data processed.

The bullets above summarize the value of Stream Analytics, but the proof of its value is clearly seen once it is put into action. As you'll see in the next chapter, many of the capabilities and characteristics of Stream Analytics will stand out as they are put to use.

One such case is the query language. The SQL-like query language includes many of the familiar functions, filters, and other operators found in SQL Server. Another example is the ease of creating inputs and outputs simply with a few clicks in the portal. But let's be clear that the portal isn't the only place jobs can be created. Azure Stream Analytics has a .NET API that enables entire job management, such as creating, configuration, and running complete Stream Analytics jobs. There are also nearly 20 Azure PowerShell cmdlets to automate common Azure Stream Analytics tasks, such as starting and stopping jobs, creating job input, and more.

That is enough background and information to get started. It is time to create and configure your Stream Analytics job and put it to use. The following section will walk through the creation of a Stream Analytic job and discuss the different configurations and settings available.

Creating an Azure Stream Analytics Job

To begin, open your favorite browser and go to `Azure.Portal.com` and sign in. Once in the portal, click New, select Internet of Things, and then select Stream Analytics Job, as shown in Figure 5-2.

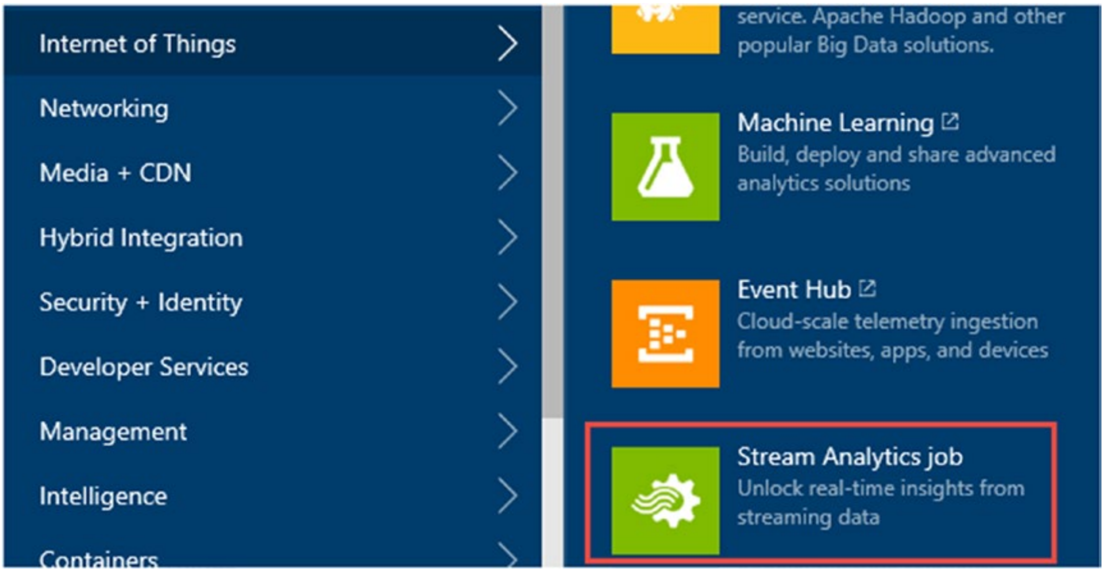

Internet of Things
Networking
Media + CDN
Hybrid Integration
Security + Identity
Developer Services
Management
Intelligence
Containers
service. Apache Hadoop and other popular Big Data solutions.
Machine Learning
Build, deploy and share advanced analytics solutions
Event Hub
Cloud-scale telemetry ingestion from websites, apps, and devices
Stream Analytics job
Unlock real-time insights from streaming data

Figure 5-2. *Creating a new Stream Analytics job*

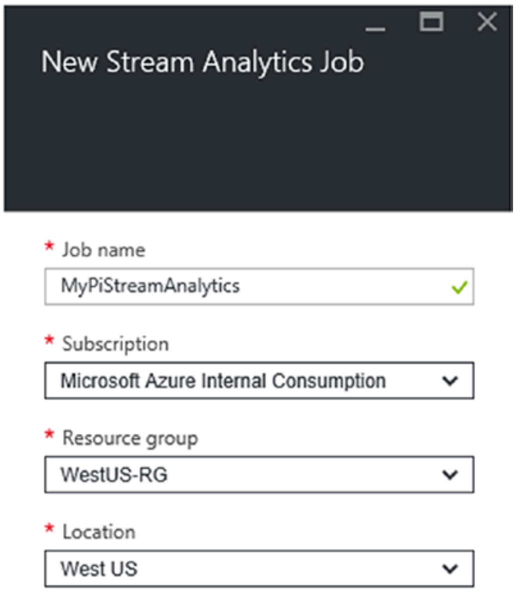
If you are familiar with the previous version of the portal, it was just called Stream Analytics, but in the new portal (**portal.azure.com**) the name is Stream Analytics job.

The New Stream Analytics Job blade will appear, similar to the one in Figure 5-3. In the blade, enter or select the following:

- **Job Name**: The name of the Stream Analytics job. In my demo, I named it MyPiStreamAnalytics.
- **Resource Group**: Either select an existing Resource Group or create a new one. Be sure to select the same resource group that your IoT Hub is in.
- **Location**: Select the appropriate region in which to host the IoT Hub.

One thing to notice that is not on this blade is the pricing and scale option to initially set the performance and cost. Once the Stream Analytics job is created, the performance level will be configurable.

Click the "Pin to dashboard" checkbox and then click Create. The creation of the Steam Analytics job will take only a moment, probably less than a minute.


New Stream Analytics Job

* Job name
MyPiStreamAnalytics

* Subscription
Microsoft Azure Internal Consumption

* Resource group
WestUS-RG

* Location
West US

Figure 5-3. *Configuration of the new Stream Analytics job*

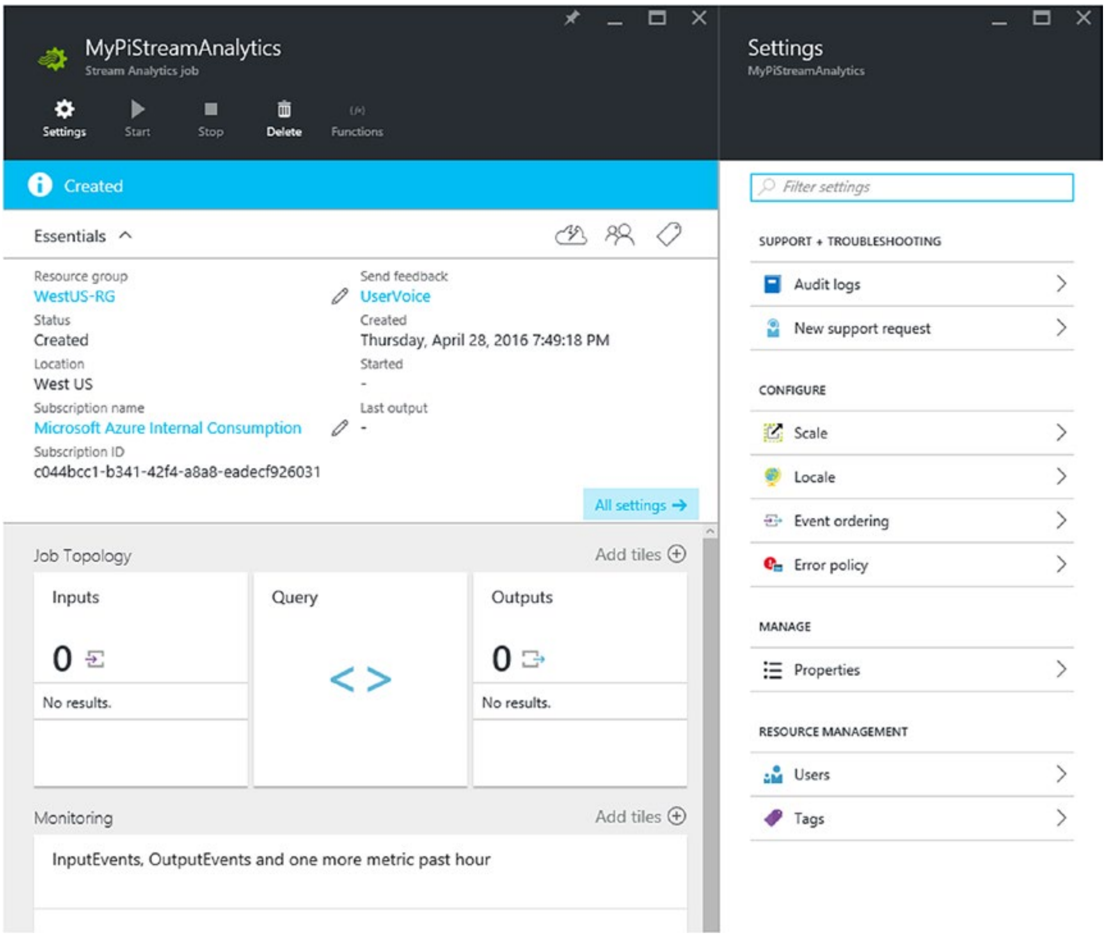
Once it is created, you will see a new icon on your dashboard. Click the new Stream Analytics job icon on your dashboard to open up both the Overview blade and the Settings blade for the newly created Stream Analytics job, similar to Figure 5-4.

The left Overview blade provides a summary of the Stream Analytics job as well as the all-important job topology information such as the number of inputs and outputs, queries, and vital job monitoring information.

Two important items to point out on the Overview blade are the Start and Stop buttons for the job. When the job is first created, those buttons are disabled simply because no job is configured, so no inputs or outputs are defined and no query is created. Therefore, the job cannot be run, so the buttons are disabled. Once an input, output, and a query have been defined, those buttons will be enabled.

There is also a Functions button that is disabled upon job creation. Functions allow you to call a REST endpoint, which allows each message received to be processed by a web service GET/POST. It allows the "self-discovery" of the endpoint parameters. Functions are primarily intended for use with AzureML (you will see this in the Azure Machine Learning chapter) where machine learning models can be enabled and invoked through a REST API.

Before getting to the Settings blade and the different configurations, a few minutes needs to be spent on monitoring and diagnostics. In Figure 5-4, there is no settings option for diagnostics. To turn on monitoring and diagnostics, click anywhere in the Monitoring graph, which will open the Diagnostics blade. In this blade is where you simply turn on or off diagnostics monitoring. When turning it on, it will ask you to pick the storage in which to store the telemetry and diagnostics data. Simply pick a storage account, ensuring that the storage selected is in the same data center as the SA job, and click Save.



Resource group: WestUS-RG
Status: Created
Location: West US
Subscription name: Microsoft Azure Internal Consumption
Subscription ID: c044bcc1-b341-42f4-a8a8-eadecf926031
Send feedback: UserVoice
Created: Thursday, April 28, 2016 7:49:18 PM
Started: -
Last output: -

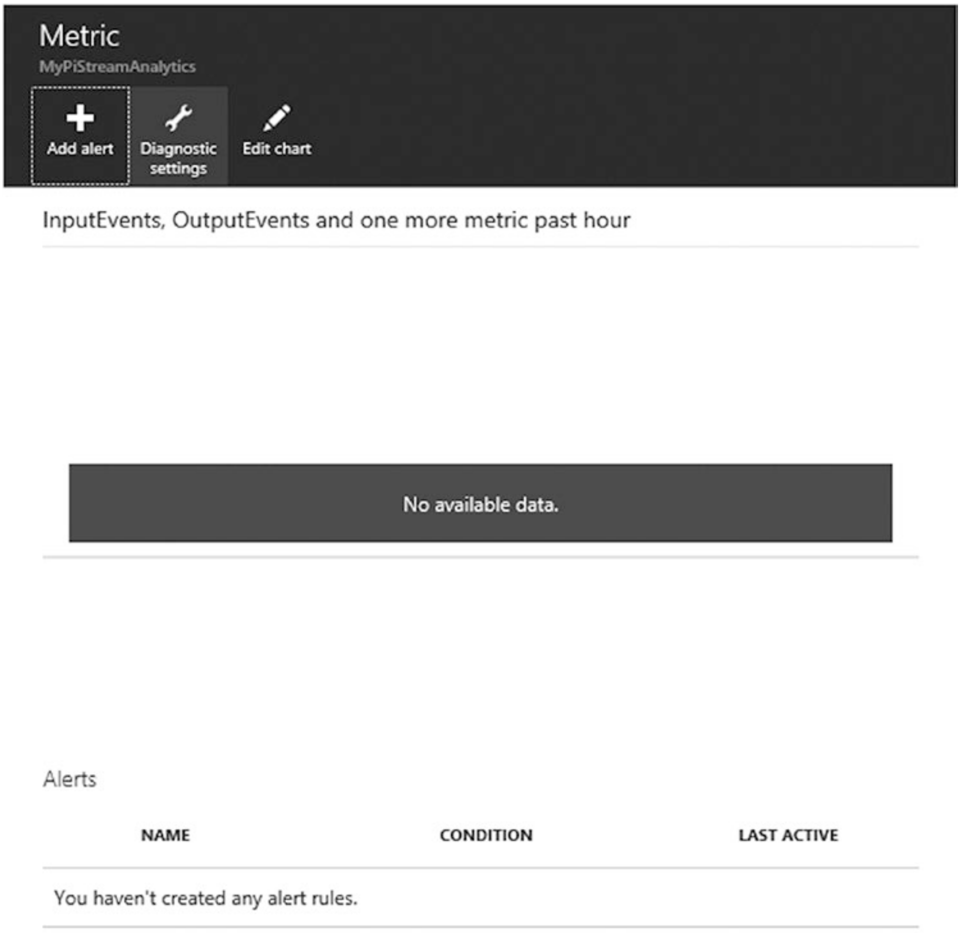
Inputs: 0 — No results.
Outputs: 0 — No results.

InputEvents, OutputEvents and one more metric past hour

Figure 5-4. *Azure Stream Analytics job overview pane*

At this point, monitoring is turned on but no diagnostic telemetry data is being gathered because no metrics have been defined. To define metrics and alerts and begin gathering diagnostic data, click again in the Monitoring graph, which will open the Metric blade, shown in Figure 5-5.

At this point, there is no data because no inputs or outputs have been defined. Also, no alerts have been added, thus the "No available data" message in the Metric graph.



InputEvents, OutputEvents and one more metric past hour

No available data.

Alerts

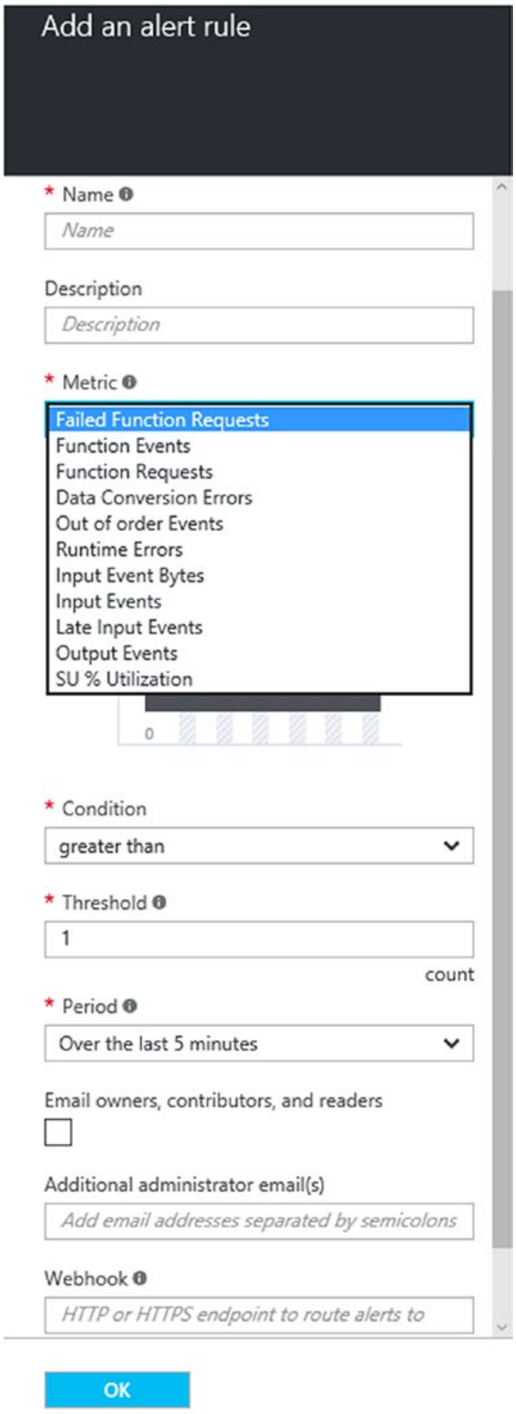
NAME	CONDITION	LAST ACTIVE
You haven't created any alert rules.		

Figure 5-5. *Azure Stream Analytics Metric pane*

However, it is worth at least mentioning here how to create and configure alerts so that when the inputs and outputs are created in the next chapter, it will be easy to come back here to configure the monitoring.

Besides providing a name, an alert is created by specifying which metric to monitor and the conditions for that metric. As shown in Figure 5-6, there are 11 key performance metrics that can be used to monitor and troubleshoot job and query performance along with their corresponding condition, threshold, and the amount of time over which to monitor the metric data.

Additionally, alerts can be raised via email to Azure subscription owners and contributors along with additional persons specified.



Add an alert rule

* Name

Name

Description

Description

* Metric

Failed Function Requests
Function Events
Function Requests
Data Conversion Errors
Out of order Events
Runtime Errors
Input Event Bytes
Input Events
Late Input Events
Output Events
SU % Utilization

* Condition

greater than

* Threshold

1

count

* Period

Over the last 5 minutes

Email owners, contributors, and readers

☐

Additional administrator email(s)

Add email addresses separated by semicolons

Webhook

HTTP or HTTPS endpoint to route alerts to

OK

Figure 5-6. *Defining Azure Stream Analytics alerts*

The current list of available metrics helps gain insight into what is happening within the Stream Analytics job through error and performance tracking. The following lists the available metrics and their definition:

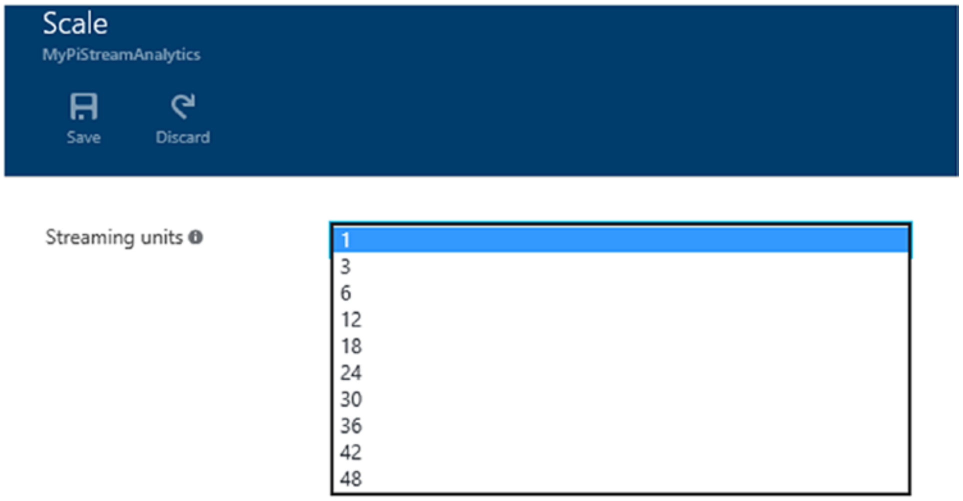
- **Failed Function Requests**: The number of failed HTTP response codes (e.g. HTTP 400/500) from a function request.
- **Function Events**: The number of events that are triggered from a function request.
- **Function Requests**: The number of requests to each and all registered functions.
- **Data Conversion Errors**: The number of data conversion errors incurred by a SA job.
- **Out of Order Events**: The number of events that were received out of order, which were either dropped or given an adjusted timestamp.
- **Runtime Errors**: The number of errors incurred during a SA job execution.
- **Input Event Bytes**: Amount of throughput data received by the job in terms of bytes.
- **Input Events**: Amount of data received by the job.
- **Late Input Events**: The number of events arriving late from the source, which have either been dropped or have an adjusted timestamp.
- **Output Events**: Amount of data sent by the SA job to the output target.
- **SU% Utilization**: The utilization of the streaming units assigned to a job from the Scale tab.

Each metric has a corresponding condition and threshold. For example, you might select the Input Events metric with a condition of "greater than or equal to" and a threshold of 10,000 to closely monitor how much data your input is receiving. You might consider combining this metric with the SU% Utilization metric to ensure that you are getting optimal performance and not peaking the limits. These metrics together can tell you if you need to increase your streaming units to handle the number of events without pegging your SU% utilization.

You can have only six metrics displayed on the chart, so be wise with the metrics you choose to monitor. If your Stream Analytics job is pulling in data from multiple streams (from multiple IoT Hubs) for example, you might consider adding the Out-of-Order Events metric to determine if there is any latency of arriving messages. More on Out-of-Order Events shortly.

Scale

One of the most difficult aspects of configuring an Azure service is figuring out how many resources are needed for optimum performance. In Azure Stream Analytics, performance is determined by selecting the appropriate streaming units. In the Settings blade, click the Scale option, which opens the Scale blade and the option to select the number of streaming units, shown in Figure 5-7.



Streaming units

1
3
6
12
18
24
30
36
42
48

Figure 5-7. *Configuring Azure Stream Analytics streaming units*

Streaming units characterize the resources and performance of executing a Stream Analytics job. They embody a blended combination of memory, CPU, and I/O. Ultimately, each SU equates to roughly 1MB/second of throughput.

The trick is determining how many SUs are needed for a particular job. It's not rocket science, but it does take some knowhow along with some testing and monitoring. The correct SU selection will depend on the query defined for the job as well as the partition configuration for the inputs. Thus, when writing the query, how it is written will have an impact on the overall performance.

Another way, and possibly an easier way, to determine SUs is to test the lowest common denominator as a scale unit and understand the data volume peaks. IoT Hub will buffer the messages so you can have a lower value of SU and a higher retention period, which would equate to a process that is just slightly less than real-time but with a lower cost.

In essence, you are trying to find the balance of price vs. performance. Can you pay less for "not quite" real time and be fine with a 5 second processing window, for example? Or, do you need the SUs to provide the absolute real-time processing with high performance? The point is that getting the performance out of Azure Stream Analytics (i.e. increasing stream data processing) depends on the partition count for the inputs and the query defined for the job. You will probably spend some time tweaking both to find the right configuration. However, as stated earlier, test with the lowest common denominator and tweak from there based on your data volume peaks.

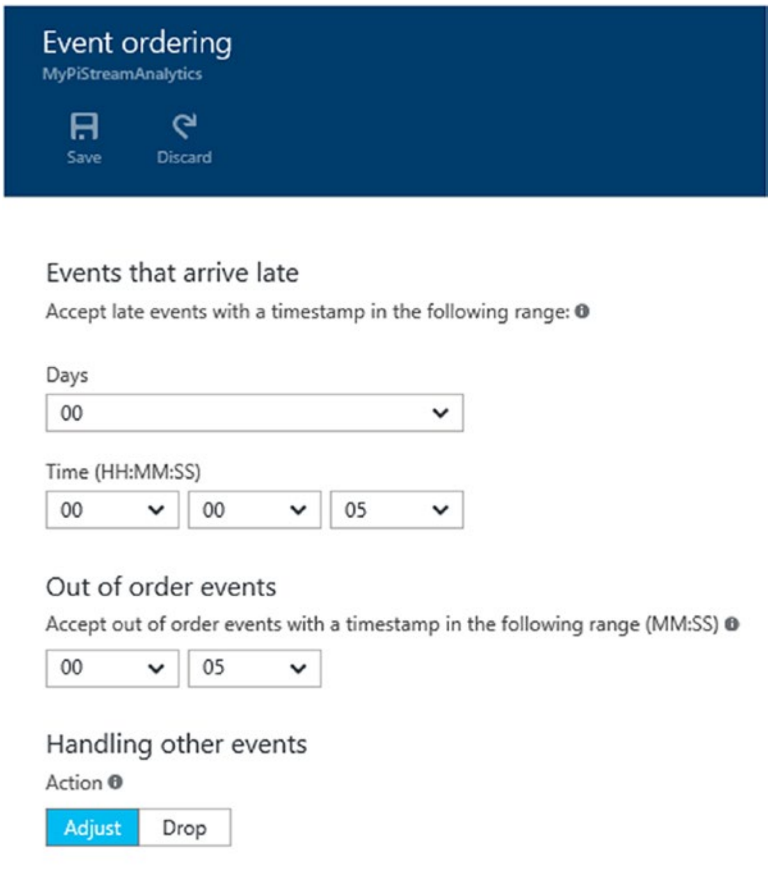
Event Ordering

Earlier it was stated that Stream Analytics was built to handle the millions of events per second coming in, up to 1GB/second. To achieve such numbers would mean that there are many devices sending a large amount data for Stream Analytics to process. This could also mean that quite possibly Stream Analytics is processing streams of data from multiple IoT Hubs or Event Hubs. SU is also based on volume of data processing and as such Stream Analytics scales nicely because device payloads are typically small, which is why it can process millions of messages on tumbling windows.

The complexity of these types of solutions might mean that some events don't make it from the client to the hub as fast as they should due to varying latency reasons, thus arriving at the hub out of order. Events might even arrive out of order once they've made the trip from the input source to Stream Analytics, especially in a multi-stream scenario. Additionally, to handle the multi-stream scenario, the complexity of the query increases to handle the arrival of late, independent events.

It is worth pointing out that AMQP defines a client side "buffer," which should preserve the order based on connectivity loss. This will even out the latency and make this more batch-enabled.

As such, consideration of what to do with these events must take place, which is the purpose of the Event Ordering blade, shown in Figure 5-8.



Events that arrive late

Accept late events with a timestamp in the following range:

Days: 00

Time (HH:MM:SS): 00 00 05

Out of order events

Accept out of order events with a timestamp in the following range (MM:SS): 00 05

Handling other events

Action: [✓ Adjust] Drop

Figure 5-8. Configuring the event ordering in Azure Stream Analytics

The first section in this blade allows you to specify which delayed or late events to accept within a given timeframe. These events can still be picked up by Stream Analytics as long as a timespan is specified in days or hours/minutes/seconds. A delay in the case of late events is defined as the difference between the event's timestamp and the system clock.

The second section deals with events that are out of order during their trip from the input source to Stream Analytics. A likely scenario would be when you are combining multiple streams. Here, the choice is to accept the events as is, or to pause for a set period of time to reorder them.

The last section deals with events that arrive outside of the times specified in the previous sections. Here the choices are to simply delete the events or to adjust the events by changing their timestamp.

Event ordering is dependent on the record timestamp, and one of two timestamps are applied and used. By default, events are timestamped based on their arrival time to the input source. For Event Hub and IoT Hub, the timestamp is the arrival time when the event was received by the hub.

However, some applications necessitate the exact timestamp of when the event occurred. For these cases, the TIMESTAMP BY clause can be added to the SQL query, allowing for the use of custom timestamp values. For these cases, the timestamp value can be any field from the event data payload.

When TIMESTAMP BY is not included in the query, the timestamp from IoT Hub or Event Hub is used.

Audit Log

The Audit Log blade provides insight into what is happening with your Steam Analytics job. This information can be valuable when debugging jobs with facts regarding job status, job failures, and the status of currently executing jobs.

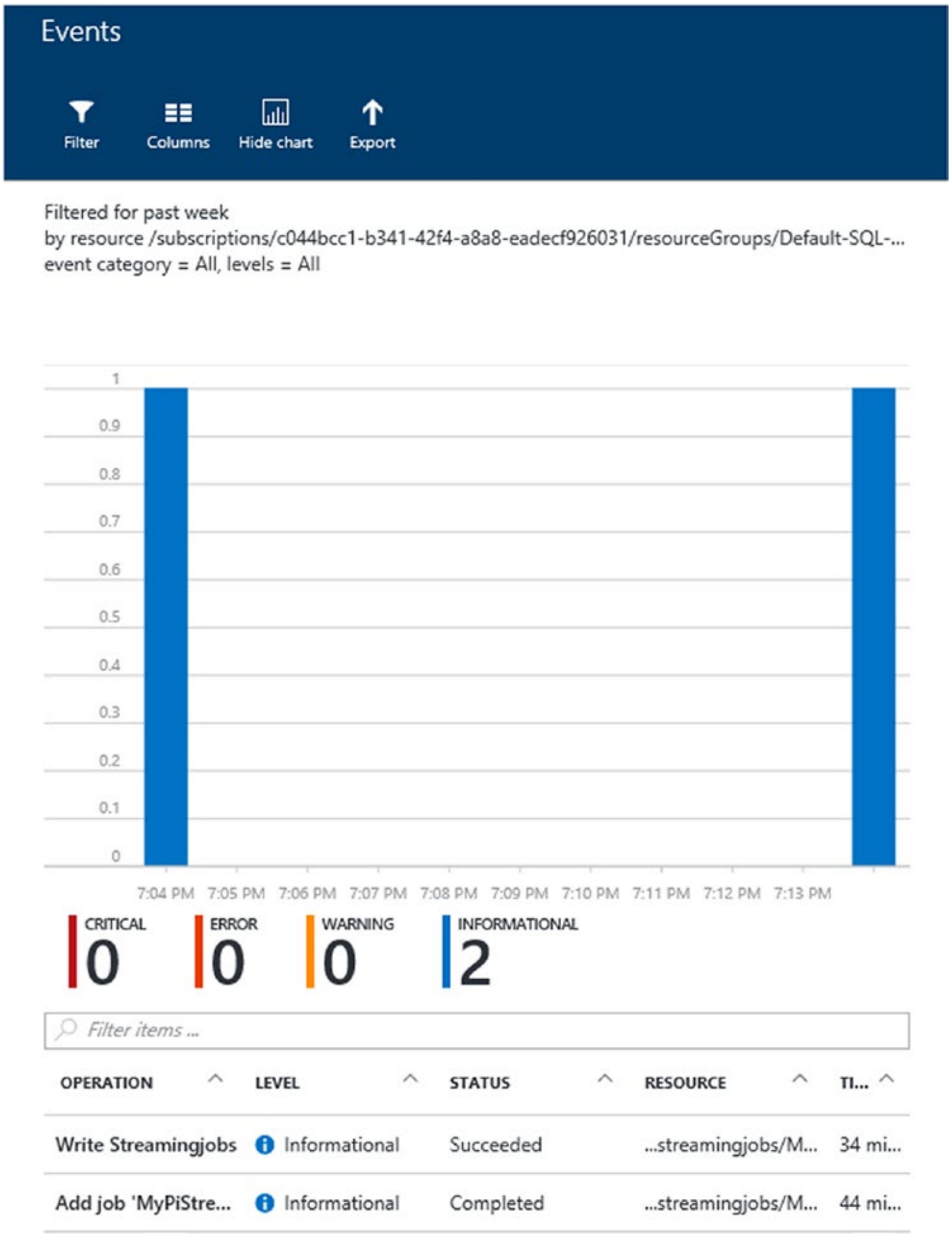
Audit information isn't just available to Stream Analytics. All Azure services provide detailed logging information for debugging and management. The Audit blade shows events from the past seven days, as shown in Figure 5-9. By default, a filter is applied, which shows events for the subscription, resource group, resource type (in this case, streaming jobs), given resource (in this case, Azure Stream Analytics jobs), and all events.

To change the filter, simply click the Filter button on the toolbar. While you can't change the severity, you can change the level of logging and the timespan in which events are tracked. By default, four types of events are tracked:

- Critical
- Error
- Warning
- Informational

As stated above, events are tracked for seven days by default. You can change this by selecting the Time span option in the Filter blade and selecting one of the following:

- Past 1 hour
- Past 24 hours
- Past week
- Custom



Filtered for past week
by resource /subscriptions/c044bcc1-b341-42f4-a8a8-eadecf926031/resourceGroups/Default-SQL-...
event category = All, levels = All

CRITICAL 0 · ERROR 0 · WARNING 0 · INFORMATIONAL 2

OPERATION	LEVEL	STATUS	RESOURCE	TI...
Write Streamingjobs	Informational	Succeeded	...streamingjobs/M...	34 mi...
Add job 'MyPiStre...	Informational	Completed	...streamingjobs/M...	44 mi...

Figure 5-9. *Azure Stream Analytics Events blade*

The Events blade also lists log entries toward the bottom of the blade. You can see details of the log entry by clicking the specific event, which will open the Detail blade for that event. The Detail blade simply provides information regarding the specific event, such as event status, event level, the timestamp of the event, and more.

Additional Settings

The other configuration settings on the Settings blade allow you to initiate a support request, change the internationalization preference, view the Stream Analytics properties, define additional users, and configure how events are handled that fail to be written to the output.

Of all of the settings just mentioned, the Locale blade is the most important because it tells Stream Analytics how to parse and sort the data based on the language and locale settings.

The Error Policy is another one that shouldn't be overlooked, but isn't too critical. The Error Policy simply provides a Drop or Retry setting for failed events. The Drop option drops the event that caused the error completely while Retry retries the event until is succeeds.

Lastly, the Support Request blade will come in handy will working with critical errors found in the audit logs. Simply taking the information found in the audit logs for critical and error information and passing it along in the support request will provide a fast turnaround time for troubleshooting Stream Analytic issues within your Azure subscription.

Summary

This chapter provided an overview of Azure Stream Analytics, beginning with a discussion about streaming data and its lifespan, followed by a look at Azure Stream Analytics, what it is, its core capabilities and key benefits, and why it is considered a valid cloud streaming solution for real-time analysis of streaming data.

The last part of the chapter created a Stream Analytics job and then looked at the crucial configuration settings and some considerations when first setting up your Stream Analytics job. The following chapter will continue the example from Chapter 3 by taking the next configuration steps in the Stream Analytics process by authoring a job by specifying an input, output, and query settings to move the data currently sitting in Azure IoT Hub along.