MICROSOFT EXHIBIT 1085
Microsoft v. Qomplx
IPR2026-00182

EXHIBIT A
U.S. Patent No. 12,143,424

As used herein and with respect to the ’424 Patent, the term “Accused ’424 Products” means:

(a) Microsoft products that incorporate, rely upon, interact with, or otherwise utilize Microsoft Fabric (“Fabric”), including Microsoft Fabric itself;

(b) any other products that utilize the libraries, applications, scripts, packages, or other modules that implement the functionality described below in a manner not materially different with respect to the claims charted below;

(c) any other products that infringe the asserted claims for analogous reasons to those described below; and,

(d) Microsoft products that practice one or more claims of the ’424 Patent.

This claim chart for the ’424 Patent covers all Accused ’424 Products. The theory of infringement described below in connection with the Asserted Claims is analogous to the theory of infringement for all the Accused ’424 Products.

I. Claim 1

A distributed computing cluster comprising:	The Accused ’424 Products are distributed computing clusters. For example, Fabric is a “Software as a Service” system that offers “a data analytics platform,” and runs on Microsoft Azure, a distributed computer cluster:[1] > Microsoft Fabric is a data analytics platform that handles your entire data workflow, from collecting and processing data to building reports and dashboards. It combines data ingestion, transformation, real-time event routing, and visualization through workloads like Data Engineering, Data Factory, Data Science, Real-Time Intelligence, Data Warehouse, and Databases. > As a Software as a Service (SaaS) platform, Fabric centralizes data storage with OneLake. It embeds AI capabilities throughout, so you don't need to integrate services manually, allowing you to transform raw data into actionable insights efficiently. Microsoft Azure includes “400 highly secure datacenters in over 70 regions.”[2] A person of skill in the art would understand that cloud services such as Fabric utilize a plurality of computer systems.

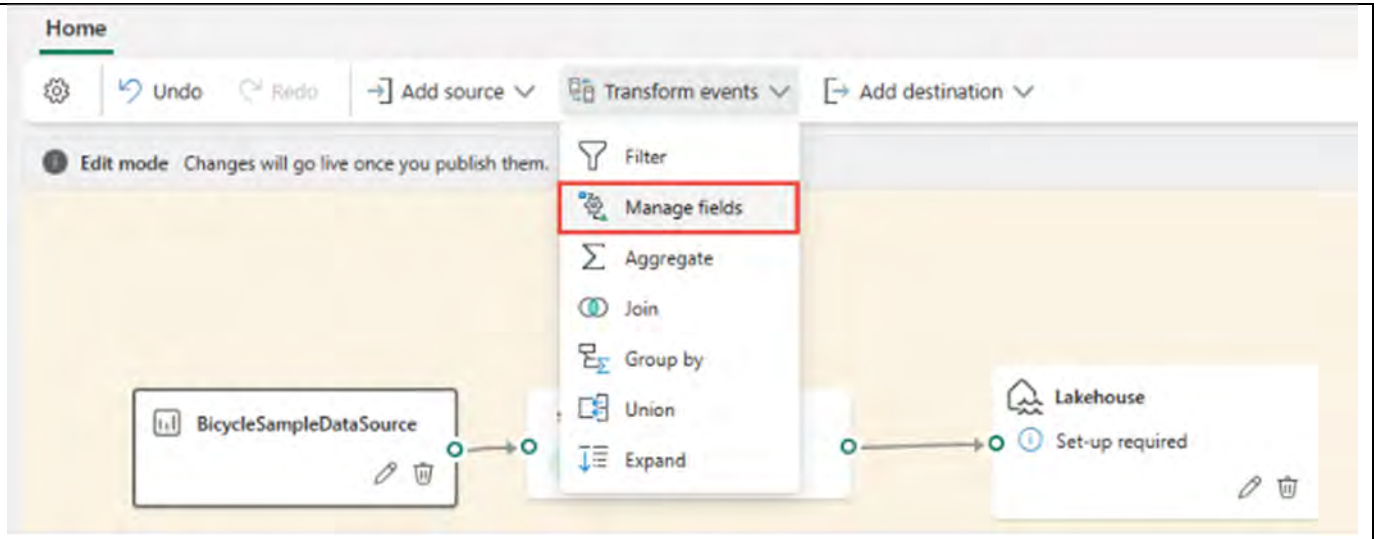
[1] Microsoft, *What is Microsoft Fabric?*, available at `https://learn.microsoft.com/en-us/fabric/fundamentals/microsoft-fabric-overview` [hereinafter *What is Microsoft Fabric?*].

[2] Microsoft, *What is Azure?*, available at `https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/` [hereinafter *What is Azure?*].

	Fabric includes eventstreams.[3]
a first plurality of computer systems, wherein each respective computer system of the first plurality of computer systems comprises a memory that stores a respective first data, wherein the respective first data represents a respective portion of a distributed computational graph and wherein the distributed computational graph describes a flow of output data of a first transformation pipeline to an input of a second transformation pipeline,	The Accused ’424 Products comprise a first plurality of computer systems, wherein each respective computer system of the first plurality of computer systems comprises a memory that stores a respective first data, wherein the respective first data represents a respective portion of a distributed computational graph and wherein the distributed computational graph describes a flow of output data of a first transformation pipeline to an input of a second transformation pipeline. For example, eventstreams allows a user to define portions of a distributed computational graph that describes transformation pipelines and the flow of data streams between the transformation pipelines, as shown in the following exemplary graphical representation:[4]

[3] Microsoft, *Overview of Microsoft Fabric eventstreams*, available at `https://learn.microsoft.com/en-us/fabric/real-time-intelligence/event-streams/overview?tabs=enhancedcapabilities` [hereinafter *Eventstreams Overview*].

[4] Microsoft, *Process event data by using the event processing editor*, available at `https://learn.microsoft.com/en-us/fabric/real-time-intelligence/event-streams/process-events-using-event-processor-editor?pivots=enhanced-capabilities` [hereinafter *Event Processing Editor*].



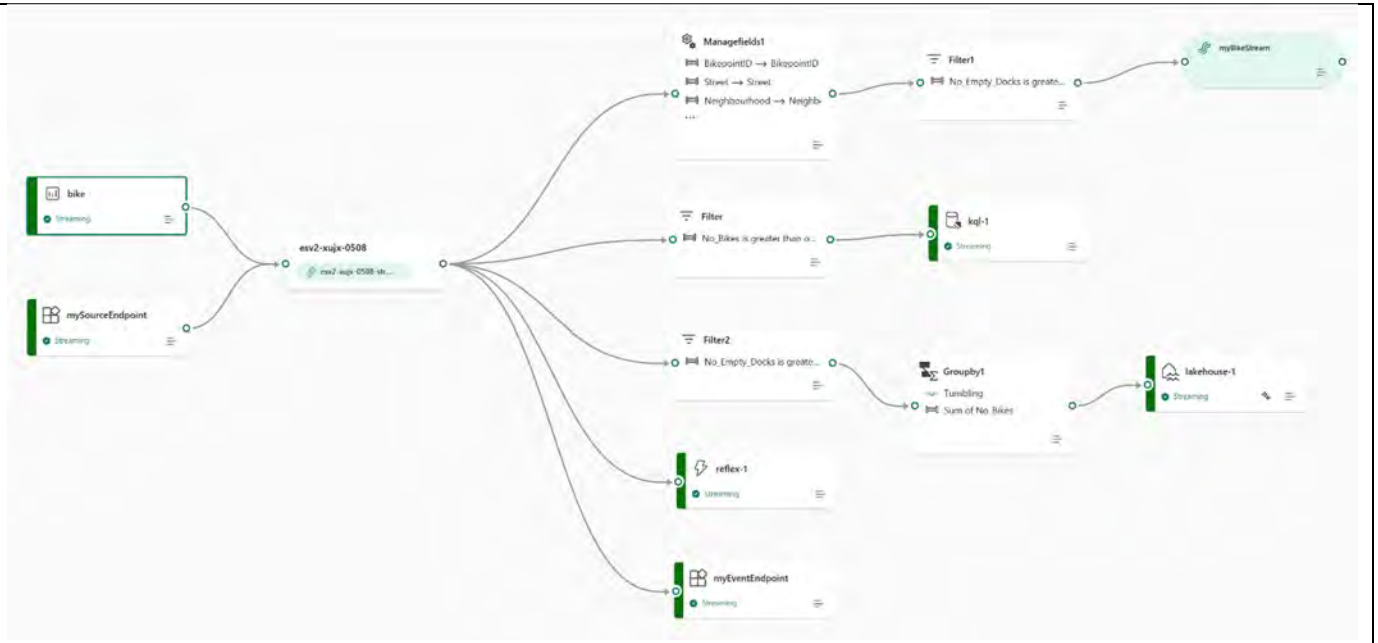
One use of such transformations is to generate derived streams, which are "logical stream[s] of data."[5] Streams may also be processed before routing them to Lakehouse, Eventhouse, and Activator.[6]

Microsoft's literature reproduces examples of such graphs, such as:[7]

[5] Microsoft, *Add a derived stream destination to an eventstream*, available at `https://learn.microsoft.com/en-us/fabric/real-time-intelligence/event-streams/add-destination-derived-stream` [hereinafter *Derived Stream Destination*].

[6] *Event Processing Editor*.

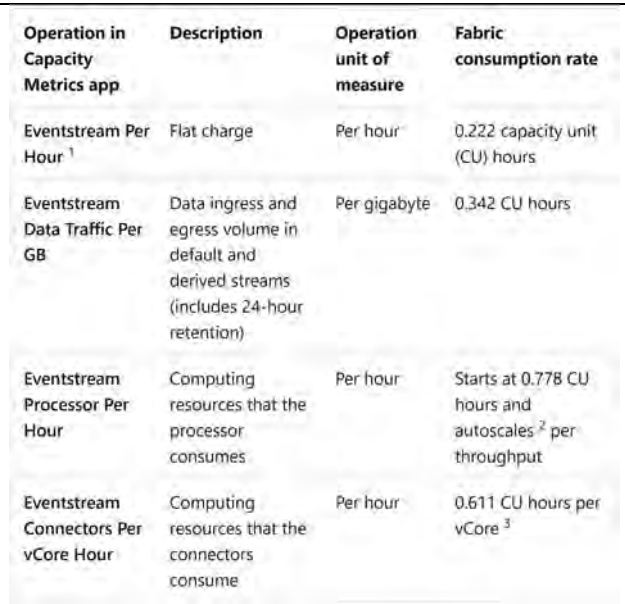
[7] *Eventstreams Overview*.



Moreover, as discussed above Fabric runs on Azure, which is a cloud platform that comprises a plurality of computer systems. Evenstreams scales the number of computer systems used for a stream processing to meet the demands of its workload.[8] For instance, one way that Microsoft charges users for Evenstreams is in "Eventstream Processor Per Hour" and "Evenstream Connectors Per vCore Hour":[9]

[8] *Id.*

[9] Microsoft, *Capacity consumption for Microsoft Fabric eventstreams*, available at `https://learn.microsoft.com/en-us/fabric/real-time-intelligence/event-streams/monitor-capacity-consumption` [hereinafter *Capacity Consumption*].

	 Moreover, Fabric runs on Microsoft Azure; the event processing runs across Fabric "Capacity Units," which Microsoft Azure and Fabric automatically map to multiple computer systems.[10]

Operation in Capacity Metrics app	Description	Operation unit of measure	Fabric consumption rate
Eventstream Per Hour [1]	Flat charge	Per hour	0.222 capacity unit (CU) hours
Eventstream Data Traffic Per GB	Data ingress and egress volume in default and derived streams (includes 24-hour retention)	Per gigabyte	0.342 CU hours
Eventstream Processor Per Hour	Computing resources that the processor consumes	Per hour	Starts at 0.778 CU hours and autoscales [2] per throughput
Eventstream Connectors Per vCore Hour	Computing resources that the connectors consume	Per hour	0.611 CU hours per vCore [3]

[10] Microsoft, *Understand Microsoft Fabric licenses*, `https://learn.microsoft.com/en-us/fabric/enterprise/licenses` [hereinafter *Understand Fabric Licenses*].

SKU*	Capacity Units (CU)	Power BI SKU	Power BI v-cores
F2	2	-	0.25
F4	4	-	0.5
F8	8	EM/A1	1
F16	16	EM2/A2	2
F32	32	EM3/A3	4
F64	64	P1/A4	8
Trial	64	-	8
F128	128	P2/A5	16
F256	256	P3/A6	32
F512	512	P4/A7	64
F1024	1024	P5/A8	128
F2048	2048	-	256

Microsoft recommends that Fabric eventstreams be used with "at least four capacity units":[11]

> ⓘ Note
>
> We recommend that you use the Fabric eventstreams feature with at least four capacity units (SKU: F4).

Microsoft's literature explains that Fabric is a SaaS application:[12]

[11] *Eventstreams Overview*.

[12] *What is Microsoft Fabric?*

	 SaaS applications, as explained by Microsoft’s literature, “are highly scalable, allowing customers to adjust their subscription levels as their needs evolve,” including by “adding features, increasing storage, or supporting more users.”[13] Based on Qomplx’s analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.
wherein a first computer system of the first plurality of computer systems is configured to:	A first computer system of the first plurality of computer systems of the Accused ’424 Products is configured to receive a stream of input data from an input feed.

[13] Microsoft, *What is software as a service (SaaS)?*, available at `https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-saas` [hereinafter *What is SaaS?*].

receive a first stream of input data from a first input feed,	For example, an eventstream "allows you to ingest, transform, and route data streams to various destinations in Fabric."[14] A stream, in turn, is "a continuous and unbounded flow of data events":[15] • A **stream** is a continuous and unbounded flow of data events. When data events flow into Fabric event streams, the stream forms inside Fabric event streams to represent the data events. There are two kinds of streams in Fabric event streams: For example, eventstreams can receive various streams of input data from various input feeds, in order to "[b]ring events into Fabric":[16] **Bring events into Fabric** Eventstreams provide you with source connectors to fetch event data from the various sources. More sources are available when you enable **Enhanced capabilities** at the time of creating an eventstream. The events include, but are not limited to, Apache Kafka, Fabric events, and MQTT, among other examples. Additionally, eventstreams can receive streams of input data from other computing steps.

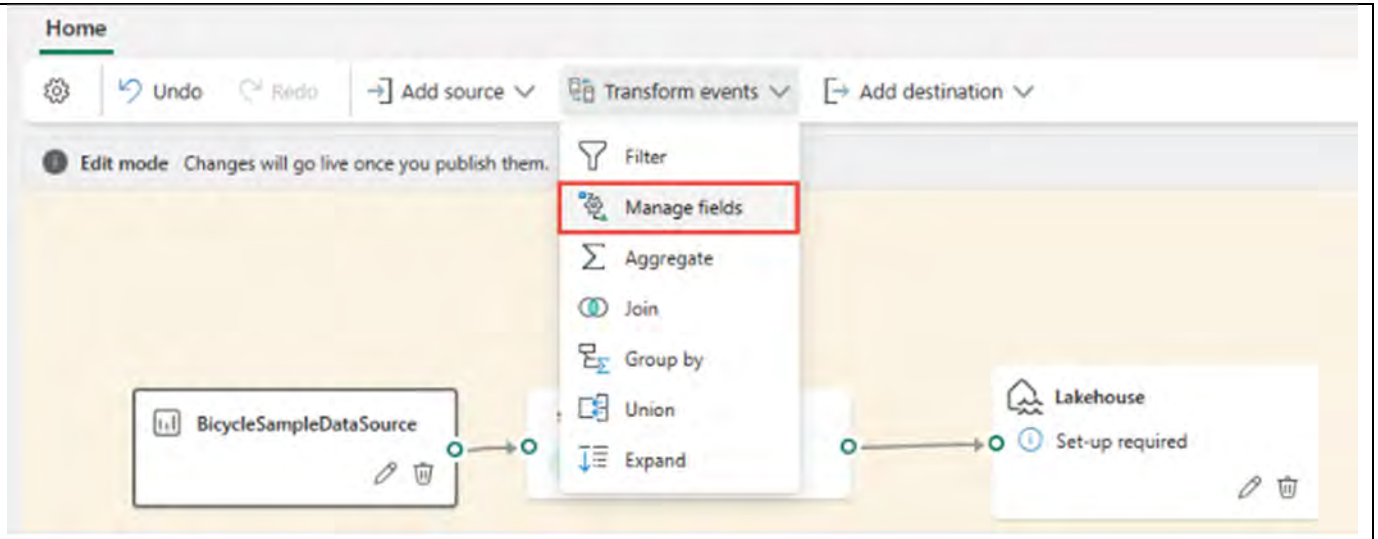
[14] Microsoft, *Create default and derived eventstreams*, available at `https://learn.microsoft.com/en-us/fabric/real-time-intelligence/event-streams/create-default-derived-streams` [hereinafter *Default and Derived Eventstreams*].

[15] *Default and Derived Eventstreams*.

[16] *Eventstreams Overview*.

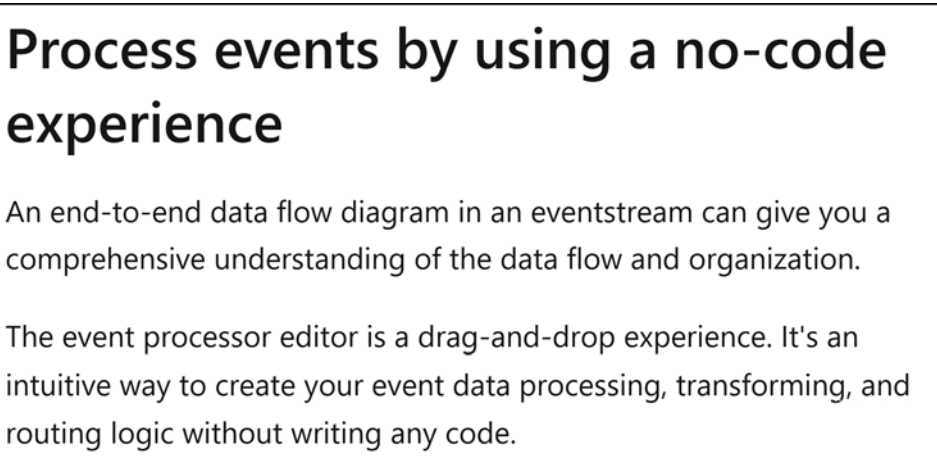
	Based on Qomplx’s analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.
process the first stream of input data substantially in real time by executing software instructions that apply the first transformation pipeline to the first stream of input data to generate first pipeline output messages, process the respective first data stored in the memory of the first computer system to determine information about the second transformation pipeline, and transmit the first pipeline output messages to a second computer system of the first plurality of computer systems in accordance with the determined information,	The first computer system of the first plurality of computer systems of the Accused ’424 Products is configured to process the stream of input data substantially in real time by executing software instructions that apply the first transformation pipeline to the stream of input data to generate first pipeline output messages, process the respective first data to determine information about the second transformation pipeline, and transmit the first pipeline output messages to a second computer system of the first plurality of computer systems of the Accused ’424 Products in accordance with the determined information. A second computer system of the first plurality of computer systems of the Accused ’424 Products is configured to receive the first pipeline output messages, process those messages substantially in real time by applying the second transformation pipeline to the first pipeline output messages and generating second pipeline output messages. For example, as discussed above, eventstreams allows a user to define portions of a distributed computational graph that represents transformation pipelines as nodes and the flow of data streams between the transformation pipelines as directional edges between the nodes:[17]

[17] *Event Processing Editor*.



Microsoft’s documentation describes eventstreams’ event processing editor as way to “create your event data processing, transforming, and routing logic without writing any code”:[18]

[18] *Eventstreams Overview*.



Thus, a first computer of Microsoft Azure executes software instructions that apply a first transformation pipeline (including, as suggested by the exemplary screenshot above, but not limited to, a “Manage fields” transformation pipeline) configured by the user to a first stream of data.

Fabric allows a user to apply various transformations that receive, process substantially in real time (including by applying transformation pipelines), and then transmit streams of data:[19]

[19] *Eventstreams Overview*.

Process events using no-code experience

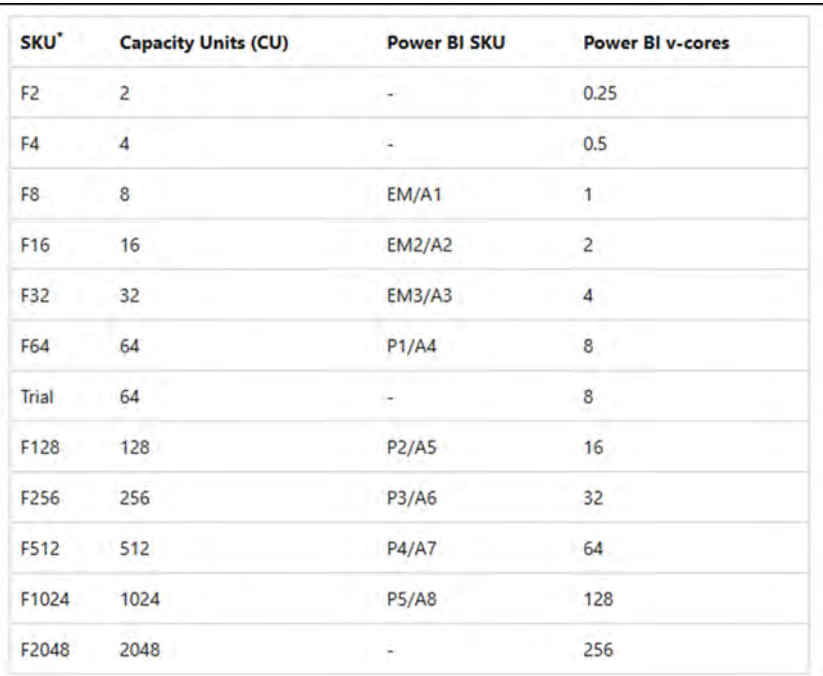
The drag and drop experience gives you an intuitive and easy way to create your event data processing, transforming, and routing logic without writing any code. An end-to-end data flow diagram in an eventstream can provide you with a comprehensive understanding of the data flow and organization. The event processor editor is a no-code experience that allows you to drag and drop to design the event data processing logic.

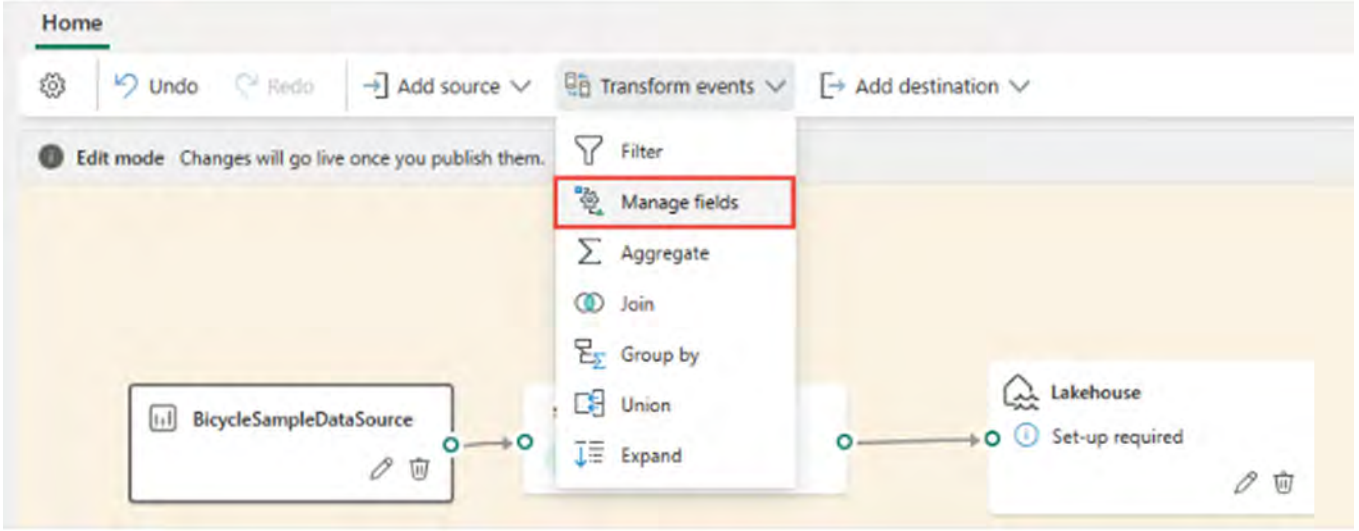
Expand table

Transformation	Description
Filter	Use the Filter transformation to filter events based on the value of a field in the input. Depending on the data type (number or text), the transformation keeps the values that match the selected condition, such as `is null` or `is not null`.
Manage fields	The Manage fields transformation allows you to add, remove, change data type, or rename fields coming in from an input or another transformation.
Aggregate	Use the Aggregate transformation to calculate an aggregation (Sum, Minimum, Maximum, or Average) every time a new event occurs over a period of time. This operation also allows for the renaming of these calculated columns, and filtering or slicing the aggregation based on other dimensions in your data. You can have one or more aggregations in the same transformation.
Group by	Use the Group by transformation to calculate aggregations across all events within a certain time window. You can group by the values in one or more fields. It's like the Aggregate transformation allows for the renaming of columns, but provides more options for aggregation and includes more complex options for time windows. Like Aggregate, you can add more than one aggregation per transformation.
Union	Use the Union transformation to connect two or more nodes and add events with shared fields (with the same name and data type) into one table. Fields that don't match are dropped and not included in the output.
Expand	Use the Expand array transformation to create a new row for each value within an array.

As discussed above, Fabric is a SaaS application that runs on Microsoft Azure; the transformation pipelines run across Fabric “Capacity Units,” which Microsoft Azure and Fabric automatically map to multiple computer systems:[20]

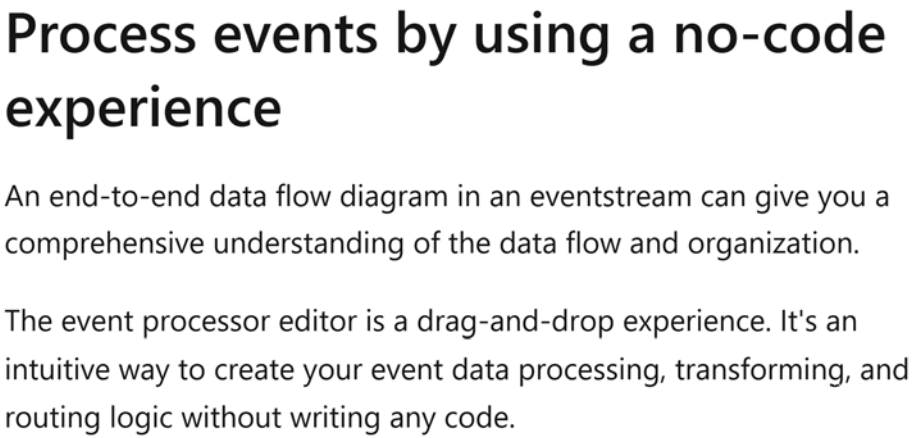
[20] *Understand Fabric Licenses*.

	 Based on Qomplx’s analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.
wherein the second computer system is configured to: receive the first pipeline output messages, and process the first pipeline output messages substantially	The second computer system of the Accused ’424 Products is configured to receive the first pipeline output messages, and process the first pipeline output messages substantially in real time by executing software instructions that apply the second transformation pipeline to the first pipeline output messages to generate second pipeline output messages. The first and second computer systems are distinct.

in real time by executing software instructions that apply the second transformation pipeline to the first pipeline output messages to generate second pipeline output messages, wherein the first and second computer systems are distinct; and	For example, as discussed above, eventstreams allows a user to define portions of a distributed computational graph that represents transformation pipelines as nodes and the flow of data streams between the transformation pipelines as directional edges between the nodes:[21]  Microsoft’s documentation describes eventstreams’ event processing editor as way to “create your event data processing, transforming, and routing logic without writing any code”:[22]

[21] *Event Processing Editor*.

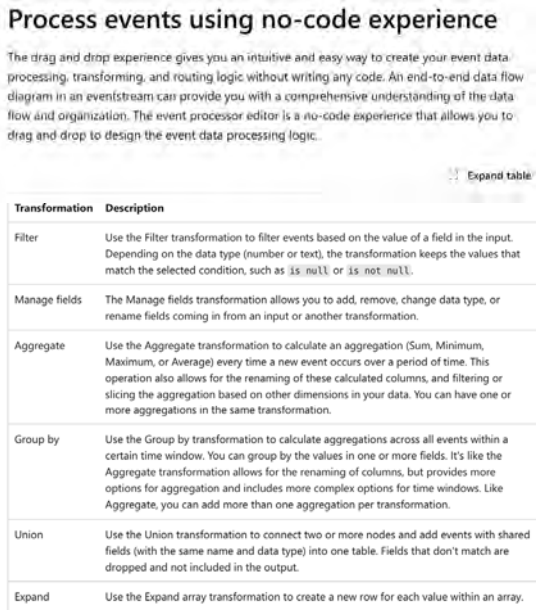
[22] *Eventstreams Overview*.



Thus, a second computer of Microsoft Azure executes software instructions that apply a second transformation pipeline configured by the user to the first pipeline output messages to generate second pipeline output messages.

Fabric allows a user to apply various transformations that receive, process substantially in real time (including by applying transformation pipelines), and then transmit streams of data:[23]

[23] *Id*.



Process events using no-code experience

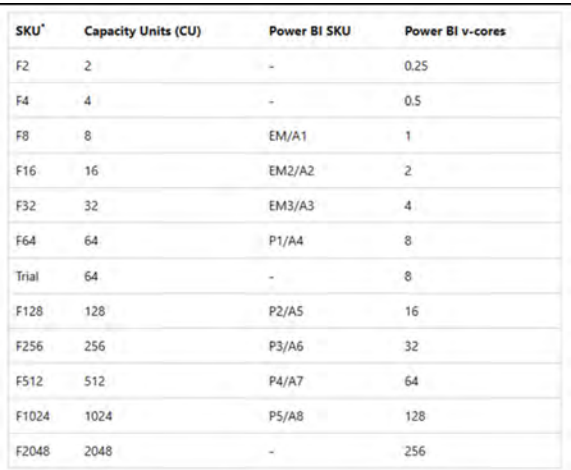
The drag and drop experience gives you an intuitive and easy way to create your event data processing, transforming, and routing logic without writing any code. An end-to-end data flow diagram in an eventstream can provide you with a comprehensive understanding of the data flow and organization. The event processor editor is a no-code experience that allows you to drag and drop to design the event data processing logic.

Expand table

Transformation	Description
Filter	Use the Filter transformation to filter events based on the value of a field in the input. Depending on the data type (number or text), the transformation keeps the values that match the selected condition, such as `is null` or `is not null`.
Manage fields	The Manage fields transformation allows you to add, remove, change data type, or rename fields coming in from an input or another transformation.
Aggregate	Use the Aggregate transformation to calculate an aggregation (Sum, Minimum, Maximum, or Average) every time a new event occurs over a period of time. This operation also allows for the renaming of these calculated columns, and filtering or slicing the aggregation based on other dimensions in your data. You can have one or more aggregations in the same transformation.
Group by	Use the Group by transformation to calculate aggregations across all events within a certain time window. You can group by the values in one or more fields. It's like the Aggregate transformation allows for the renaming of columns, but provides more options for aggregation and includes more complex options for time windows. Like Aggregate, you can add more than one aggregation per transformation.
Union	Use the Union transformation to connect two or more nodes and add events with shared fields (with the same name and data type) into one table. Fields that don't match are dropped and not included in the output.
Expand	Use the Expand array transformation to create a new row for each value within an array.

As discussed above, Fabric is a SaaS application that runs on Microsoft Azure; the transformation pipelines run across Fabric “Capacity Units,” which Microsoft Azure and Fabric automatically map to multiple computer systems:[24]

[24] *Understand Fabric Licenses*.

	 Based on Qomplx’s analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.
a second plurality of computer systems; wherein a third computer system of the first plurality of computer systems is configured to execute software instructions that cause a fourth computer system of the second plurality of computer systems to execute software	The Accused ’424 Products comprise a second plurality of computer systems, wherein a third computer system of the first plurality of computer systems of the Accused ’424 Products is configured to execute software instructions that cause a fourth computer system of the second plurality of computer systems of the Accused ’424 Products execute software instructions that apply at least one of the first transformation pipeline and the second transformation pipeline. For example, Fabric and Microsoft Azure use an autoscaler that automatically recruits additional computer systems to apply the user’s configured processing pipelines as needed to scale up with an increased flow of data through the eventstream:[25]

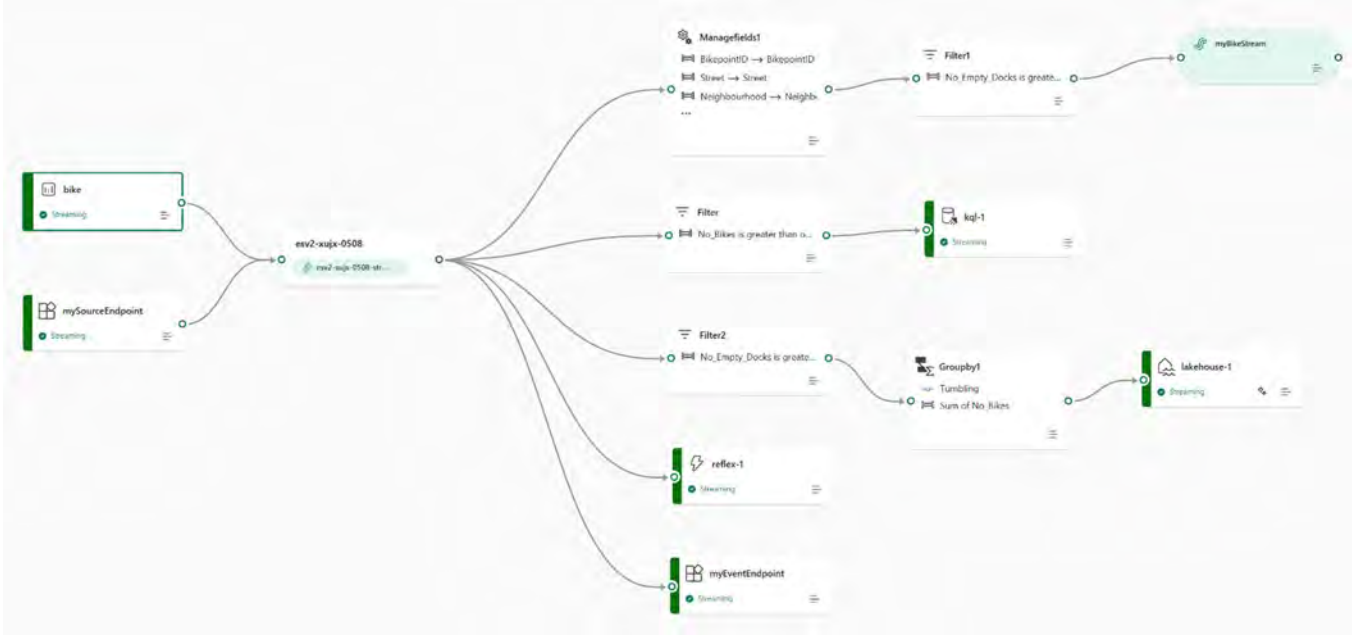
[25] *Capacity Consumption*.

instructions that apply at least one of the first transformation pipeline and the second transformation pipeline.	• **Note 2: Eventstream Processor Per Hour**. The CU consumption rate of the Eventstream processor is correlated to the throughput of event traffic, the complexity of the event processing logic, and the partition count of input data: ◦ With "Low" set in "Event throughput setting", the processor CU consumption rate starts at 1/3 base-rate (0.778 CU hour) and autoscale within 2/3 base-rate (1.555 CU hour), 1 base-rate (2.333 CU hour), 2 base-rates, and 4 base-rates. ◦ With "Medium" set in "Event throughput setting", the processor CU consumption rate starts at 1 base-rate and autoscale within multiple possible base-rates. ◦ With "High" set in "Event throughput setting", the processor CU consumption rate starts at 2 base-rates and autoscale within multiple possible base-rates. Other Azure products such as Microsoft Service Fabric show that units of computation are often distributed "across a cluster of machines" "at massive scale":[26] **Container orchestration** Service Fabric is Microsoft's container orchestrator for deploying and managing microservices across a cluster of machines, benefiting from the lessons learned running Microsoft services at massive scale. Service Fabric can deploy applications in seconds, at high density with hundreds or thousands of applications or containers per machine. With Service Fabric, you can mix both services in processes and services in containers in the same application. Based on Qomplx's analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents.

[26] Microsoft, *Overview of Azure Service Fabric*, `https://learn.microsoft.com/en-us/azure/service-fabric/service-fabric-overview` [hereinafter *Overview of Azure Service Fabric*].

	For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.

II. Claim 2

The distributed computing cluster of claim 1, wherein:	See above for an analysis of Claim 1.
at least one of the first transformation pipeline and the second transformation pipeline is non-linear.	At least one of the first transformation pipeline and the second transformation pipeline of Claim 1 is non-linear. For example, Microsoft’s documentation shows examples of eventstreams that apparently comprise non-linear transformation pipelines:[27] 

[27] *Eventstreams Overview*.

	As another example, Microsoft's documentation explains that it is possible to "route the streaming data in derived stream to multiple destinations in Fabric."[28] "You can attach multiple destinations in an eventstream to simultaneously receive data from your eventstreams without the eventstreams interfering with each other."[29] It also explains that "schema inferencing in Microsoft Fabric supports inferring multiple schemas from various sources and the evenstream itself."[30] Based on Qomplx's analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.

[28] *Default and Derived Eventstreams*.

[29] *Eventstreams Overview*.

[30] Microsoft, *Enhance event processing by using multiple schema inferencing (preview)*, available at `https://learn.microsoft.com/en-us/fabric/real-time-intelligence/event-streams/process-events-with-multiple-schemas` [hereinafter *Multiple Schema Inferencing*].

III. Claim 3

The distributed computing cluster of claim 2, wherein:	See above for an analysis of Claim 2.
wherein the third computer system of the first plurality of computer systems is configured to: identify a fifth computer system of the first plurality of computer systems, wherein the fifth computer system is not configured to apply the first transformation pipeline to any stream of input data, and cause the fifth computer system to process a second stream of input data received from a second input feed substantially in real time by executing software instructions that apply the first	The third computer system of the first plurality of computer systems of Claim 2 is configured to identify a fifth computer system of the first plurality of computer systems, wherein the fifth computer system is not configured to apply the first transformation pipeline to any stream of input data, and cause the fifth computer system to process a second stream of input data received from a second input feed substantially in real time by executing software instructions that apply the first transformation pipeline to the second stream of input data. For example, Fabric and Microsoft Azure use an autoscaler that automatically recruits additional computer systems to apply the user’s configured processing pipelines in a way that scales up with an increased flow of data through the eventstream.[31] • **Note 2: Eventstream Processor Per Hour**. The CU consumption rate of the Eventstream processor is correlated to the throughput of event traffic, the complexity of the event processing logic, and the partition count of input data: ◦ With "Low" set in "Event throughput setting", the processor CU consumption rate starts at 1/3 base-rate (0.778 CU hour) and autoscale within 2/3 base-rate (1.555 CU hour), 1 base-rate (2.333 CU hour), 2 base-rates, and 4 base-rates. ◦ With "Medium" set in "Event throughput setting", the processor CU consumption rate starts at 1 base-rate and autoscale within multiple possible base-rates. ◦ With "High" set in "Event throughput setting", the processor CU consumption rate starts at 2 base-rates and autoscale within multiple possible base-rates. Other Azure products such as Microsoft Service Fabric show that units of computation are often distributed “across a cluster of machines” “at massive scale”:[32]

[31] *Capacity Consumption*

[32] *Overview of Azure Service Fabric*.

transformation pipeline to the second stream of input data.	 Accordingly, when the first transformation pipeline is connected to a new input feed, for example, using the eventstream editor, Fabric and Microsoft Azure may scale the computing resources of the eventstream by recruiting an additional computer system to process the streaming data received from the new input feed using the first transformation pipeline. Based on Qomplx’s analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.

IV. Claim 5

The distributed computing cluster of claim 2, wherein:	See above for an analysis of Claim 2.
wherein the third computer system of the first plurality of computer systems is configured to: identify a fifth computer system of the first plurality of computer systems, wherein the fifth computer system is not configured to apply the second transformation pipeline to any pipeline output message, and cause the fifth computer system to process third pipeline output messages substantially in real time by executing software instructions that apply the second transformation pipeline to the third pipeline output messages.	The third computer system of the first plurality of computer systems of Claim 2 is configured to identify a fifth computer system of the first plurality of computer systems, wherein the fifth computer system is not configured to apply the second transformation pipeline to any pipeline output message, and cause the fifth computer system to process third pipeline output messages substantially in real time by executing software instructions that apply the second transformation pipeline to the third pipeline output messages. For example, Fabric and Microsoft Azure use an autoscaler that automatically recruits additional computer systems to apply the user’s configured processing pipelines in a way that scales up with increased data flow through the eventstream.[33] • **Note 2: Eventstream Processor Per Hour**. The CU consumption rate of the Eventstream processor is correlated to the throughput of event traffic, the complexity of the event processing logic, and the partition count of input data: ◦ With "Low" set in "Event throughput setting", the processor CU consumption rate starts at 1/3 base-rate (0.778 CU hour) and autoscale within 2/3 base-rate (1.555 CU hour), 1 base-rate (2.333 CU hour), 2 base-rates, and 4 base-rates. ◦ With "Medium" set in "Event throughput setting", the processor CU consumption rate starts at 1 base-rate and autoscale within multiple possible base-rates. ◦ With "High" set in "Event throughput setting", the processor CU consumption rate starts at 2 base-rates and autoscale within multiple possible base-rates. Other Azure products such as Microsoft Service Fabric show that units of computation are often distributed “across a cluster of machines” “at massive scale”:[34]

[33] *Capacity Consumption*.

[34] *Overview of Azure Service Fabric*.

Accordingly, when the second transformation pipeline is not processing its input stream fast enough, or when the second transformation pipeline is connected to another input, for example, using the eventstreams editor, Fabric and Microsoft Azure may scale the computing resources of the eventstream by recruiting an additional computer system to apply the second transformation pipeline, and directing pipeline output messages to the additional computer system.

Based on Qomplx’s analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.

V. Claim 7

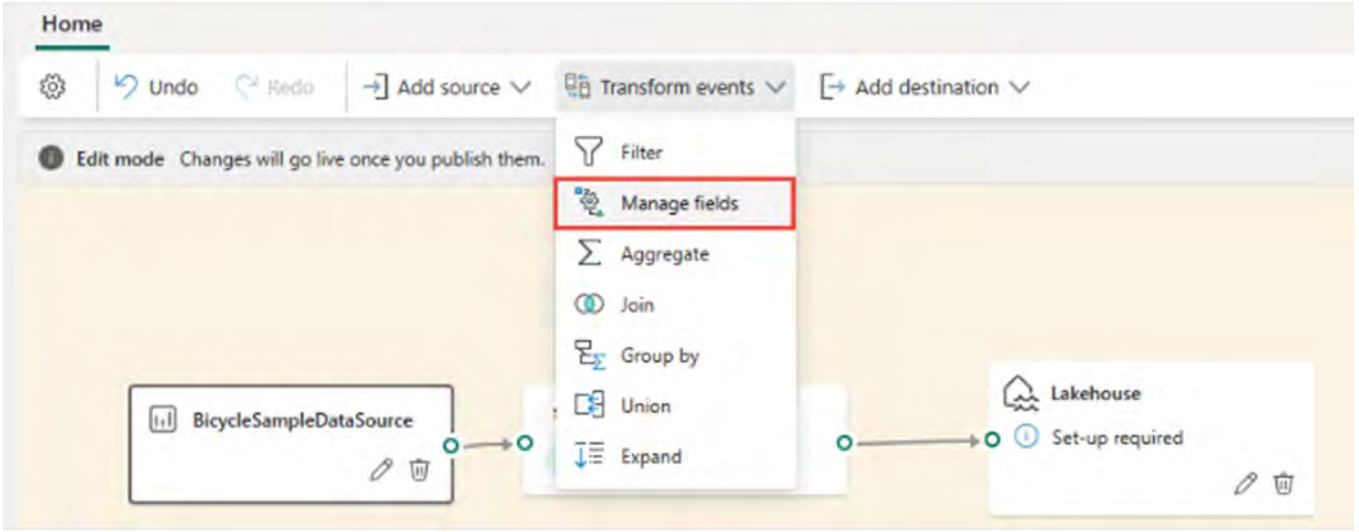
The distributed computing cluster of claim 1,	See above for an analysis of Claim 1.
wherein the first computer system is further configured to transmit the first pipeline output messages to the fourth computer system of the second plurality of computer systems,	The first computer system of Claim 1 is further configured to transmit the first pipeline output messages to the fourth computer system of the second plurality of computer systems. As discussed above, Fabric and Microsoft Azure will scale the computing resources when the second transformation pipeline is not processing its input stream quickly enough. As part of that scaling, Fabric and Microsoft Azure will direct the first pipeline output messages to the newly recruited computer system.
wherein the first computer system is distinct from the fourth computer system, and wherein the second computer system is distinct from the fourth computer system.	The first computer system of Claim 1 is distinct from the fourth computer system and the second computer system is distinct from the fourth computer system. As discussed above, when Fabric and Microsoft Azure scale the computing resources to keep up with increasing amounts of streaming data, they will recruit additional computer systems. Scaling up computer resources includes recruiting a new computer system not previously used by the eventstream as the fourth computer system. Based on Qomplx's analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.

VI. Claim 9

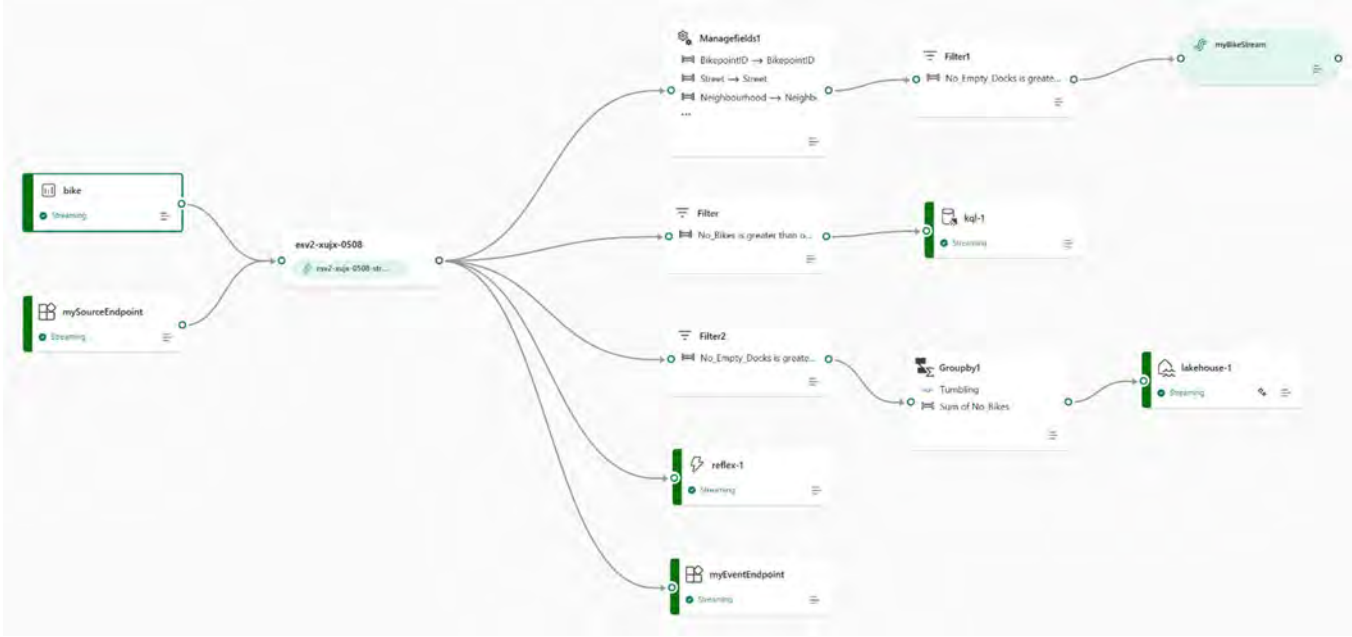
A distributed computing cluster comprising:	The Accused ’424 Products are distributed computing clusters. For example, as discussed above, Fabric is a “Software as a Service” system offering “a data analytics platform,” and runs on Microsoft Azure, a distributed computer cluster:[35] > Microsoft Fabric is a data analytics platform that handles your entire data workflow, from collecting and processing data to building reports and dashboards. It combines data ingestion, transformation, real-time event routing, and visualization through workloads like Data Engineering, Data Factory, Data Science, Real-Time Intelligence, Data Warehouse, and Databases. > As a Software as a Service (SaaS) platform, Fabric centralizes data storage with OneLake. It embeds AI capabilities throughout, so you don't need to integrate services manually, allowing you to transform raw data into actionable insights efficiently. Microsoft Azure includes “400 highly secure datacenters in over 70 regions.”[36] A person of skill in the art would understand that cloud services such as Fabric utilize a plurality of computer systems.

[35] *What is Microsoft Fabric?*.

[36] *What is Azure?*

a first plurality of computer systems, wherein each respective computer system of the first plurality of computer systems comprises a respective memory that stores a respective first data that: describes at least a respective portion of a configuration of a first transformation pipeline, and identifies at least a second transformation pipeline,	The Accused ’424 Products comprise a first plurality of computer systems, wherein each respective computer system of the first plurality of computer systems comprises a respective memory that stores a respective first data that describes at least a respective portion of a configuration of a first transformation pipeline, and identifies at least a second transformation pipeline. For example, eventstreams allows a user to define portions of a distributed computational graph that describes transformation pipelines and the flow of data streams between the transformation pipelines, as shown in the following exemplary graphical representation:[37] 

[37] Microsoft, *Process event data by using the event processing editor*, available at `https://learn.microsoft.com/en-us/fabric/real-time-intelligence/event-streams/process-events-using-event-processor-editor?pivots=enhanced-capabilities` [hereinafter *Event Processing Editor*].

	One use of such transformations is to generate derived streams, which are "logical stream[s] of data."[38] Streams may also be processed before routing them to Lakehouse, Eventhouse, and Activator.[39] Microsoft's literature reproduces examples of such graphs, such as:[40] 

[38] Microsoft, *Add a derived stream destination to an eventstream*, available at `https://learn.microsoft.com/en-us/fabric/real-time-intelligence/event-streams/add-destination-derived-stream` [hereinafter *Derived Stream Destination*].

[39] *Event Processing Editor*.

[40] *Eventstreams Overview*.

Moreover, as discussed above, Eventstream runs on Azure, a cloud platform comprising a plurality of computer systems. Evenstream scales the number of computer systems used for stream processing to meet the demands of its workload.[41] For example, one way that Microsoft charges users for Evenstreams is in "Eventstream Processor Per Hour" and "Evenstream Connectors Per vCore Hour":[42]

Operation in Capacity Metrics app	Description	Operation unit of measure	Fabric consumption rate
Eventstream Per Hour [1]	Flat charge	Per hour	0.222 capacity unit (CU) hours
Eventstream Data Traffic Per GB	Data ingress and egress volume in default and derived streams (includes 24-hour retention)	Per gigabyte	0.342 CU hours
Eventstream Processor Per Hour	Computing resources that the processor consumes	Per hour	Starts at 0.778 CU hours and autoscales [2] per throughput
Eventstream Connectors Per vCore Hour	Computing resources that the connectors consume	Per hour	0.611 CU hours per vCore [3]

Moreover, Fabric runs on Microsoft Azure; the event processing runs across Fabric "Capacity Units," which Microsoft Azure and Fabric automatically map to multiple computer systems.[43]

[41] *Id.*

[42] *Capacity Consumption.*

[43] *Understand Fabric Licenses.*

SKU*	Capacity Units (CU)	Power BI SKU	Power BI v-cores
F2	2	-	0.25
F4	4	-	0.5
F8	8	EM/A1	1
F16	16	EM2/A2	2
F32	32	EM3/A3	4
F64	64	P1/A4	8
Trial	64	-	8
F128	128	P2/A5	16
F256	256	P3/A6	32
F512	512	P4/A7	64
F1024	1024	P5/A8	128
F2048	2048	-	256

Microsoft recommends that Fabric eventstreams be used with “at least four capacity units”:[44]

> ⓘ Note
>
> We recommend that you use the Fabric eventstreams feature with at least four capacity units (SKU: F4).

Microsoft’s literature explains that Fabric is a SaaS application:[45]

[44] *Eventstreams Overview*.

[45] *What is Microsoft Fabric?*

	As a Software as a Service (SaaS) platform, Fabric centralizes data storage with OneLake. It embeds AI capabilities throughout, so you don't need to integrate services manually, allowing you to transform raw data into actionable insights efficiently. As explained by Microsoft’s literature, SaaS applications “are highly scalable, allowing customers to adjust their subscription levels as their needs evolve,” including by “adding features, increasing storage, or supporting more users.”[46] Based on Qomplx’s analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.
wherein a first computer system of the first plurality of computer systems is configured to: receive a first stream of input data from a first input feed,	A first computer system of the first plurality of computer systems of the Accused ’424 Products is configured to receive a stream of input data from an input feed. As explained above, an eventstream “allows you to ingest, transform, and route data streams to various destinations in Fabric.”[47] A stream, in turn, is “a continuous and unbounded flow of data events”:[48]

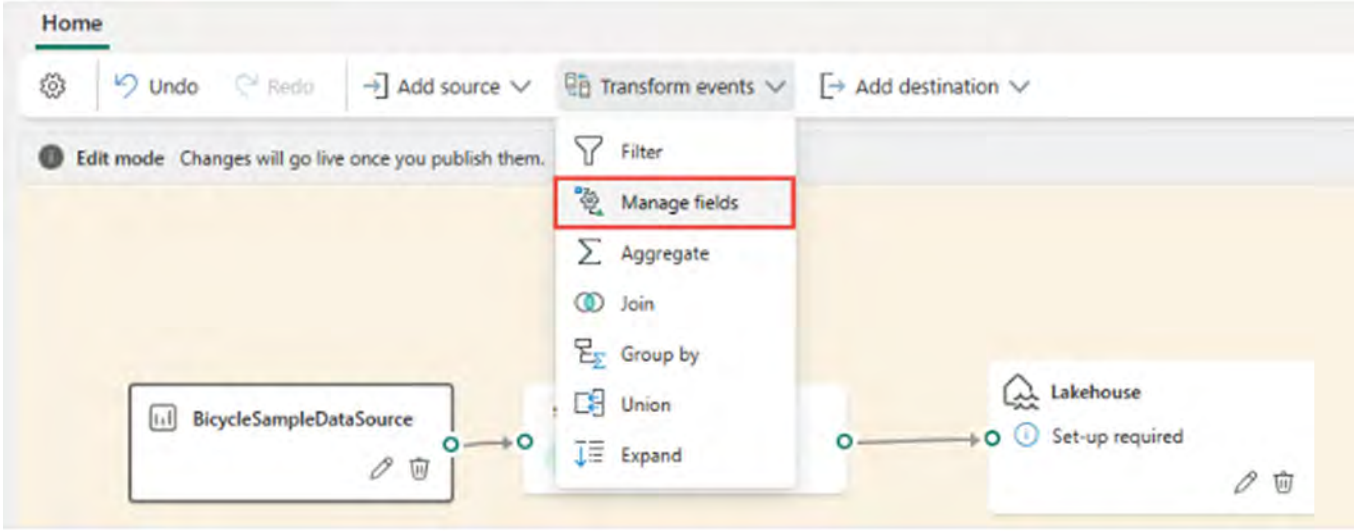
[46] *What is SaaS?*.

[47] *Default and Derived Eventstreams*.

[48] *Id*.

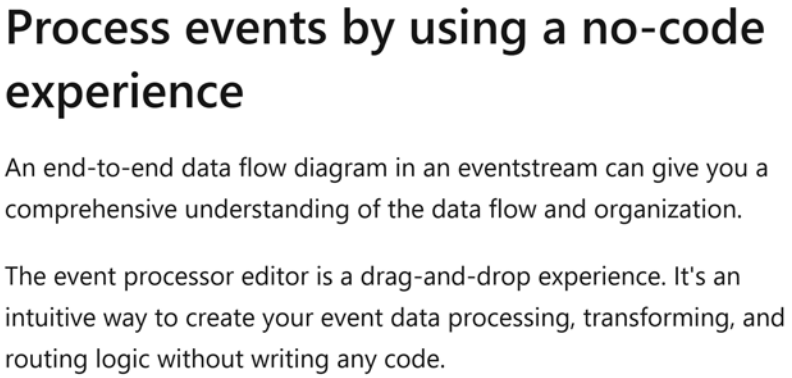
	• A **stream** is a continuous and unbounded flow of data events. When data events flow into Fabric event streams, the stream forms inside Fabric event streams to represent the data events. There are two kinds of streams in Fabric event streams: Eventstreams can receive various streams of input data from various input feeds, in order to “[b]ring events into Fabric”:[49] **Bring events into Fabric** Eventstreams provide you with source connectors to fetch event data from the various sources. More sources are available when you enable **Enhanced capabilities** at the time of creating an eventstream. The events include, but are not limited to, Apache Kafka, Fabric events, and MQTT, among other examples. Additionally, Eventstreams can receive streams of input data from other computing steps. Based on Qomplx’s analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.
process the first stream of input data substantially in real time by executing software	The first computer system of the first plurality of computer systems of the Accused ’424 Productes is configured to process the first stream of input data substantially in real time by executing software instructions that apply the first transformation pipeline to the first stream of input data to generate first

[49] *Eventstreams Overview*.

instructions that apply the first transformation pipeline to the first stream of input data to generate first pipeline output messages, process the respective first data to determine an identification of the second transformation pipeline, and transmit the first pipeline output messages to a second computer system of the first plurality of computer systems in accordance with the determined identification,	pipeline output messages, process the respective first data to determine an identification of the second transformation pipeline, and transmit the first pipeline output messages to a second computer system of the first plurality of computer systems in accordance with the determined identification. For example, as discussed above, Eventstreams allows a user to represent transformation pipelines as nodes and the flow of data streams between the transformation pipelines as directional edges between the nodes:[50]  Microsoft’s documentation goes on to describe eventstreams’ event processing editor as way to “create your event data processing, transforming, and routing logic without writing any code”:[51]

[50] *Event Processing Editor*.

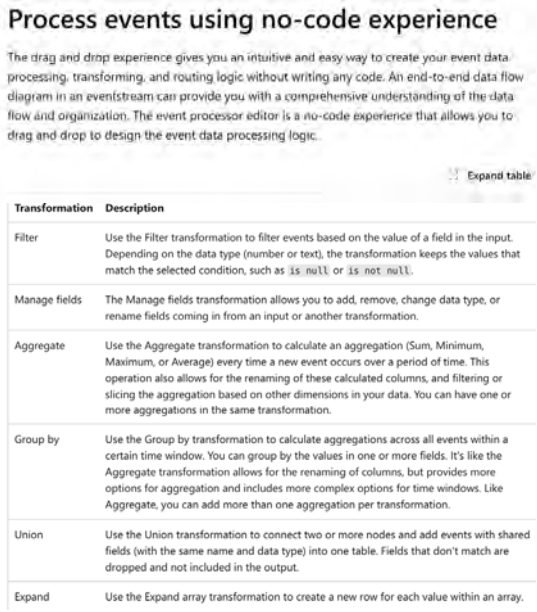
[51] *Eventstreams Overview*.



Thus, a first computer executes software instructions that apply a first transformation pipeline (including, as suggested by the exemplary screenshot above, but not limited to, a “Manage fields” transformation pipeline) configured by the user to a first stream of data (including, as suggested by the example above, but not limited to, a sample data source).

Fabric allows a user to apply various transformations that receive, process substantially in real time (including by applying transformation pipelines), and then transmit streams of data:[52]

[52] *Id*.



Process events using no-code experience

The drag and drop experience gives you an intuitive and easy way to create your event data processing, transforming, and routing logic without writing any code. An end-to-end data flow diagram in an eventstream can provide you with a comprehensive understanding of the data flow and organization. The event processor editor is a no-code experience that allows you to drag and drop to design the event data processing logic.

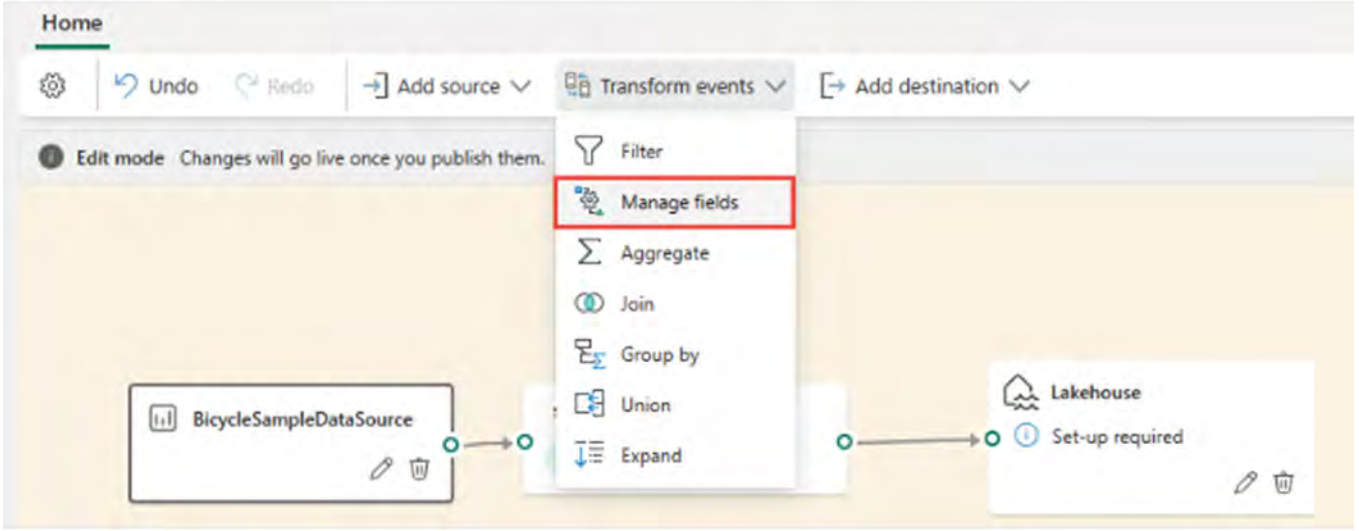
Expand table

Transformation	Description
Filter	Use the Filter transformation to filter events based on the value of a field in the input. Depending on the data type (number or text), the transformation keeps the values that match the selected condition, such as `is null` or `is not null`.
Manage fields	The Manage fields transformation allows you to add, remove, change data type, or rename fields coming in from an input or another transformation.
Aggregate	Use the Aggregate transformation to calculate an aggregation (Sum, Minimum, Maximum, or Average) every time a new event occurs over a period of time. This operation also allows for the renaming of these calculated columns, and filtering or slicing the aggregation based on other dimensions in your data. You can have one or more aggregations in the same transformation.
Group by	Use the Group by transformation to calculate aggregations across all events within a certain time window. You can group by the values in one or more fields. It's like the Aggregate transformation allows for the renaming of columns, but provides more options for aggregation and includes more complex options for time windows. Like Aggregate, you can add more than one aggregation per transformation.
Union	Use the Union transformation to connect two or more nodes and add events with shared fields (with the same name and data type) into one table. Fields that don't match are dropped and not included in the output.
Expand	Use the Expand array transformation to create a new row for each value within an array.

As discussed above, Fabric is a SaaS application that runs on Microsoft Azure; the transformation pipelines run across Fabric “Capacity Units,” which Microsoft Azure and Fabric automatically map to multiple computer systems.[53]

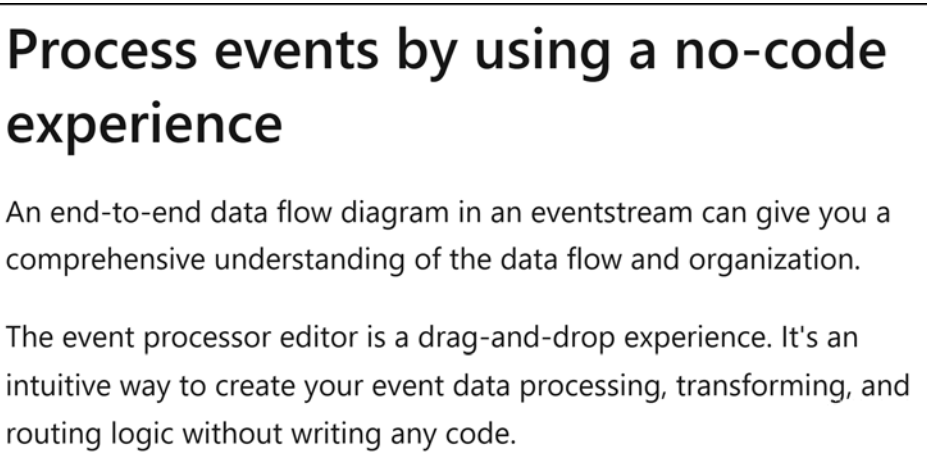
[53] *Understand Fabric Licenses*.

	<table><tr><th>SKU*</th><th>Capacity Units (CU)</th><th>Power BI SKU</th><th>Power BI v-cores</th></tr><tr><td>F2</td><td>2</td><td>-</td><td>0.25</td></tr><tr><td>F4</td><td>4</td><td>-</td><td>0.5</td></tr><tr><td>F8</td><td>8</td><td>EM/A1</td><td>1</td></tr><tr><td>F16</td><td>16</td><td>EM2/A2</td><td>2</td></tr><tr><td>F32</td><td>32</td><td>EM3/A3</td><td>4</td></tr><tr><td>F64</td><td>64</td><td>P1/A4</td><td>8</td></tr><tr><td>Trial</td><td>64</td><td>-</td><td>8</td></tr><tr><td>F128</td><td>128</td><td>P2/A5</td><td>16</td></tr><tr><td>F256</td><td>256</td><td>P3/A6</td><td>32</td></tr><tr><td>F512</td><td>512</td><td>P4/A7</td><td>64</td></tr><tr><td>F1024</td><td>1024</td><td>P5/A8</td><td>128</td></tr><tr><td>F2048</td><td>2048</td><td>-</td><td>256</td></tr></table> Based on Qomplx’s analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.
wherein the second computer system is configured to: receive the first pipeline output messages,	The second computer system of the Accused ’424 Products is configured to receive the first pipeline output messages, and process the first pipeline output messages substantially in real time by executing software instructions that apply the second transformation pipeline to the first pipeline output messages. The first and second computer systems are distinct.

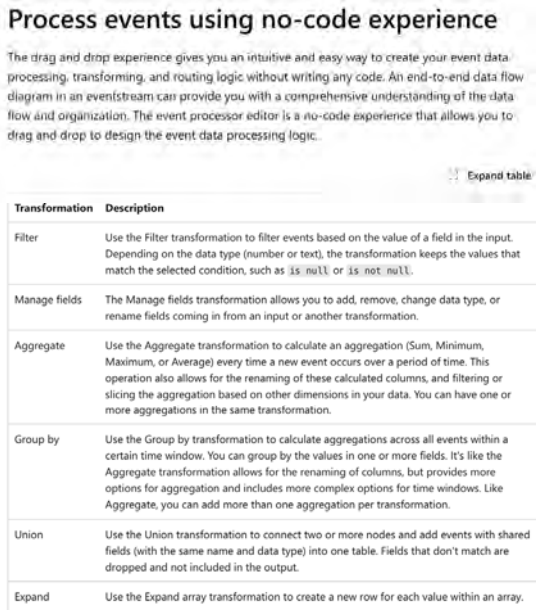
and process the first pipeline output messages substantially in real time by executing software instructions that apply the second transformation pipeline to the first pipeline output messages, wherein the first and second computer systems are distinct; and	For example, as discussed above, Eventstream allows a user to define portions of a distributed computational graph that represents transformation pipelines as nodes and the flow of data streams between the transformation pipelines as directional edges between the nodes:[54]  Microsoft’s documentation describes Eventstream’s event processing editor as way to “create your event data processing, transforming, and routing logic without writing any code”:[55]

[54] *Event Processing Editor*.

[55] *Eventstreams Overview*.

	 Thus, a second computer of Microsoft Azure executes software instructions that apply a second transformation pipeline configured by the user to the first pipeline output messages. Fabric allows a user to apply various transformations that receive, process substantially in real time (including by applying transformation pipelines), and then transmit streams of data:[56]

[56] *Eventstreams Overview*.



Process events using no-code experience

The drag and drop experience gives you an intuitive and easy way to create your event data processing, transforming, and routing logic without writing any code. An end-to-end data flow diagram in an eventstream can provide you with a comprehensive understanding of the data flow and organization. The event processor editor is a no-code experience that allows you to drag and drop to design the event data processing logic.

Expand table

Transformation	Description
Filter	Use the Filter transformation to filter events based on the value of a field in the input. Depending on the data type (number or text), the transformation keeps the values that match the selected condition, such as `is null` or `is not null`.
Manage fields	The Manage fields transformation allows you to add, remove, change data type, or rename fields coming in from an input or another transformation.
Aggregate	Use the Aggregate transformation to calculate an aggregation (Sum, Minimum, Maximum, or Average) every time a new event occurs over a period of time. This operation also allows for the renaming of these calculated columns, and filtering or slicing the aggregation based on other dimensions in your data. You can have one or more aggregations in the same transformation.
Group by	Use the Group by transformation to calculate aggregations across all events within a certain time window. You can group by the values in one or more fields. It's like the Aggregate transformation allows for the renaming of columns, but provides more options for aggregation and includes more complex options for time windows. Like Aggregate, you can add more than one aggregation per transformation.
Union	Use the Union transformation to connect two or more nodes and add events with shared fields (with the same name and data type) into one table. Fields that don't match are dropped and not included in the output.
Expand	Use the Expand array transformation to create a new row for each value within an array.

As discussed above, Fabric is a SaaS application that runs on Microsoft Azure; the transformation pipelines run across Fabric “Capacity Units,” which are automatically mapped to multiple computer systems.[57]

[57] *Understand Fabric Licenses*.

	<table><tr><th>SKU*</th><th>Capacity Units (CU)</th><th>Power BI SKU</th><th>Power BI v-cores</th></tr><tr><td>F2</td><td>2</td><td>-</td><td>0.25</td></tr><tr><td>F4</td><td>4</td><td>-</td><td>0.5</td></tr><tr><td>F8</td><td>8</td><td>EM/A1</td><td>1</td></tr><tr><td>F16</td><td>16</td><td>EM2/A2</td><td>2</td></tr><tr><td>F32</td><td>32</td><td>EM3/A3</td><td>4</td></tr><tr><td>F64</td><td>64</td><td>P1/A4</td><td>8</td></tr><tr><td>Trial</td><td>64</td><td>-</td><td>8</td></tr><tr><td>F128</td><td>128</td><td>P2/A5</td><td>16</td></tr><tr><td>F256</td><td>256</td><td>P3/A6</td><td>32</td></tr><tr><td>F512</td><td>512</td><td>P4/A7</td><td>64</td></tr><tr><td>F1024</td><td>1024</td><td>P5/A8</td><td>128</td></tr><tr><td>F2048</td><td>2048</td><td>-</td><td>256</td></tr></table> Based on Qomplx’s analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.
a second plurality of computer systems; wherein a third computer system of the first plurality of computer systems is configured to execute software	The Accused ’424 Products comprise a second plurality of computer systems, wherein a third computer system of the first plurality of computer systems is configured to execute software instructions that cause the second plurality of computer systems to execute at least one of the first transformation pipeline and the second transformation pipeline.

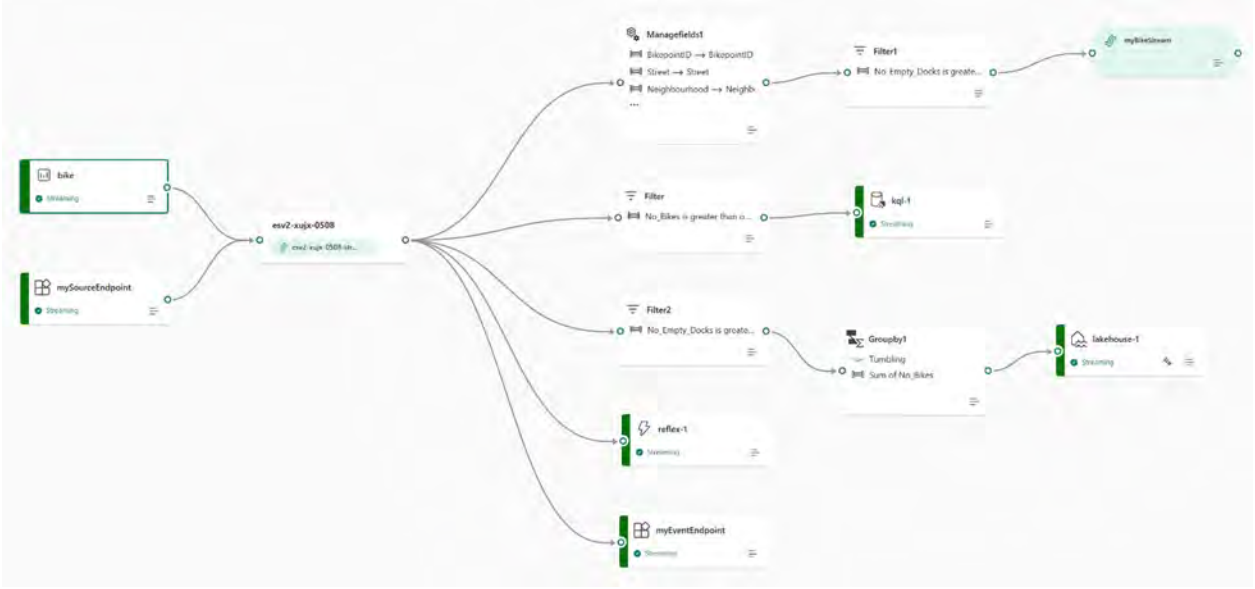
instructions that cause the second plurality of computer systems to execute at least one of the first transformation pipeline and the second transformation pipeline.	For example, Fabric and Microsoft Azure use an autoscaler that automatically recruits additional computer systems to apply the user’s configured processing pipelines in a way that keeps up with increased data flow through an eventstream:[58]
	
	• **Note 2: Eventstream Processor Per Hour**. The CU consumption rate of the Eventstream processor is correlated to the throughput of event traffic, the complexity of the event processing logic, and the partition count of input data:
	  ○ With "Low" set in "Event throughput setting", the processor CU consumption rate starts at 1/3 base-rate (0.778 CU hour) and autoscale within 2/3 base-rate (1.555 CU hour), 1 base-rate (2.333 CU hour), 2 base-rates, and 4 base-rates.
	  ○ With "Medium" set in "Event throughput setting", the processor CU consumption rate starts at 1 base-rate and autoscale within multiple possible base-rates.
	  ○ With "High" set in "Event throughput setting", the processor CU consumption rate starts at 2 base-rates and autoscale within multiple possible base-rates.
	
	Other Azure products such as Microsoft Service Fabric show that units of computation are often distributed “across a cluster of machines” “at massive scale”:[59]

[58] *Capacity Consumption*

[59] *Overview of Azure Service Fabric*.

	 Based on Qomplx’s analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.

VII. Claim 10

The distributed computing cluster of claim 9,	See above for an analysis of Claim 9.
wherein one or more of the first transformation pipeline and the second transformation pipeline is non-linear.	One or more of the first transformation pipeline and the second transformation pipeline of Claim 9 is non-linear. For example, Microsoft’s documentation shows examples of eventstreams that apparently comprise non-linear transformation pipelines:[60] 

[60] *Eventstreams Overview*.

	As another example, Microsoft’s documentation explains that it is possible to “route the streaming data in derived stream to multiple destinations in Fabric.”[61] “You can attach multiple destinations in an eventstream to simultaneously receive data from your eventstreams without the eventstreams interfering with each other.”[62] It also explains that “schema inferencing in Microsoft Fabric supports inferring multiple schemas from various sources and the evenstream itself.”[63] Based on Qomplx’s analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.

[61] *Default and Derived Eventstreams*.

[62] *Eventstreams Overview*.

[63] *Multiple Schema Inferencing*.

VIII. Claim 11

The distributed computing cluster of claim 10,	See above for an analysis of Claim 10.
wherein the third computer system of the first plurality of computer systems is configured to: identify a fourth computer system of the first plurality of computer systems, wherein the fourth computer system is not configured to apply the first transformation pipeline to any stream of input data, and cause the fourth computer system to process a second stream of input data received from a second input feed substantially in real time by executing software instructions that apply the first	The third computer system of the first plurality of computer systems of Claim 10 is configured to identify a fourth computer system of the first plurality of computer systems, wherein the fourth computer system is not configured to apply the first transformation pipeline to any stream of input data, and cause the fourth computer system to process a second stream of input data received from a second input feed substantially in real time by executing software instructions that apply the first transformation pipeline to the second stream of input data. For example, as explained above, Fabric and Microsoft Azure use an autoscaler that automatically recruits additional computer systems to apply the user’s configured processing pipelines to keep up with increased flow of data through the eventstream.[64] • **Note 2: Eventstream Processor Per Hour**. The CU consumption rate of the Eventstream processor is correlated to the throughput of event traffic, the complexity of the event processing logic, and the partition count of input data: ◦ With "Low" set in "Event throughput setting", the processor CU consumption rate starts at 1/3 base-rate (0.778 CU hour) and autoscale within 2/3 base-rate (1.555 CU hour), 1 base-rate (2.333 CU hour), 2 base-rates, and 4 base-rates. ◦ With "Medium" set in "Event throughput setting", the processor CU consumption rate starts at 1 base-rate and autoscale within multiple possible base-rates. ◦ With "High" set in "Event throughput setting", the processor CU consumption rate starts at 2 base-rates and autoscale within multiple possible base-rates. Other Azure products such as Microsoft Service Fabric show that units of computation are often distributed “across a cluster of machines” “at massive scale”:[65]

[64] *Capacity Consumption*

[65] *Overview of Azure Service Fabric*.

transformation pipeline to the second stream of input data.	 Accordingly, when the first transformation pipeline is connected to a new input feed, for example, using the eventstream editor, Fabric and Azure may scale up the computing resources of the event stream by recruiting an additional computer system to process the streaming data received from the new input feed using the first transformation pipeline. Based on Qomplx’s analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.

IX. Claim 13

The distributed computing cluster of claim 9,	See above for an analysis of Claim 9.
wherein the first computer system is further configured to transmit the first pipeline output messages to a fourth computer system of the second plurality of computer systems,	The first computer system of Claim 9 is further configured to transmit the first pipeline output messages to a fourth computer system of the second plurality of computer systems. As discussed above, Fabric and Microsoft Azure scale computing resources when the second transformation pipeline is not processing its input stream quickly enough. As part of that scaling, Fabric and Microsoft Azure will direct the first pipeline output messages to the newly recruited computer system. Based on Qomplx's analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.
wherein the first computer system is distinct from the fourth computer system, and wherein the second computer system is distinct from the fourth computer system.	The first computer system of Claim 1 is distinct from the fourth computer system, and the second computer system is distinct from the fourth computer system. As discussed above, when Fabric and Microsoft Azure scale the computing resources to keep up with increasing amounts of streaming data, they will recruit additional computer systems. Scaling up computer resources includes recruiting a new computer system not previously used by the eventstream as the fourth computer system:[66]

[66] *Capacity Consumption*.

	• **Note 2: Eventstream Processor Per Hour**. The CU consumption rate of the Eventstream processor is correlated to the throughput of event traffic, the complexity of the event processing logic, and the partition count of input data: ◦ With "Low" set in "Event throughput setting", the processor CU consumption rate starts at 1/3 base-rate (0.778 CU hour) and autoscale within 2/3 base-rate (1.555 CU hour), 1 base-rate (2.333 CU hour), 2 base-rates, and 4 base-rates. ◦ With "Medium" set in "Event throughput setting", the processor CU consumption rate starts at 1 base-rate and autoscale within multiple possible base-rates. ◦ With "High" set in "Event throughput setting", the processor CU consumption rate starts at 2 base-rates and autoscale within multiple possible base-rates. Based on Qomplx’s analysis to date, this element appears to be practiced literally, including as charted above. Any differences between the charted material and this claim element are expected to be insubstantial, in which case the element is expected to be practiced under the Doctrine of Equivalents. For example, the technology described above performs substantially the same function in substantially the same way to achieve substantially the same result as this claim element.