DataStream User Guide

April 11, 2022

Contents

Welcome to DataStream 1

DataStream 1 is a reporting service that lets you monitor all request-response cycles delivered through the Akamai platform, and sends the transaction log data to one or more destinations. It provides real-time access to application activity data, including aggregated metrics on complete request-response cycles as well as origin response times.

DataStream 1 complements your existing application performance monitoring (APM) infrastructure by offering a secure, simple, and transparent point of integration for accessing data and applications delivered over the Akamai platform.

⚠ **Attention:** DataStream 1 now is an End of Sale (EOS) product under decommissioning. The contents of this document are archived and may be currently inaccurate. Migrate your streams to DataStream 2 before this product becomes End of Life (EOL) to ensure uninterrupted data flow. See the *DataStream migration* guide for details on how to migrate and the differences between DataStream 1 and DataStream 2.

This feature helps simplify web application monitoring and includes applications deployed to cloud hosting providers and applications that leverage third-party services.

To integrate DataStream 1 API data feed with your existing APM tool set, use Property Manager to enable DataStream in your existing configuration and monitoring infrastructure.

DataStream 1 doesn't require changes to your origin application code, networking infrastructure, or end-user device instrumentation.

Here are the key features of DataStream 1 and what challenges they can help you solve:

DataStream 1 features

Challenge	DataStream 1 solution
Obtaining real-time data to check the health of your CDN	DataStream 1 provides easy control over your streams and delivers live log data at low latency, under one minute. You can: • Create, edit, view, and delete stream configurations. DataStream 1 adapts to configuration changes within an hour of the change. • Collect, aggregate, and stream raw log records to chosen destinations at selected time windows. • Pause and stop log deliveries as required. • Monitor multiple properties in a data stream.

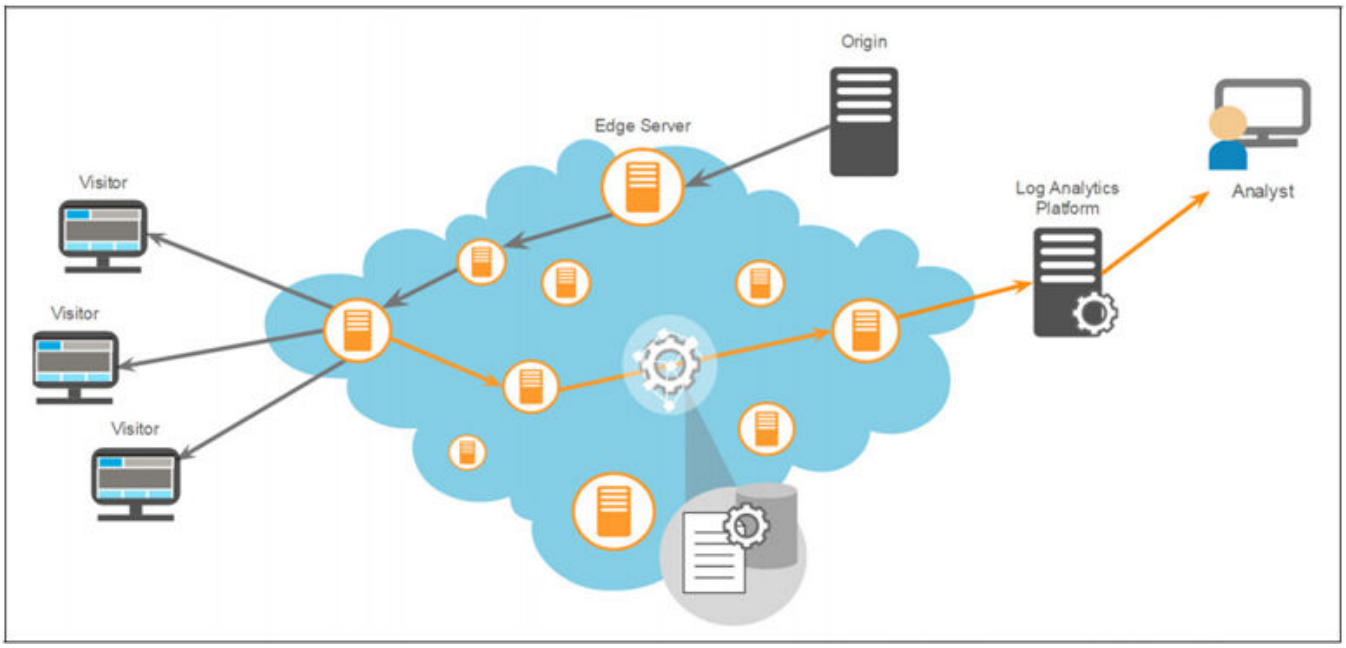
Challenge	DataStream 1 solution
	• Monitor a property in multiple data streams. • Send your log data to one or more destinations.
Sending data to multiple destinations	DataStream 1 supports a push mechanism to upload logs to multiple customer-owned destinations. You can send your log data to: • Amazon S3 • Azure Storage • Datadog • Sumo Logic • Splunk • Custom HTTPS endpoint
Storing and accessing data on demand without maintaining any endpoints or destinations	DataStream 1 provides a buffer that you can use to store your logs and fetch them with the *DataStream Pull API*. You can: • Select a DataStream buffer as a destination. See *Stream logs to a DataStream buffer* on page 32. • Use the DataStream 1 Buffer Pull API to view data from the previous 12 hours. • Integrate web performance data directly into your existing reporting environment. See *Pull integrations* on page 38.
Granular control and over data fields monitored by a data stream	DataStream 1 provides control over what request-response cycle data and metrics you collect and send to your destination or store in a DataStream 1 buffer. You can: • Select parameters or aggregation metrics from predefined data sets. • Control the order of parameters in a log line. • Set delivery conditions for your log files.

Challenge	DataStream 1 solution
	• Control aggregation time windows. • Enable logging custom parameters in your data streams.
Aggregating log data on CDN performance	DataStream 1 can deliver pre-aggregated metrics over a specific window of time. Unlike raw logs, this is meaningful, actionable data without requiring significant downstream processing at your end. You can: • Select aggregated metrics that your data stream sends to destinations or makes available with the Pull API. See *Aggregated metrics data set* on page 26.

How DataStream 1 works

DataStream 1 captures a complete request-response cycle between the end users and edge servers as a log record. It then aggregates a set of log records to a file and sends them to a configured destination on meeting specified conditions.

The basic function of the DataStream 1 application occurs after a requesting device receives a response from a property. The edge server collects all data contained in the stream, formats the log line, and submits the data to its local log line aggregation process.



DataStream 1 workflow

DataStream use cases

DataStream proves an unparalleled tool on many occasions. Let's have a look at the most typical ones.

Monitoring

You can configure your data streams to send raw logs every 30 seconds or aggregated logs as often as every minute. This low latency streaming can help you proactively monitor performance, detect, and quickly resolve performance degradations.

You can use a preferred log analytics platform to continuously ingest logs and set up real-time dashboards or alerts. This helps you proactively mitigate connectivity problems, service disruptions, and configuration-tuning complications as well as minimize your mean time to recovery (MTTR).

You can use DataStream for:

Performance and QoS monitoring	• Proactively monitor performance, detect, and quickly resolve performance degradations. • Determine traffic split based on real time performance and availability metrics. • Validate the success of your new code or CDN configuration enhancements in real time. You can do this type of monitoring in production and at scale, without risking downtime or jeopardizing end-user experience. At the same time, you can measure the impact of a recent code change or new deployments on CDN health, efficiency, and usage.
Event monitoring	• Monitor live and streaming events • Monitor software and game releases
Usage tracking	Monitor for usage spikes to avoid exceeding commits.
Alerting	Receive alerts with potential issues based on violation of thresholds.

Troubleshooting

After you've detected an issue, you need the necessary data to investigate and isolate the root cause. The log data you receive can help enable long term trend analysis. You can use your data to perform:

- Forensic analysis
- Session analysis
- Content analysis

DevOps and diagnostics	Understand the impact of changes caused by the configuration, player, or workflow updates You can receive pre-aggregated metrics over a specific window of time. Use this option to easily switch between aggregated and raw data views for diagnostics or root cause analysis.

For example, you can have your aggregated stream always on for a continuous, high-level view of your CDN health. If the aggregated logs identify high error counts or high average origin response times, you can turn on a raw logs stream for root cause analysis and diagnostics.

Benchmarking Access recent data for up to 12 hours on CDN health, efficiency, and usage for analysis and benchmarking, without the overhead of managing an endpoint infrastructure.

Key concepts and terms

Here's a brief overview of key concepts in the DataStream application.

DataStream Lets you create data streams for your properties that provide low latency streaming of data in raw and aggregated forms. You can configure your data stream to either send log data to one or more destinations or integrate it with third-party web services or your own solution and access the data through the DataStream pull API.

Stream Collects logs about edge request-response cycles for multiple properties. Based on the type of destination, it either streams raw data or aggregated metrics to a destination or stores the data in DataStream buffer and makes it available through the DataStream pull API

Types of logs You can configure your data stream to collect and deliver these types of logs:

- **Raw data** Raw log information about individual edge request-response cycles on the Akamai platform. Depending on your stream's destination type, your stream sends bundled JSON logs to destinations or makes these logs available through the DataStream pull API. You can use raw data logs to find details about specific incidents, search the logs for instances using a specific IP address, or analyze the patterns of multiple attacks.
- **Aggregated metrics** Information about edge request-response cycles aggregated over selected metrics and a time window. Depending on your stream's destination type, your stream sends aggregated JSON metrics to destinations or makes these metrics available through the DataStream pull API. You can use aggregated logs to search for the root cause of an error and monitor the performance, security status, and general behavior of an application.

Data set A predefined set of parameters that you can monitor in edge request-response cycles or metrics that you use as categories for aggregation. You can choose or ignore which parameters you want to receive in your logs or metrics over which you want to aggregate these logs. See *Data set parameters* on page 11.

Destination The destination where a data stream sends or stores log data. You can configure multiple destinations for a data stream. These are possible destinations:

- **Push destination** Defines a destination that sends log data to a specific destination for storing, monitoring, and analytical purposes.
  - **Amazon S3** Provides cloud object storage for your data. See *Getting started with Amazon S3*.

Splunk	Provides an interface that lets you search, monitor, and analyze your data. See *Splunk solutions*.
Sumo Logic	Provides a cloud based service that lets you easily collect, centralize, and integrate data sources. See *How it works*.
HTTPS	Provides an ability to send your data to a secure URL of your choosing. Make sure that the URL handles user authentication.
Datadog	Provides an interface that lets you monitor cloud applications through analyzing data. See *Datadog documentation*.

Pull destination Defines a destination that doesn't send log data to any specific destination but stores it in a DataStream buffer and makes it available to your integrations through the DataStream pull API for up to 12 hours. See Integration.

DataStream buffer	Provides storage of log data for up to 12 hours. It lets you access your log data with the DataStream pull API. Note that you can configure only one DataStream buffer for a stream.

Integration A web service that you want to integrate with a data stream and pull your data for analytical purposes. You can only configure an integration for a stream that uses a DataStream buffer as a destination. These are some of the possible integrations:

BigQuery	Provides data analysis without having to take care of the underlying infrastructure. It also lets you visualize your data with an integrated tool called Data Monitor. See *BigQuery integration* on page 39.
Datadog	Provides out-of-the-box integration with DataStream to measure the health of your application performance in real-time. See *Datadog integration* on page 38.

Content delivery property You manage content delivery by creating and configuring delivery properties. These configurations control how edge servers handle and respond to HTTP(S) requests from end users. For example, requests for web sites, software or game downloads, audio and video streams, or images. See *Content delivery products*.

DataStream offers the ability to collect performance logs against selected delivery properties and streams them to configured destinations. This facilitates downstream data reporting, analytics for usage, and delivery performance visibility.

DataStream lets you:

- monitor up to 15 properties in a data stream.

- monitor logs for the same property in multiple data streams.
- enable log collection for all requests within a property.
- enable log collection for requests matching specific criteria.

Pull API A reporting API that lets you access log data collected by your data streams if you *store it in a DataStream buffer*. This API can fetch log data for up to 12 hours. See *DataStream pull API*.

DataStream 1 migration guidance

Learn about the main differences between DataStream 1 and 2, enhancements in DataStream 2, and get to know how to migrate to DataStream 2.

⚠ **Attention:** DataStream 1 now is an End of Sale (EOS) product under decommissioning. The contents of this document are archived and may be currently inaccurate. Migrate your streams to DataStream 2 before this product becomes End of Life (EOL) to ensure uninterrupted data flow. See the *DataStream migration* guide for details on how to migrate and the differences between DataStream 1 and DataStream 2.

The future of data delivery at Akamai is now

Data is more important than ever and Akamai is committed to providing the data our customers need at low latency. DataStream 2 is more than the latest version of DataStream, it is the evolution of our log collection and processing infrastructure in order to provide fast raw data delivery at planetary scale. DataStream 2 provides the same access logs that power our reporting and billing applications, as a result there are some key differences between DataStream 1 and 2 that customers will need to know in order to make migration as seamless as possible.

Key differences between DataStream 1 and 2

DataStream 2 is designed to scale and support data delivery for all requests on the Akamai platform, going beyond the limitations of DataStream 1. DataStream 2 is also included with content delivery and comes with no additional charge. DataStream 2 is still under active development, which means there are a few features that have not yet been implemented in DataStream 2. Check the *DataStream release notes* to stay up to date on new features.

Here are the DataStream 2 product enhancements:

Supported destinations

- Amazon S3
- Azure Blob Storage
- Datadog
- Google Cloud Storage
- Splunk

	• custom HTTPS endpoint
Data fields and formats	• *Available data fields*, including geo data fields • Custom data fields • Structured, space-delimited and JSON logs
Additional features	• No limits, data provided for every edge request • Upload failure alerting

Low latency aggregated data in ACC reports

Akamai understands customers need more than just raw data, and low latency metrics are essential for monitoring. In order to provide consistent, high quality aggregated data with low latency, we are making significant improvements to the reporting experience by leveraging the same log processing infrastructure as DataStream 2.

Using third party cloud destinations, customers may also leverage DataStream 2 raw logs to build their own aggregated reports, which provide additional flexibility to capture the metrics that matter most to their business.

How to migrate to DataStream 2

All customers will need to migrate from DataStream 1 to DataStream 2. There are functional differences between DataStream 1 and 2, so before planning your migration, make sure that your data delivery requirements are met by DataStream 2. You can refer to the list of available and planned features above and in product documentation. When you are ready to proceed with the migration, reach out to your account team to get DataStream 2 enabled in the Akamai Marketplace. If you don’t have access to the Marketplace, reach out to your Akamai sales representative.

Once you have DataStream 2 added to your account, you can configure your DataStream 2 streams by following the steps in the *user guide*. You can enable DataStream 2 alongside Datastream 1 to ensure an uninterrupted flow of data. The decommission of DataStream 1 is planned for May 31st, 2022 to give customers ample time to migrate their existing workflows.

Add a data stream

To monitor and gain real-time access to delivery performance, you need to create a data stream for your properties and specify a data set that you want this stream to deliver. You can associate a property with multiple data streams to collect aggregated and raw data at the same time, specify different data sets that you want to receive about your application, and extract transaction log data for some streams or send it to specific destinations for others.

For example, you may want to create two streams for your property. The first stream is an aggregated metrics stream that sends data to Datadog or Splunk. You can use it to visualize your logs, metrics, and performance data. The second stream is a raw log stream that doesn't push data to any destination but stores it in a DataStream buffer. You can pull raw data monitored by this stream with your custom client and have real-time insight into the performance of your web application at your convenience.

Provide basic details for a data stream

Start by providing basic details and selecting properties for your data stream. By doing this, you tell edge servers which properties to monitor in the data stream. Naming your data stream also lets you search it out easily when you need it.

Before you begin

Determine the main content delivery properties that you want to monitor in your data stream.

How to

1. Log in to Control Center using a **User ID** and **Password** that have been configured for access to DataStream.
2. Go to > **COMMON SERVICES** > **DataStream**.
3. On the **DataStream** page, click **Create stream**.
   A **Create DataStream** wizard opens on a **Configuration** tab.
4. In **Name**, enter a name for the data stream.
5. In **Group**, select the relevant account control group.
6. In **Contract ID**, select the relevant identifier of your contract.
7. In **Include properties**, select up to 15 properties that you want this data stream to collect logs for.

   Note: You can monitor one property in multiple streams.

8. Click **Next** to continue to the **Data Sets** tab.

Next steps

You can specify parameters that you want to monitor in your data stream. See *Select data set parameters for your data stream*.

Select data set parameters for a data stream

First, decide which types of logs you want your data stream to collect, either raw logs or logs aggregated over selected metrics. Here, you can also select or ignore parameters or metrics that you want to focus on in your logs.

Data set parameters that you can select depend on the type of logs your data stream collects. See *Types of logs*.

How to

1. In the **Data Sets** tab, do the following:
2. In **DataStream 1 type**, select either **Raw** or **Aggregated**.
3. Depending on the type of logs you've chosen, select parameters that you want to receive in log lines or metrics that you want to use to aggregate and categorize your logs.

   See *Data set parameters* on page 11.
4. Click **Next** to continue to the **Delivery** tab.

Next steps

Now you can select a destination where you want to send or store the logs that your data stream collects. See *Select a destination for your logs* on page 27.

Data set parameters

Each stream type can collect different sets of data. A data set lets you define the format of the data received by your origin server, giving you the ability to selectively choose or ignore specific parameters from log data fields.

Note: DataStream provides log data in JSON format. See *Report JSON format*.

Raw logs data set

Raw logs data set parameters

Group	Data element	Description
Message exchange data	**Bytes**	The content bytes served in the response.
	Client IP	The IP address of the client.
	Forward hostname	The hostname of the forward origin server where an edge server sends a request.
	HTTP status code	The HTTP response status returned to the client.
	HTTP version	The version of the `HTTP` protocol used for the transaction.
	Protocol type	The protocol of the monitored request-response cycle , either `HTTP` or `HTTPS`.
	Protocol version	The version of SSL for the request-response cycle.

Group	Data element	Description
	Request Host	The value of the `Host` header in the request. It specifies the domain name of the server and the TCP port number on which the server is listening. If no port is included, the default port for the service requested is implied. For example, 443 for an HTTPS URL, and 80 for an HTTP URL. A `Host` header must be present in HTTP/1.1 requests. If a request lacks a `Host` header or has more than one, a server may respond with a 400 status code. See *Host in RFC 7230*.
	Request method	The method of the request. For example, `GET`, `POST`, `PUT`, or `HEAD`.
	Request path	The path to a resource in the incoming URI. It doesn't include query parameters.
	Request port	The port number for the service requested.
	Response Content-Length	The value of the `Content-Length` header in the response. It indicates the size of the entity body sent to the recipient in bytes. See *Content-Length in RFC 7230*.
	Response Content-Type	The value of the `Content-Type` header in the response. It informs about the media type of the returned content. See: *Content-Type in RFC 7231*. *Partial Content in RFC 7233*. *Media types*.
	User-Agent	The value of the `User-Agent` header in the request. It lets edge servers identify the application, operating system, vendor, or version of the requesting user agent. **Note:** To monitor this parameter in your logs, you need to update you data stream's properties to include a **Log Request Details** behavior that logs the **User-Agent** header. See *Enable logging custom parameters* on page 51. See:

Group	Data element	Description
		User-Agent in RFC 7231. *User-Agent in RFC 2616*.
	Query string	The query string in the incoming URI from the client.
Request header data	**Accept-Encoding**	The value of the `Accept-Encoding` header in the request. It informs which content encoding the client is able to understand. It usually is a compression algorithm. The server may not compress the body of a response. It may happen with images where the data to be sent is already compressed, or the server can't afford the computational overhead caused by the compression requirement. See: *Accept-Encoding in RFC 7231*. *IANA list of encodings*.
	Accept-Language	The value of the `Accept-Language` header in the request. It provides a list of acceptable human languages for response. See *Accept-Language in RFC 7231*.
	Authorization	Provides credentials for HTTP authentication. See: *Authorization in RFC 7235*. *IANA registry of authentication schemas*.
	Cache-Control	The directives that must be obeyed by all caching mechanisms along the request-response chain. See *Request Cache-Control Directives in RFC 7234*.
	Connection	The control options for the current connection. It lists the hop-by-hop request headers. See *Connection in RFC 7230*.
	Request Content-MD5	An MD5 digest of the entity body. This digest allows for an end-to-end message integrity check (MIC) of the entity body. See *RFC 1864*.
	Cookie	Lists the HTTP cookies previously sent by the server in the `Set-Cookie` field.

Group	Data element	Description
		Note: To monitor this parameter in your logs, you need to update you data stream's properties to include a **Log Request Details** behavior that logs the **Cookie** header. See *Enable logging custom parameters* on page 51. See *The Cookie Header in RFC 6265*.
	DNT	Requests a web application to disable tracking of an individual user. This is Mozilla's version of the `X-Do-Not-Track` header field and versions of Firefox, Safari, and IE9 support this.
	Expect	Indicates that the client requires particular server behaviors. A server that doesn't understand or is unable to comply with any of the values in this field responds with an appropriate error status such as a 417 Expectation Failed error response. For example, the server may reject a request if its `Content-Length` is too large. See *Expect in RFC 7231*.
	If-Match	Performs an action if the client-supplied entity matches the same entity on the server. For example, when using a `PUT` method, a resource is only updated if it hasn't been modified since the user last updated it. See *If-Match in RFC 7232*.
	If-Modified-Since	Determines whether the item cached is older or newer. The server returns the requested resource with a 200 status only if it has been modified after the given date. The server returns a 304 Not Modified status to requests if content is unchanged. It's ignored when used with `If-None-Match`, unless the server doesn't support `If-None-Match`. See *If-Modified-Since in RFC 7237*.
	If-None-Match	Determines whether the item cached is identical to the requested one. The server returns the requested resource if it matches any of the listed ETags. The server returns a 304 Not Modified status if the requested content is unchanged. When used together with `If-Modified-Since`, `If-None-Match` takes precedence.

Group	Data element	Description
		See *If-None-Match in RFC 7323*
	If-Range	Sends the client any parts of the entity that are missing or sends the client the full entity. This header can be used with the `Last-Modified` or `ETag` header, but not with both. *See If-Range in RFC 7233*.
	If-Unmodified-Since	Only sends the response if the entity hasn't been modified since a specific time. Used with `If-Range`, it ensures that the new requested range comes from an unmodified document. Used with non-safe methods, it can be used to reject editions if the stored object has been modified since the original was retrieved. See *If-Unmodified-Since in RFC 7232*.
	Range	Requests a specific part of an entity by providing a single byte range or a set of byte ranges. Bytes are numbered from 0. See *Range in RFC 7233*.
	Referer	The address of the resource from which the requested URI was followed. **Note:** To monitor this parameter in your logs, you need to update you data stream's properties to include a **Log Request Details** behavior that logs the **Referer** header. See *Enable logging custom parameters* on page 51 See *Referer in RFC 7231*.
	Request time	The time when the edge server accepted the request from the client.
	TE	The transfer encodings the user agent is willing to accept. See *TE in RFC 7230*.
	Upgrade	Allows the client to specify what additional protocols it supports if the server needs to switch protocols. For example to upgrade an HTTP connection to use WebSocket.

Group	Data element	Description
		See *Upgrade in RFC 7230*.
	Via	Any proxies that processed the response. It helps track message forwards, avoid request loops, and identify the protocol capabilities of senders along the request and response chain. See *Via in RFC 7230*.
	X-Forwarded-For	The originating IP address of the client connecting to a web server through an HTTP proxy or load balancer. It helps debug, gather statistics, and generate location-dependent content and by design exposes privacy sensitive information,such as the IP address of the client. See *Forwarded For in RFC 7239*.
	X-Requested-With	Identifies Ajax requests.
Response header data	**Accept-Ranges**	Whether the edge server supports partial requests. With an `Accept-Ranges` header, the client can resume interrupted download instead of downloading the resource from the beginning. See *Accept-Ranges in RFC 7233*.
	Access-Control-Allow-Origin	Whether the response can be shared within the given origin. See *Access-Control-Allow-Origin In Fetch specification*.
	Age	The time in seconds that the object has been in cache. See *Age in RFC 7234*.
	Allow	Lists the request methods supported by a resource. An empty header indicates that the resource allows no request methods. See *Allow in RFC 7231*.
	Cache-Control	Specifies the caching rules for the response. See *Response Cache-Control Directives In RFC7234*.
	Connection	Controls whether the network connection stays open once the current request-response cycle finishes. See *Connection in RFC 7230*.

Group	Data element	Description
	Content-Disposition	Indicates how the content is to be displayed, whether on a screen or as a file download. See *RFC 6266*.
	Content-Encoding	Specifies encodings applied to the entity-body. It informs about how to decode to obtain the media-type referenced by the `Content-Type` header. See *Content-Encoding in RFC 7231*.
	Content-Language	Lists the languages for the intended audiences. See: *Content-Language in RFC 7231*. *Tags for Identifying Languages*.
	Response Content-MD5	Checks the integrity of the message body. See *RFC 1864*.
	Content-Range	Specifies where in a full body message a partial message belongs. See *Date in RFC 7233*.
	Date	The date and time that the message originated. See *Date in RFC 7231*.
	ETag	Identifies a specific version of a resource. It helps caches be more efficient and save bandwidth by eliminating the need to resend a full response if the content hasn't changed. Additionally, etags help prevent simultaneous updates of a resource from overwriting each other. When used with `If-Match` headers, it can detect mid-air edit collisions. When used with `If-None-Match`, it can cache resources that are unchanged. See *ETag in RFC 7237*.
	Expires	The date and time when the message expires. Invalid dates represent a date in the past and mean that the resource is already expired. For example, 0. If the response includes a `Cache-Control` header with the `max-age` or `s-maxage` directives, the `Expires` header is ignored. See *Expires in RFC 7234*.

Group	Data element	Description
	Last-Modified	The date and time when the resource was last modified by the origin. Conditional requests specifying the `If-Modified-Since` or `If-Unmodified-Since` headers use this header. See *Last-Modified in RFC 7232*.
	Link	Links to a resource containing additional information. See: *Link Serialization in HTTP headers in RFC 8288*. *Link in RFC 5988*.
	P3P	States the data that will be collected about requesting users.
	Retry-After	The length of time the user agent should wait before sending a follow-up request. In a 503 (Service Unavailable) response, it indicates how long the service is expected to be unavailable. In a 429 (Too Many Requests) response, it indicates how long to wait before making a new request. In a redirect response, such as 301 (Moved Permanently), it indicates the minimum time that the user agent is asked to wait before issuing the redirected request. See *Retry-After in RFC 7231*.
	Server	Information about the software that the origin server used to handle the request. See *Server in RFC 7231*
	Set-Cookie	Allows sending cookies in the response to the user agent so it can send them back to the server later. See *Set-Cookie in RFC 6265*.
	Trailer	Allows the sender to include additional fields at the end of chunked messages to supply metadata that might be dynamically generated while the message body is sent, such as a message integrity check, digital signature, or post-processing status. See *Trailer in RFC 7230*

Group	Data element	Description
	Transfer-Encoding	Enables the use of metadata fields. It is a hop-by-hop header attached to a message between two nodes, not to a resource. Each segment of a multi-node connection can use different `Transfer-Encoding` values. See *Transfer-Encoding in RFC 7230*.
	Vary	The headers used to determine whether to send the response to a subsequent request without any additional validation. See *Vary in RFC 7230*.
	Via	The protocols used to send the response from the originating server to the requesting client. See *Via in RFC 7230*.
	Warning	Provides information about transformations made to the message's entity body. See: *Warning in RFC 7234*. *HTTP Warn Codes*.
	WWW-Authenticate	The authentication method that should be used to gain access to a resource. It is required for all 401 Unauthorized response messages. See *WWW-Authenticate in RFC 7235*.
	X-Powered-By	The type of technology the web application uses.
Network performance data	**Asnum**	Autonomous systems number for the client request.
	Client RTT	The round trip time (RTT) in milliseconds from when a request goes from a client to an edge server and back again to the starting point.
	Download time	The time in milliseconds from when the edge server first accepts the request to when it sends the last byte, not when the client acknowledges receiving the last byte.
	Download status	The overall download status of an object represented by a series of four boolean values. It provides data in the following format: `<first_byte><last_byte><full_object><if_aborted>` where:

Group	Data element	Description
		• *<first_byte>* specifies whether the edge server returned the first byte of the object. `1` indicates that the server returned the first byte. `0` indicates that it didn't. • *<last_byte>* specifies whether the edge server the last byte of the object. `1` indicates that the server returned the last byte. `0` indicates that it didn't. • *<full_object>* specifies if the edge server returned the full requested object. `1` indicates the edge server returned the requested object. `0` indicates that it didn't. **Note:** Returning a full object may not always mean returning the first and last bytes of an object. When you request a range of bytes, returning a full object means returning the first and last bytes of the requested range. • *<if_aborted>* specifies whether the client aborted the transaction. `1` indicates that the client aborted the transaction. `0` indicates that they didn't.
	Edge IP	The IP address of the edge server that served the response to the client. This is useful when resolving issues with your account representative.
	Error code f29	If there is an error during forwarding requests from an edge server, a string indicating the problem is logged here.
	Error code r14	If there is an error serving the request, a string indicating the problem is logged here.
	Last byte	The last byte of the object was served by this response. `0` would indicate part of a byte-range response.
	Mid mile latency	The time it takes the Akamai platform to process a request. Usually, it is the time for a complete

Group	Data element	Description
		request and response cycle, but these values could be separated.
	Net origin latency	The time in milliseconds from when the last byte of the request leaves the edge server that is closest to the data center to when this edge server receives the first byte of the response from the data center. This value includes: • Time the origin takes to process the request before delivering the response • Network latency between an edge server and a data center This value shouldn't include: • Time to establish the connection with the origin ⓘ **Note:** If included, the origin connection time may or may not include the TCP and SSL/TLS establishment times or any possible failover and retry cycles that may have happened. • Network time or computing events that may have happened upstream in the Akamai transaction
Geo	**Area**	The area where the request originated.
	City	The city where the request originated.
	Country	The country where the request originated.
	Latitude	The latitude where the request originated.
	Longitude	The longitude where the request originated.
	Region	The region where the request originated. The region may be a state, province, or other large territory.
	Zip	The zip code the request was sent from.
Network data	**Bandwidth**	Specifies the bandwidth usage.
	Network	The network that originated the request.
	Network type	The type of network that originated the request.
	Proxy	The proxy or browser type.
	Throughput	The average throughput.

Group	Data element	Description
Cache data	**Cacheable**	Indicates whether the object was cacheable. `1` indicates that the server determined, based on response headers and metadata, that the object was cacheable. `0` indicates that it wasn't.
	Cache hierarchy	Categorizes the bytes served to the client by the forward server type that sent them. It provides data in the following order: `<peer_server>/<parent_server>/<origin_server>/<NetStorage>/<Akamai_origin>` where: • *<peer_server>* are the bytes served by an in-region peer edge server. • *<parent_server>* are the bytes served by a parent edge server. • *<origin_server>* are the bytes served by the origin server. • *<NetStorage>* are the bytes served from NetStorage. • *<Akamai_origin>* are the bytes served by any edge server that a request was forwarded to.
	Cache Hit	Indicates whether the requested object was served entirely from the cache memory. `1` indicates the edge server retrieved the entire object from the cache. `0` indicates that the server had to fetch some bytes of the object.
	Cache stats	Logs the bytes served entirely from the cache. It provides data in the following format: `<bytes_from_cache>/<total_bytes_to_client>` where: • *<bytes_from_cache>* are the bytes of the object or requested range served from the cache.

Group	Data element	Description
		• *<total_bytes_to_client>* are the bytes of the object or requested range sent to the client. **Note:** For regular objects, either none or all bytes of the object come from the cache.
	Cache status	Specifies whether a request was a cache hit or a cache miss and indicates the server type that provided the object. `0` indicates that the content was non-cacheable. `1` indicates that the object was served from a child edge server. `2` indicates that the object was served from an in-region peer edge server or a parent edge server. `3` indicates that the object was served from the origin server. `4` indicates that the response to the request had a code status other than: `200`, `203`, `301`, `302`, `410`. It also indicates that the object was served from the cache.
Waf data	**Anom scr**	A comma-delimited list of anomaly scores for the triggered rules. **Note:** This field's value is URL-encoded.
	Deny actions	The resulting actions of the deny rules triggered by the request as specified in the `Deny rules` field. `3` indicates that the rule denied the request. See *About rules* in the Cloud Monitor Help. **Note:** This field's value is URL-encoded.
	Deny data	Additional information about the risk group that triggered the deny action.

Group	Data element	Description
		Note: This field's value is URL and Base64-encoded to prevent control characters from impacting parsing.
	Deny msg	The messages reported by the deny rules triggered by the request. This is a semicolon-delimited list. **Note:** This field's value is URL-encoded. See *About rules* in the Cloud Monitor Help.
	Deny rules	Identifiers of all deny rules triggered by the request. This is a colon-delimited list. See *About rules* in the Cloud Monitor Help.
	Deny slrs	The locations in the request that triggered each deny rule. This is a semicolon-delimited list.
	P action	The resulting action for a slow POST attack, either `W` for warn, or `A` for deny (abort).
	Policy id	The identifier of the firewall policy applied to the request. See *Security policies* in the Cloud Security Help.
	P rate	The recorded rate in bytes per second of a slow POST attack.
	Risk groups	Risk groups whose rule thresholds have been triggered. This is a colon-delimited list. See *KONA WAF rules*.
	Risk tuples	Identifiers of the rules triggered within each risk group from the `Risk groups` field. Within a colon-delimited risk group, multiple rules are hyphen-delimited.
	Risk scores	Risk scores of each triggered rule from the `Risk tuples` field. Within a colon-delimited risk group, each rule's score is hyphen-delimited.
	Waf version	The version of a Web Application Firewall (WAF) data set. This is version 2.0. See *Update rule set* in the Cloud Monitor Help.
	Warn actions	The resulting actions of the warn rules triggered by the request as specified in the `Warn rules` field. This is a colon-delimited list.

Group	Data element	Description
		`2` indicates that the rule logged an alert. See *About rules* in the Cloud Monitor Help. **Note:** This field's value is URL-encoded.
	Warn data	The *user data* of the triggered rules from the `Warn rules` field. *User data* is a specific string within a selector that triggered the rule. This is a colon-delimited list. **Note:** This field's value is URL and Base64-encoded to prevent control characters from impacting parsing. See *About rules* in the Cloud Monitor Help.
	Warn msg	The messages reported by the warn rules triggered by the request. This is a semicolon-delimited list. **Note:** This field's value is URL-encoded. See *About rules* in the Cloud Monitor Help.
	Warn rules	Identifiers of the rules triggered by the request. This is a semicolon-delimited list. See *About rules* in the Cloud Monitor Help.
	Warn slrs	The selectors of the triggered rules from the `Warn rules` field. A *selector* is the location of the request or response that triggered the rule, such as the name of an HTTP header. This is a semicolon-delimited list.
	Warn tags	The tags of the triggered rules from the `Warn rules` field . Tags are used for classification and categorization. This is a semicolon-delimited See *KONA WAF rules*.

Aggregated metrics data set

Aggregated metrics data set parameters

Group	Metric	Description	Example
Edge response time	**Edge response time**	Specifies the latency observed for requests that results from: • a cache-hit at Akamai • a cache-miss at Akamai • a cache-hit at child or parent level • a cache-miss at child or parent level • non-cacheable requests	`"edgeResponseTime": 8.46`

HTTP status code	**2xx**
	3xx
	4xx
	5xx
		The count of requests that resulted in 2xx, 3xx, 4xx, and 5xx status codes.
		Note: Apart from aggregating status codes for each category, DataStream also returns `x_dist` member to show the distribution of specific status codes within these categories.
			`"2xx": 53,`
			`"2xx_dist": {`
			`"200": 53`
			`}`

Traffic volumes	**Requests per second**	Specifies the requests sent per second to the edge server.	`"bytesPerSecond": 29558.67`
	Bytes per second	Specifies the bytes per second received from the edge server.	`"requestsPerSecond": 0.88`
CDN offload	**Cache Hits**	Specifies the count of requests that were cache hits.	`"numCacheHit": 53`
	Cache misses	Specifies the count of requests that were cache misses.	`"numCacheMiss": 0`
	Offload rate	Specifies the offload rate over the period.	`"offloadRate": 100.0`

Group	Metric	Description	Example
Origin response time		The time in milliseconds from when the last byte of the request leaves the edge server that is closest to the data center to when this edge server receives the first byte of the response from the data center. This value includes: • Time the origin takes to process the request before delivering the response • Network latency between an edge server and a data center This value shouldn't include: • Time to establish the connection with the origin ⓘ **Note:** If included, the origin connection time may or may not include the TCP and SSL/TLS establishment times or any possible failover and retry cycles that may have happened. • Network time or computing events that may have happened upstream in the Akamai transaction	`"originResponseTime": 0`

Select a destination for your logs

Set your stream to push log data to one or more destinations or don't push it anywhere and make it available to your web services through the pull API. Depending on the type of logs that you set for you data stream to collect, you can also specify the sampling of data that you want to store or send to your destination or the aggregation time window for your metrics.

Before you begin

Determine whether you want to push log data to a destination or integrate your data stream with a third-party web service and pull log data with the DataStream pull API. See *Destination* in *Key concepts and terms* on page 6.

How to

1. On the **Delivery** tab, provide details for your destination. The fields to fill in differ depending on the destination type you select.
    - For a **Datadog** push destination, see *Stream logs to Datadog* on page 29.
    - For an **Amazon S3** push destination, see *Stream logs to Amazon S3* on page 29.
    - For a **Custom HTTPS** push destination, see *Stream logs to a custom HTTPS endpoint* on page 31.
    - For a **Sumo Logic** push destination, see *Stream logs to Sumo Logic* on page 31.
    - For a **Splunk** push destination, see *Stream logs to Splunk* on page 31.
    - For a **DataStream buffer** pull destination, see *Stream logs to a DataStream buffer* on page 32.
2. Depending on the type of logs you've chosen for you data stream when providing the basic details:
    - If you've chosen **Raw**, specify a **Sample rate**. This is a global sample rate for all properties in this data stream that defines a sample of data sent to the destination.
    - If you've chosen **Aggregated**, select your **Time frame** from the dropdown. This is the time window over which the data stream aggregates your log data and sends to the destination. You can set these values: 1, 5, 15, 30 minutes, or 1 hour.
3. Optional: To configure any other destination for your logs, click **Add destination**.

    You can configure only one DataStream buffer destination for a data stream.
4. Click **Next** to continue to the **Summary** tab.

Next steps

You're almost done. You can now review the information your provided and activate a data stream. See *Review and activate a data stream* on page 33.

Destinations

When you're adding or editing a data stream to push logs to a destination, you need to provide integration details to enable log streaming to this destination. Here, you'll find detailed information about each destination type available for DataStream.

Each destination has its own unique stipulations for receiving data streams with respect to authorization, authentication, and web service protocols. Also, each destination approaches data protection and storage differently. Destinations are expected to support secure transport, secure data at rest, and manage life cycle controls natively.

ⓘ **Note:** DataStream doesn't support storing logs in NetStorage.

Stream logs to Datadog

DataStream supports sending logs to Datadog. Datadog is a cloud-based monitoring and analytics solution that allows you to see inside applications within your stack and aggregate the results.

DataStream uploads logs to Datadog over HTTP(s) endpoints in an uncompressed file.

For security reasons, DataStream sends logs over TLS even if Datadog policies allow insecure requests.

Before you begin

- Register for a Datadog account. The place where you register your Datadog account, either in the United States (US) or the European Union (EU), affects commands and endpoints you use when configuring Datadog as destination in a data stream configuration. See *Datadog site*.
- Generate a Datadog API key dedicated to a data stream. See *Datadog API keys*.
- Gather static custom tags that you want to send with the log streams: tags, source, and service. See *Datadog logs over HTTP* and *Datadog tagging*.
- Identify the HTTPS endpoint in a hosting region, either US or EU. See *Datadog logs over HTTP*.

How to

1. In **Destination**, select **Datadog**.
2. In **Name**, enter a human readable description for the endpoint.
3. In **Endpoint**, enter the Datadog endpoint where you want to send and store logs.

   Example

   For example:

   ```
   http-intake.logs.datadoghq.com/v1/input
   http-intake.logs.datadoghq.eu/v1/input
   ```
4. Optional: In **Tags**, enter a comma-delimited list of tags that you use to filter and group your metrics in your Datadog account.
5. Optional: In **Source**, enter the source name from which logs originate associated with your Datadog account.

   The system sets Akamai as a default source of logs.
6. Optional: In **Service**, enter the name of the application or service generating the log events associated with you Datadog account.

   See *Services list in Datadog*.
7. In **API key**, enter the API key associated with your Datadog account.
8. Click **Validate&Save** to validate the connection to the destination and save the details you provided.

Stream logs to Amazon S3

DataStream supports sending logs to Amazon Simple Storage Service (Amazon S3). Amazon S3 is a static file storage that lets you organize your data and configure finely-tuned access controls to meet your specific business, organizational, and compliance requirements.

DataStream sends GZIP-compressed logs to Amazon S3.

For security reasons, DataStream sends logs over TLS even if Amazon S3 policies allow insecure requests.

Before you begin

- Create an Identity and Access Management (IAM) user in Amazon S3. See *Overview of access management: permissions and policies*.
- Create a dedicated storage bucket in an AWS region. See *Create storage buckets*.
- Grant the user or role that can access the bucket the appropriate permissions to the bucket contents, including `ListBucket`, `GetObject`, and `PutObject`.
- Make note of the access keys and client secret associated with your account. See *Understanding and getting your security credentials*.
- Set up and manage server side encryption (SSE) in the container's settings. See *Server side encryption for Amazon S3*.

How to

1. In **Destination**, select **S3**.
2. In **Name**, enter a human-readable description for the destination.
3. In **Bucket**, enter the name of the S3 bucket your created in the S3 account where you want store logs.
4. In **Folder path**, provide the path to the folder within the bucket where you want to store logs. If the folders don't exist in the bucket, Amazon creates them. You can also use `{}` to enter these time variables: year, month, day, and hour. For example, `logs` or `logs/{year}/diagnostics`.

   Note: Amazon treats objects that end with `/` as folders. For example, if you start your path with `/` like `/logs`, Amazon creates two folders in your bucket. The first one is named `/` and it contains the `logs` folder. See *Using folders in AWS*.

5. In **Access key ID**, enter the access key associated with the Amazon S3 bucket.
6. In **Secret access key**, enter the secret key associated with the Amazon S3 bucket.

   Tip: You can check your authentication details in the .csv file that you saved when creating your access key. If you didn’t download the .csv file, or if you lost it, you may need to delete the existing access key and add a new one. See *Managing access keys (console) in AWS*.

7. In **Region**, select the AWS region name where the bucket resides.
   See *Region names and codes in AWS*.
8. Click **Validate&Save** to validate the connection to the destination and save the details you provided. As part of this validation process, the system uses the provided access key identifier and secret and to save an `akamai_write_test_2147483647.txt` file in your S3 folder. You can only see this file if the validation process is successful, and you have write access to the Amazon S3 bucket and folder that you are trying to send logs to.

Stream logs to a custom HTTPS endpoint

DataStream supports sending logs to a secure HTTPS endpoint to allow on-premise software to receive and process logs.

For security reasons, DataStream sends logs over TLS even if the endpoint's policies allow insecure requests.

Before you begin

- Deploy a dedicated HTTPS endpoint that supports URL token authentication.
- Enable TLS transport for the endpoint to receive your data stream's logs.

How to

1. In **Destination**, select **HTTPS Custom End Point**.
2. In **Name**, enter a human-readable description for the destination
3. In **Endpoint URL**, enter the secure URL where you want to send and store your logs.
4. Click **Validate&Save** to validate the connection to the destination and save the details you provided.

Stream logs to Sumo Logic

DataStream supports sending log files to Sumo Logic to help you make data-driven decisions and reduce the time to investigate security and operational issues.

For security reasons, DataStream sends logs over TLS even if Sumo Logic policies allow insecure requests.

Before you begin

In Sumo Logic, configure an HTTP logs and metrics source and get the Sumo Logic URL endpoint to upload your data. See *Sumo Logic source configuration*.

How to

1. In **Destination**, select **Sumo Logic**.
2. In **Name**, enter a human-readable description for the destination.
3. In **Endpoint URL**, enter an HTTP source address where you want to send logs.
4. Click **Validate&Save** to validate the connection to the destination and save the details you provided.

Stream logs to Splunk

DataStream supports sending uncompressed logs to Splunk. It is an interface that lets you search, monitor, and analyze your data.

Before you begin

- To use Splunk as a destination for your logs, you need to set up an HTTP event collector instance (HEC), create a token, and enable it. Set up an HEC instance that matches the type of Splunk software you use. See *Set up and use HTTP Event Collector in Splunk Web*.
- Make note of the HEC token that you enabled.
- Make note of the URL for your event collector. The URL structure depends on the type of your Splunk instance. See *Send data to Event Collector*.

How to

1. In **Destination**, select **Splunk**.
2. In **Name**, enter a human-readable description for the destination.
3. In **Splunk cloud URL**, enter the URL where you want to send your logs.

   Example

   For example, `https://<splunk-host>:8088/services/collector/event/logs`
4. In **Access key**, enter the HEC token for your event collector.

Stream logs to a DataStream buffer

DataStream supports storing your stream's log data for up to 12 hours in a DataStream buffer. You can fetch your data with the DataStream pull API and ingest it into your preferred log analytics platform, such as Datadog or BiqQuery. This lets you set up real-time dashboards or alerts for constant, proactive mitigation of connectivity problems, service disruptions, and configuration-tuning complications.

You can enable several data sets for your platform to consume, such as response times from the edge, bytes transferred per second, and a breakdown of HTTP status codes. You can use these metrics to build sophisticated visualizations and alerts. For example, anomaly detection alerts that use past trends to reveal when request traffic deviates from normal levels.

You can also monitor the effectiveness of your caching setup with metrics on cache hits, cache misses, and the offload rate of your origin. Integrating your stream with a platform can help you troubleshoot issues behind the CDN by tracking the origin response time, or the latency in returning objects that Akamai requests from your origin infrastructure.

ⓘ **Note:** You can configure only one DataStream buffer for a stream.

How to

1. In **Destination**, select **DataStream buffer**.
2. In **Name**, enter a human-readable description for the destination.
3. Click **Save**.
   To start using the DataStream Pull API to fetch log data from a configured buffer destination, you need to wait around 10 minutes after activating the data stream on production.

Next steps

- Configure the DataStream pull API. See *DataStream pull API*.
- Learn how to integrate third-party solutions with data streams that store data in a DataStream buffer. See *Pull integrations* on page 38.

Review and activate a data stream

Review your data stream's details and activate it on the production network.

How to

1. On the **Summary** tab, review the accuracy of the data provided in each section.
2. Click **Save stream**.
   Activating a data stream takes about 15 minutes. After this time, your data stream is active on the production network.

> **Tip:** You can deactivate an entire data stream in the application or disable log collection for specific properties in Property Manager. See *Activate or deactivate a stream* on page 46 and *Activate or deactivate log collection for a property* on page 48.

Next steps

Your data stream configuration is almost ready to collect logs. To start receiving log data, add and enable the **DataStream** behavior in each property associated with this data stream. See *Enable DataStream in a property* on page 33.

Enable DataStream in a property

To start collecting logs for properties in your data stream, you need to enable a DataStream behavior in each property that is part of the stream. You can't receive logs from DataStream-disabled properties even if they're part of active data streams.

When enabling the DataStream behavior, you can configure it to:

- Collect logs from all data streams that monitor this property.
- Collect logs from specific data streams that monitor this property.

Apart from enabling the behavior in the default rule to apply it to all requests, you can enable it in a custom rule to capture data for specific hostnames, paths, status codes, and other requests matching your criteria.

Before you begin

- Check if DataStream has been added to your contract, and that a content provider (CP) code is available for use with DataStream. For assistance, contact your account representative.
- Check the identifiers of the data streams that monitor this property. You'll need this data to if you want to receive logs only form these data streams. See *Viewing options* on page 47.

How to

1. Use the top-right menu in the header to select the appropriate Control Center account (one with access to the applicable product).
2. Access Property Manager configurations associated with the selected Control Center account. Go to  > **CDN** > **Properties** (or just enter **Properties** in the search box).

The Property Groups page opens.

3. Click the **Property Name** link for your property.
4. On the **Property Details** page, click the **Version** of your configuration that you want to access in **Manage Versions and Activations**.
   The **Property Manager Editor** appears.
5. Select or create the rule where you want to enable the **DataStream** behavior:

   - To apply the data stream to all hostnames in the property, add and enable the DataStream behavior in the **Default Rule**.
   - To apply the data stream to specific hostnames in the property or specific requests, add and enable the DataStream behavior in a custom rule. See *Configure DataStream in a custom rule* on page 49.

6. In the rule, click **Add Behavior**, and select **DataStream**.
7. Click **Insert Behavior**.
   The DataStream behavior appears in your rule.
8. In **Stream version**, select **DataStream 1**.
9. In the **DataStream 1** section:

   a. Set the **Enable** switch to **On**.
   This enables log collection from all active data streams that this property is part of.

   b. In **DataStream ID(s)**, enter identifiers of the active data streams associated with the property and separate them with a hyphen. You'll only receive logs from these data streams.

   Example

   For example, `51-75-145` means that you'll receive data from the data streams with the following identifiers: 51, 75, 145.

   Tip: You can find the identifiers of all data streams that monitor this property by entering this property name in the search box of the DataStream application.

   If you leave this field empty, the DataStream behavior collects data from all active data streams associated with the property.

10. Click **Save**.
11. Enable logging custom data set parameters in the **Log Request Details** behavior. See *Enable logging custom parameters* on page 51.
    Do this step only if you selected these parameters in your data set:

    - User-Agent
    - Accept-Language
    - Referer
    - Host

- Cookie
- Custom field

12. On the **Activate** tab, click **Activate v# on Production**.
    You can only activate a property with the DataStream behavior on the production network.

What you should see

After the DataStream-enabled property is active on the production network, you're ready to monitor and manage your data.

Caching

This section can help you understand when downloads are cache hits and how to determine the number of bytes served from cache.

Objects served from cache

This section assumes that the object is `cacheable`. For non-cacheable objects, the `cacheable` and `cache hit` parameters return 0.

In general, if an edge server doesn't got forward with the request and serves a 2xx response, it's safe to assume that the object was served entirely from the cache. When partial object caching is enabled for the object, you can consider the download a cache hit only if an edge server doesn't go forward to fetch any of the fragments needed to satisfy the request.

Here are the use cases when the download can be considered a hit even though the edge server goes forward:

- Successful If-Modified-Since (IMS) validation for objects without partial object caching
- Successful partial object caching master/fragment check

If the edge server goes forward to fetch the object, and the forward server returns a 304 status code (not modified), the edge server still serves the object from cache. You can consider the download a cache hit. However, if the forward server returns a 20x status code, the edge server evicts the content in cache and replaces it with the updated object. You can consider the download a cache miss. Note that the updates on the `If-Modified-Since` header are considered a cache hit as the content body is served from cache.

When a POC master entry needs to be checked, the edge server forwards a 0-0 range request to the origin. It checks again the object size based on the `Content-Length` value included in the `Content-Range` header. If the check fails, the edge server fetches the updated object. This is a cache miss.

When a POC fragment needs to be checked, the edge server forwards a one-byte range request that starts from the fragment's offset. It checks again the fragments size based on the `Content-Length` value included in the `Content-Range` header. If the check fails, the edge server fetches the complete fragment. This is a cache miss.

If all POC check requests are successful, the download is a cache hit even though the edge server forwarded the check requests.

Bytes served from cache

This section assumes that the object is `cacheable`. For non-cacheable objects, the `cacheable` and `cache hit` parameters return 0. Also, the `bytes_from_cache` value from the `cache stats` returns 0.

For objects without partial object (POC) caching enabled, either none or the entire object comes from cache. However, for POC-enabled objects, some fragments may already be in cache, and the edge server needs to go forward to fetch the rest of the needed fragments. At the end of the download, the `cache stats` parameter returns the number of cached fragments multiplied by the requested fragment size in its `bytes_from_cache` value and the total number of fragments multiplied by the requested fragment size in its `total_bytes_to_client` value.

For example, if two fragments are in cache for an object split into 10 fragments, each consisting of 2000 bytes, the `cache stats` parameters returns `4000/20000`.

Pull integrations

You can integrate your data stream with third-party services for storage and analysis.

These are examples of platforms that you can use:

- Datadog
- BigQuery

Datadog integration

DataStream provides out-of-the-box integration with Datadog to measure the health of your application and performance in real-time.

Datadog is a monitoring service for cloud-scale applications, providing monitoring of servers, databases, tools, and services through a SaaS-based data analytics platform. See *Integrate Akamai with Datadog to monitor CDN performance*.

> **Note:** You can integrate only an aggregated metrics stream that stores log data in a DataStream buffer with Datadog. See *Key concepts and terms* on page 6.

How to

1. Get started with DataStream.

   Configure an aggregated metrics stream that allows only for pulling data. See *Add a data stream* on page 10.

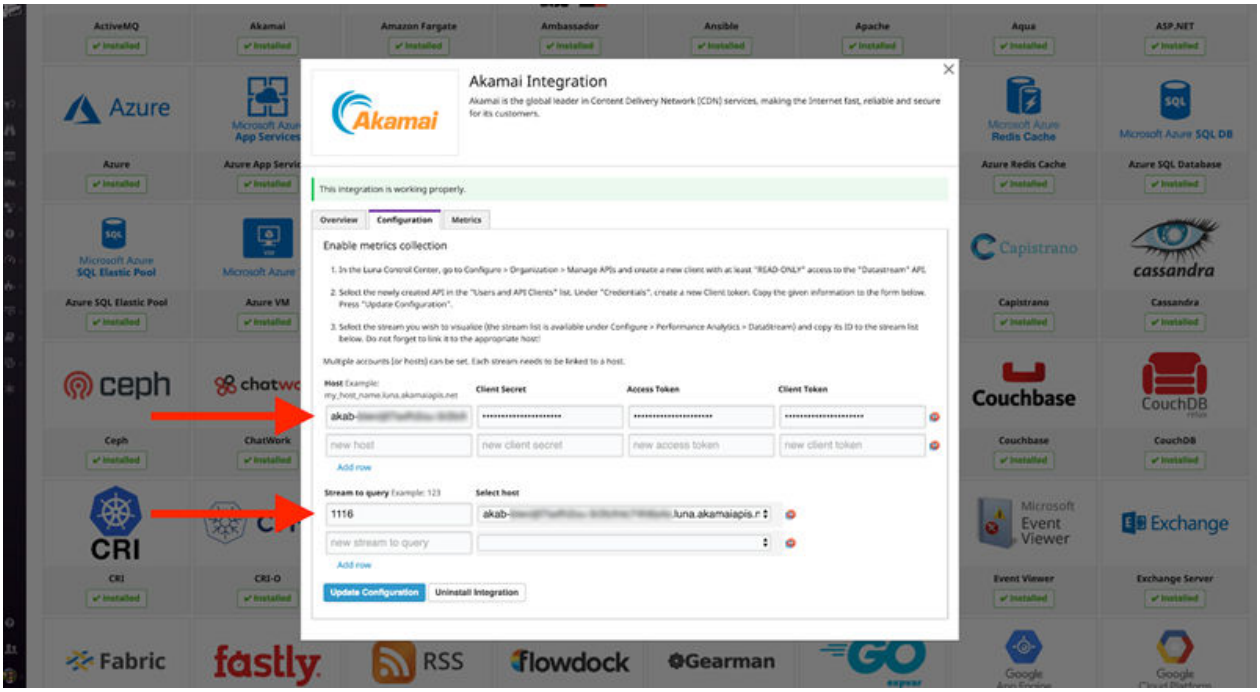
2. Set up an API client for the DataStream Pull API.

   To integrate with Datadog, you need an API client with at least read-only access to the DataStream Pull API. You can create an API client in Control Center. See *Get started with APIs*.

3. Enable DataStream integration in Datadog.

   In your Datadog account, open the **Akamai integration** tile from *the list of available integrations*. Paste in the previously created credentials: host, client secret, access token, and client token. Next, add the Stream ID of your aggregated metrics stream. Select your host from the dropdown and click **Update Configuration** to complete the integration.

Example



4. Visualize your data in Datadog.

   Once you've enabled the Akamai integration in Datadog, your metrics start flowing into your Akamai dashboard. This template dashboard includes several high-level metrics on caching, request rate, and response times. You can also easily clone this dashboard to customize it and suit your needs.

BigQuery integration

Google Cloud Platform (GCP) BigQuery is a columnar database tool that provides data analysis without having to take care of the underlying infrastructure. It also lets you visualize your data with an integrated tool called Data Monitor. You can now integrate DataStream with BigQuery to find meaningful insights, use familiar SQL, and take advantage of a pay-as-you-go model.

> ⓘ **Note:** You can integrate raw logs and aggregated metrics streams with BigQuery. This example shows how to integrate a raw logs stream that pushes data to BigQuery. See *Key concepts and terms* on page 6.

How to

1. Get started with DataStream.

   In DataStream, configure a raw logs stream and select your data sets. For example, you can select **Request Header Data** to choose the headers that you want to receive when calling the API. You may want to receive headers such as `Authorization`, `Range`, `Accept-Encoding`, and many others.

   You can also choose a sample rate. For details, see *Add a data stream* on page 10.

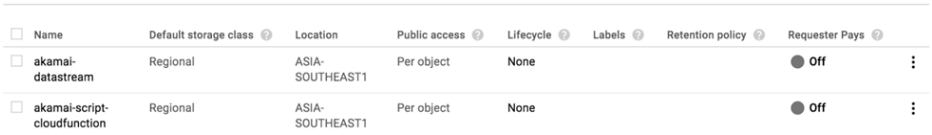
2. Set up an API client for the DataStream Pull API.

   To integrate with BigQuery, you need an API client with at least read-only access to the DataStream Pull API. You can create an API client in Control Center. For details, see *Get Started with APIs*.

3. Set up a GCP account.

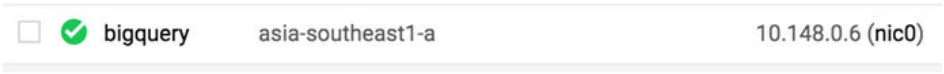
   Open a new project in Google Cloud Platform and start creating the following products:

   Cloud Storage — Set up two buckets. One to store the logs, and the other to store a cloud function script. For details, see *Creating buckets in Google Cloud Storage*.

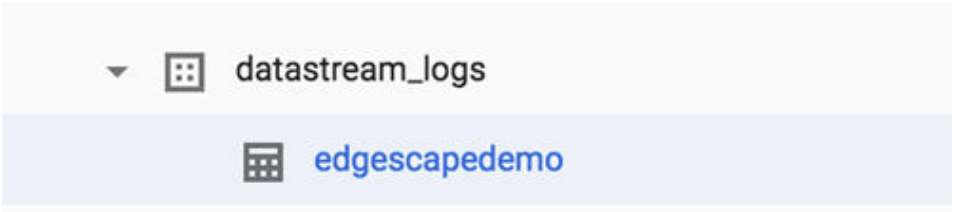


Name	Default storage class	Location	Public access	Lifecycle	Labels	Retention policy	Requester Pays	
akamai-datastream	Regional	ASIA-SOUTHEAST1	Per object	None			Off	⋮
akamai-script-cloudfunction	Regional	ASIA-SOUTHEAST1	Per object	None			Off	⋮

   Compute Engine — Set up one compute workload to call the DataStream Pull API and copy it to cloud storage.



   BigQuery Database — Create a BigQuery database for your logs. You'll add a table later.



   Once you are done, go to **API services** in your Google Cloud Platform account and enable the following APIs cloud functions: BigQuery and Cloud Storage.

4. Integrate DataStream with BigQuery.

   Compute Engine setup — SSH into the compute engine that you previously set up. Then, install the gcloud API. For details, see *Install Google Cloud SDK*.

   Install the Akamai APIs and clients. Copy the previously created credentials and paste them in the `.edgerc` file. For details, see *Gett started with APIs*.

   Next, grant the compute engine access to the GCP resources, such as storage, BigQuery, and the cloud function. For details, see *Granting, changing, and revoking access to resources in Google Cloud Storage*.

   BigQuery table setup — First, you need to get the DataStream Pull API's schema. See *DataStream Pll API schema*.

   Next, prepare a BigQuery schema that matches the DataStream schema. The BigQuery schema looks exactly like the one here.

Schema


```
[
 {
 "name": "data",
 "type": "RECORD",
 "mode": "REPEATED",
 "fields": [
   {
    "name": "cp",
    "type": "string"
   },
   {
    "name": "guid",
    "type": "string"
   },
   {
    "name": "id",
    "type": "integer"
   },
   {
    "name": "reqid",
    "type": "string"
```


Note: The schema has a lot of nested records.

Then, use the prepared schema to create a table in BiqQuery. This command uses the schema called `schema.json` to create a table called `edgescapedemo`:

```
bq mk --table
akamai-206503:datastream_logs.edgescapedemo ./schema.json
```

Cloud Function setup

You also need to write a cloud function. The cloud function is a serverless computing product. For details, see *Cloud function in Google Cloud*.

It can act on triggers. Here, the trigger that we use is `google.storage.object.finalize`. As soon as something is uploaded to cloud storage, the trigger will fire. For details, see *Storage triggers in Google Cloud*.

Once you've prepared the cloud function, you can deploy it with this command:

```
gcloud beta functions deploy datastream-cloud-function --
trigger-resource=akamai-datastream --trigger-event
google.storage.object.finalize --source=. --stage-
bucket=gs://akamai-script-cloudfunction --entry-
point=jsonLoad
```

5. Call the DataStream API.

Now all the pieces are in place, you can start your API calls script and push the DataStream JSON response file to cloud storage. Once the file is uploaded to cloud storage, the finalize trigger activates the cloud function and stores the file or your data in a BigQuery table.

Here is the flow:

- Make an API call for the DataStream API from the compute engine. This can be a Cron job:

```
http --auth-type edgegrid -a datastream-pull-api: ":/datastream-pull-api/v1/streams/851/raw-logs?start=2018-10-30T06:30:00Z&end=2019-10-23T06:40:00Z&page=0&size=100"
```

- Push the output to the bucket for DataStream logs:

```
gsutil cp output.json gs://akamai-datastream
```

  As soon as it's in the bucket, it'll activate the cloud function. Looking at the cloud function logs, you can verify if it has successfully completed. You can return the logs with this command:

```
gcloud beta functions logs read datastream-cloud-function
```

- Open the BigQuery interface and query the table. You'll see something similar:

Example

Cloud function logs

LEVEL	NAME	EXECUTION_ID	TIME_UTC	LOG
D	datastream-cloud-function	282735249746417	2018-10-30 03:03:27.801	Function execution started
I	datastream-cloud-function	282735249746417	2018-10-30 03:03:28.607	Starting job for output2.json
I	datastream-cloud-function	282735249746417	2018-10-30 03:03:32.395	Job complete for output2.json
D	datastream-cloud-function	282735249746417	2018-10-30 03:03:32.399	Function execution took 4599 ms, finished with status: 'ok'

BigQuery example use

In this example, consider a customer who uses DataStream to ingest logs every five minutes. The customer has received performance complaints over the past few minutes or has some side statistics showing an increase in load times, such as page loads. Here, the customer can quickly make a query to see the load times for all objects or scripts on his page.

You can use BigQuery to get file types together with their download times. This example SQL query returns `ts` and `m3u8` file types:

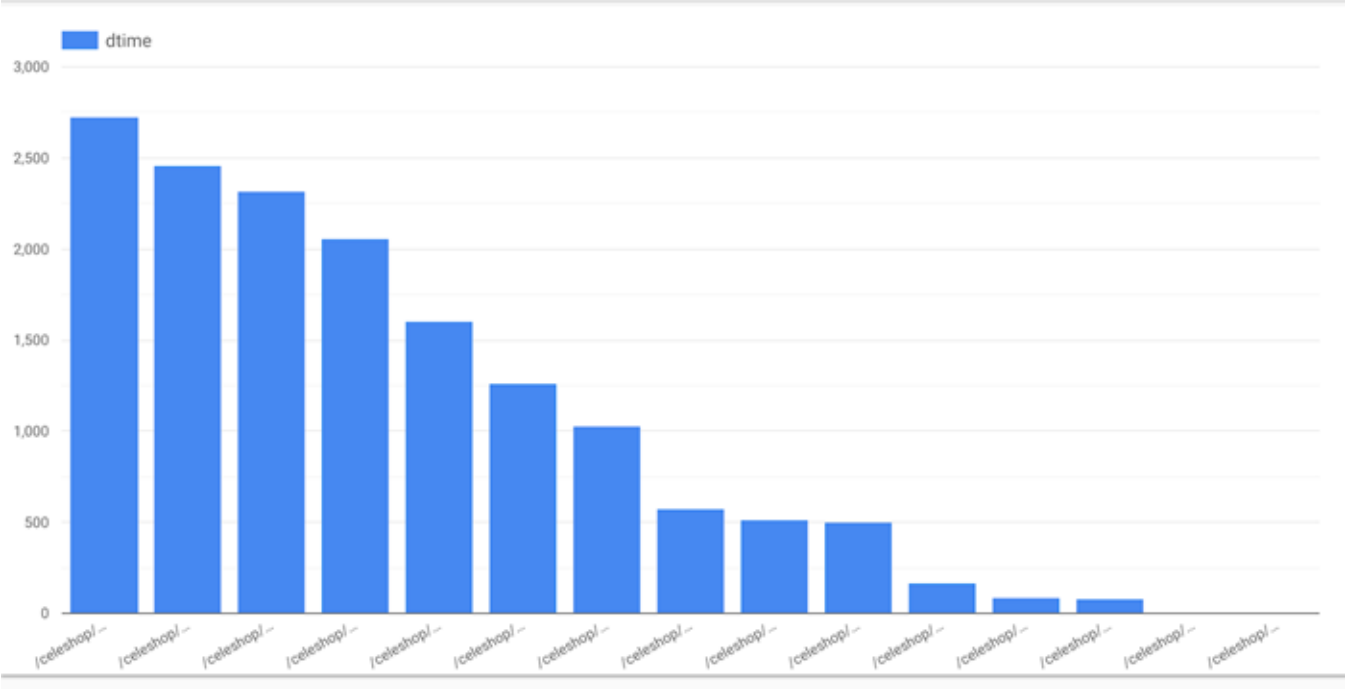
```
select d.message.reqPath, CAST(d.netPerf.downloadTime AS INT64) * 1 As dtime
FROM `akamai-206503.datastream_logs.edgescapedemo` ,
UNNEST(data) as d
where d.message.reqPath LIKE "%.ts" or d.message.reqPath LIKE "%.m3u8"
order by dTime desc
```

Here is the result:

Row	reqPath	dtime
1	/celeshop/360p_007.ts	2486
2	/celeshop/480p_004.ts	2458
3	/celeshop/360p_004.ts	2090
4	/celeshop/480p_003.ts	2059
5	/celeshop/720p.m3u8	1601
6	/celeshop/480p_006.ts	1265

You easily point out the files that take longer to download. Then, you can investigate further and make specific queries about the title, allowing the customer to identify the root cause of the problem.

What's more, Google Data Studio is an integrated feature, making it easy to visualize any query or table in a dashboard or a report. You can use it to convert the table into a graph.



Aggregation

Using an aggregate stream, you can also find out if the numbers of errors have increased. Aggregate data streams retrieve real-time of 4xx and 5xx HTTP error occurrences.

Here is the DataStream Pull API call:

```
http --auth-type edgegrid -a datastream-pull-api: ":/datastream-pull-api/v1/
streams/1201/aggregate-logs?
start=2019-02-15T09:19:37Z&end=2019-02-17T10:40:00Z&aggregateMetric=2xx%2C3xx
%2C4xx%2C5xx&page=0&size=100"
```

You can then ingest the return JSON file into BigQuery and visualize the errors in time.

DataStream operations

The DataStream application provides a control panel that lets you quickly overview and manage existing streams and destinations. From this panel on the landing page, you can globally activate and deactivate multiple streams across all associated properties and edit almost all parts of their configurations.

Version management

Every time you edit a stream, you create a new version. This lets you quickly adapt your existing streams to start collecting logs for different properties, modify data set parameters the streams monitor, or change destinations where they send logs. You can keep track of changes to your stream in the version history panel accessible from the DataStream control panel.

Each version that you edit becomes a default version for a data stream with the same activation status as its base version. For example, editing version 1 of an active stream creates version 2. Once saved, version 2 becomes active on the production network. Conversely, by editing version 1 of an inactive stream, you create and save an inactive version 2 of this data stream.

The DataStream control panel lets you view and compare all configuration versions within a stream, but you can only manage the activation status of the latest version. You also can't revert a stream to any previous version.

On the DataStream main page, you can access the version history panel for a stream by selecting **Actions** > **History** next to the relevant stream.

Edit a data stream

The DataStream control panel lets you edit almost every part of your stream configurations. For example, you can update the contract that you created the stream for, the name of you stream, the properties that you want to monitor, the parameters it logs, and the destinations to send log files.

By editing an active data stream, you create and activate a version that becomes the default version used by this stream. Once you publish the edited version, you can't revert to any previous version. See *Version management* on page 45.

How to

1. Log in to Control Center using a **User ID** and **Password** that have been configured for access to DataStream.
2. Go to ☰ > **COMMON SERVICES** > **DataStream**.
3. On the **DataStream** page, find the data stream that you want to edit and select **Actions** > **Edit**.
   See *Viewing options* on page 47.

   Tip: Pay attention to the activation status of the data stream that you want to edit. When editing, DataStream creates a version with the same activation status as the edited version. Editing and saving an active stream starts activating it on the production network.

   The **Edit stream** wizard appears.

4. In **Configuration**, you can do the following:

    a. Edit the **Contract ID** where you initially created the stream.

    b. In **Include properties**, edit the properties that the stream monitors.

    You can select up to 15 properties. All properties that are part of streams that are changing their statuses, either being activated or deactivated, are grayed out, and you can't select them.

    c. Click **Next** to continue to the **Data Sets** tab.

5. In **Data sets**, edit the parameters that the stream collects.

    See *Select data set parameters for a data stream* on page 11.

6. In **Delivery**, edit the destination where the stream sends logs and the delivery conditions for your log files.

    See *Select a destination for your logs* on page 27.

7. In **Summary**, review your changes, decide whether to publish the edited stream, and click **Save stream**.

    It takes about 15 minutes to activate a stream on the production network.

Next steps

If you saved an inactive version of a stream, you may want to activate it the control panel or manage log collection for specific properties in Property Manager. See *Activate or deactivate a stream* on page 46 and *Activate or deactivate log collection for a property* on page 48.

Activate or deactivate a stream

You can control receiving log data by activating and deactivating your streams in the DataStream application. For example, you may need to deactivate your stream when you're implementing downstream changes, planning a roll-out, or witnessing an incident. You may also need to quickly activate a stream that you saved in an inactive state to start log deliveries.

Having all your streams at hand, you can easily control the flow of data without creating new versions of the property configuration to implement your changes. Changes you introduce in the DataStream application take about an hour to take effect across the Akamai platform.

How to

1. Log in to Control Center using a User ID and Password that have access to DataStream.
2. Go to > **COMMON SERVICES** > **DataStream**.
3. On the **DataStream** page, find the stream that you want to manage, click  from the **Action** column, and click **Activate** or **Deactivate** to turn it on or off.

    indicates that the version is inactive on the production network.

    indicates that the version is active on the production network.

    It may take up to 90 minutes for the changes to take effect on the Akamai platform.

Note: By managing the stream's activation status, you activate or deactivate collecting logs for all active properties associated with this stream.

Viewing options

The DataStream control panel lets you quickly search for streams and check their details.

Checking a stream's details is especially useful before you enable the DataStream behavior in your property configuration. You can enter the name or identifier of your stream to check:

- If the stream you want to receive logs from is active. You can't receive logs from inactive streams.
- The properties monitored by the stream. This lets you know which properties the stream can collect logs for and where you can enable the DataStream behavior.

You can also enter the name of your property to check:

- Details of all streams monitoring this property. It's useful when you want to customize logging for a property and need to provide names or identifiers of the associated streams to collect your logs.

The control panel lists all streams created within the context of your contract and group. It provides the following details about each stream:

- **Name**. The name you previously provided for the stream.
- **ID**. The unique identifier for the stream that the system assigns automatically.
- **Properties**. The names of properties that the stream is associated with.
- **Type**. The version of DataStream that you used to create this stream.
- **Version**. The number of the latest version for the stream.
- **Status**. Shows the current status of the stream. A stream can be **Inactive**, **Activating**, **Activated**, **Deactivating**, **Deactivated**, or **Failed**.
- **Last modified**. Shows when you last made an action on this stream in the YYYY-MM-DD format.
- **Actions**. Lists the operations you can perform on a stream. Depending on the status of the stream, you can **Retry**, **View**, **Edit**, **Activate**, **Deactivate**, or check stream version **History**.

You can use the search box to filter the list of streams by:

- Name of the stream. Enter the name of the stream to view its details.
- Identifier of the stream. Enter the identifier of the stream to view its details.
- Property name. Enter the name of a property to view details of all streams associated with this property.

Tip: When configuring the DataStream behavior in a property, you may need to look up the names or identifiers of the streams that this property is part of. This way you know which streams can collect and send logs for this property.

Advanced control over a stream

You may want to enhance control over your log contents and delivery. In this section, you will learn how to edit your property configuration to enable logging custom content, include or exclude log lines over match criteria, and override log availability specified in the DataStream application. Additionally, you can set alarms to notify you when log files fail to upload to a destination.

Note: To apply changes to an active property, you need to create a version of this property's configuration in Property Manager. Also, all changes you introduce in Property Manager come first before the global setting configured for your stream in the DataStream application.

Activate or deactivate log collection for a property

To better control log data that your DataStream collects for individual properties, you can update the DataStream behavior in a property configuration.

You can configure the behavior to:

- Disable log collection for a property.
- Collect logs only from specific data streams.

To do any of that, you need to create a version of your property configuration in Property Manager and edit the existing DataStream configuration.

It's a quick way of overriding the global settings configured in the DataStream application and excluding a property from being part of its associated data streams. Changes you introduce in your property configuration take about 15 minutes to take effect across the Akamai platform. For example, you can quickly disable log collection for one property in a data stream on finding a security incident but keep monitoring other properties associated with this data stream.

Before you begin

Check the identifiers of the data streams that monitor this property. You'll need this data to if you want to receive logs only form these data streams. See *Viewing options* on page 47.

How to

1. Use the top-right menu in the header to select the appropriate Control Center account (one with access to the applicable product).
2. Access Property Manager configurations associated with the selected Control Center account. Go to > **CDN** > **Properties** (or just enter **Properties** in the search box).
   The Property Groups page opens.
3. Click the **Property Name** link for your property.
4. On the **Property Details** page, click the **Version** of your configuration that you want to access in **Manage Versions and Activations**.
   The **Property Manager Editor** appears.

5. If your property is already active, click **Edit New Version** to introduce your changes.
6. Find the **DataStream** behavior in the **Default Rule** or custom rule and introduce your changes.
7. In the **DataStream** section, do the following:

    a. Set the **Enable** switch to enable or disable log collection for this property.

    b. Optional: In **DataStream ID(s)**, select the identifiers of the associated data streams that you want to log this property's transactions. These are the only data streams that will collect logs for this property.

    > **Note:** If you leave this field empty, the DataStream behavior collects data from all active streams associated with the property.

8. On the **Activate** tab, click **Activate v# on Production**.

What you should see

Once the property version is active on the production network, your data streams take about 15 minutes to adapt to the changes.

Next steps

If you deployed any of the data streams monitoring this property in an inactive state, remember to activate them in the DataStream application. See *Activate or deactivate a stream* on page 46.

Configure DataStream in a custom rule

You can create any number of custom rules in your property configuration to associate match criteria with behaviors. You can configure a DataStream behavior in a custom rule to define the hostnames within your property that you want DataStream to collect data for. Also, you can specify match criteria for incoming requests to decide when to apply the DataStream behavior. By combining match criteria with the behavior's stream selection, DataStream ensures flexibility in collecting data based on specific requests and streams.

For more information, see *Rules and matches in Property Manager*.

Before you begin

Check if the property is associated with any active data streams and make note of their identifiers or names. You'll need this data when specifying the data streams that you want to collect logs for this property. See *Viewing options* on page 47.

> **Caution:** If you associate match criteria with the **CP code**, **Status code** or the **WAF variable** behavior, DataStream may stop collecting and delivering log files. Before adding these behaviors, enable the **DataStream** behavior in the default rule, setting the **Stream ID** value to `1`. Use your actual **Stream ID** in your desired match configuration.

How to

1. On the **Property Manager Editor** page, click **Add Rule** in **Property Configuration Settings**.
2. If your property is already active, click **Edit New Version** to introduce your changes.
3. In **Add New Rule**, from **Available Rules (by Category)**, select **Blank Rule**.

4. Select **Blank Rule Template** and click **Insert Rule**.
   A **New Rule** entry appears in the rules column.
5. In **Criteria**, add match criteria and choose one or more property hostnames to apply the DataStream behavior to.
   For a full list of matches, see *Available match criteria in Property Manager*.
6. Click **Add Behavior**, and select **DataStream**.
   Depending on your configuration, you may only need to take only some of these steps:
7. Configure the behavior.
   See step 7 in *Activate or deactivate log collection for a property* on page 48.

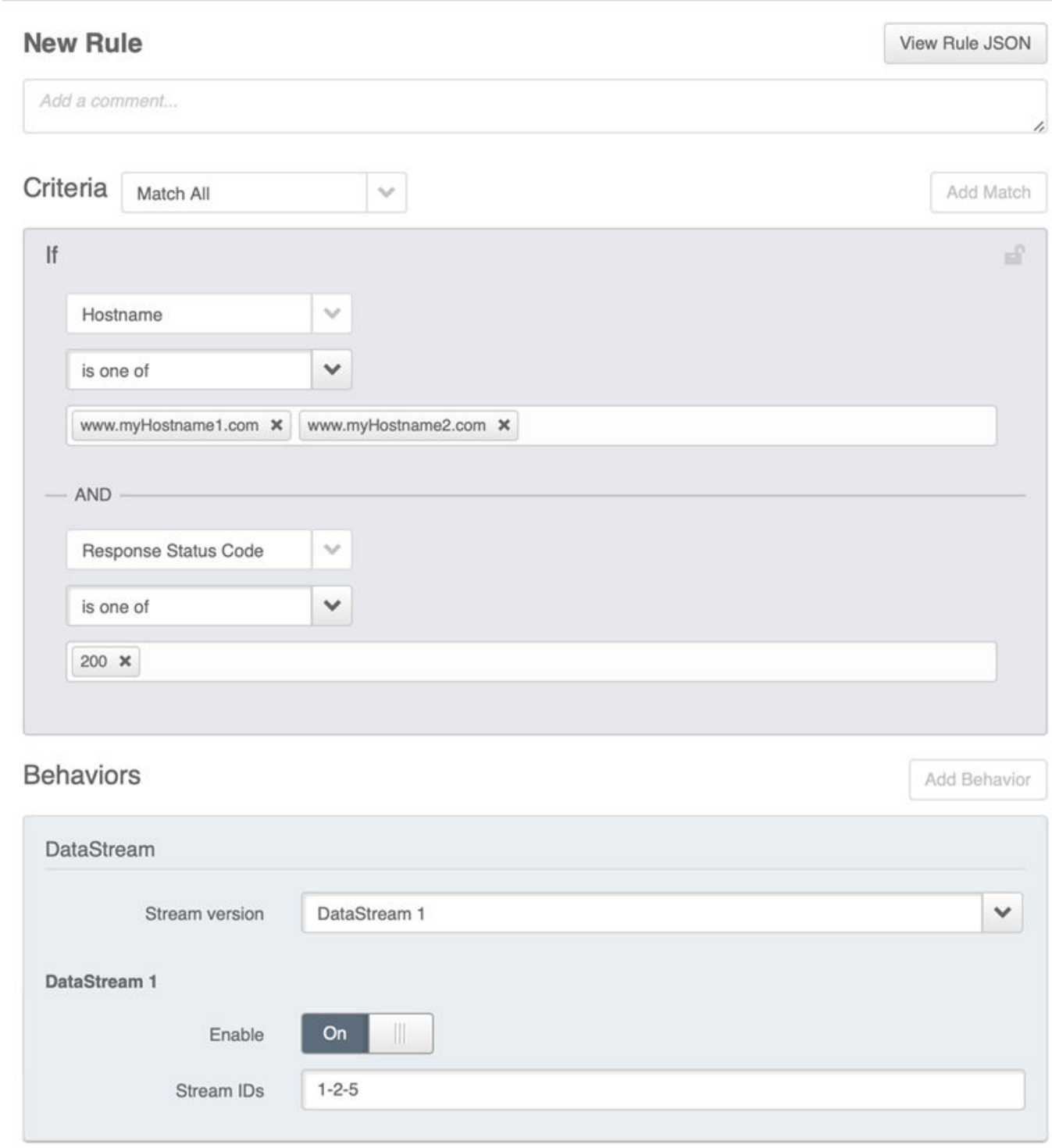
What you should see

Once the property version is active in the production environment, your data streams take about 15 minutes to adapt to the changes.

Next steps

If you deployed any of the data streams monitoring this property in an inactive state, remember to activate them in the DataStream application. See *Activate or deactivate a stream* on page 46.

Here, you've specified only two hostnames from your property configuration that DataStream collects logs for. Also, you'll receive data sets from three data streams monitoring only the requests with an `HTTP 200` response status code.



Enable logging custom parameters

You can use the Log Request Details behavior in your property configuration to include specific content in log lines generated by edge servers.

In this behavior, you can control:

- logging the value you set for the custom log field
- what cookie information is logged
- which of the following headers are part of your logs:

  User-Agent
  Accept-Language
  Cookie
  Referer
  X-Forwarded-For

Before you begin

Deactivate a stream associated with the property configuration that you want to update. See *Activate or deactivate a stream* on page 46.

How to

1. Use the top-right menu in the header to select the appropriate Control Center account (one with access to the applicable product).
2. Access Property Manager configurations associated with the selected Control Center account. Go to > **CDN** > **Properties** (or just enter **Properties** in the search box).
   The Property Groups page opens.
3. Click the name of your property.
4. On the **Property Details** page, click the version of your configuration that you want to access.
   The **Property Manager Editor** appears.
5. If your property is already active, click **Edit New Version** to introduce your changes.
6. Select or create the rule where you want to enable the **Logs Request Details** behavior:
   - To apply the behavior to all requests, add it in the **Default Rule**.
   - To apply the behavior to specific hostnames in the property or specific requests, add it in a custom rule.
7. In the rule, add or edit the **Log Request Details** behavior.
8. Enable the headers and specify the custom field that you want include in your streams.
   a. **Log Host Header**. Specifies the domain name of the server for virtual hosting, and optionally the TCP port number on which the server is listening.
   b. **Log Referer Header**. Contains the address of the previous web page from which a link to the currently requested page was followed. The **Referer** header allows servers to identify from where people are visiting.
   c. **Log User-Agent Header**. Contains a distinctive string that allows the network protocol peers to identify the application type, operating system, software vendor, or software version of the requesting client.
   d. **Log Accept-Language Header**. Advertises which languages the client is able to understand, and which locale variant is preferred.

e. In **Cookie Mode**, choose which cookies you want to log.

- Select **Don't log any cookies** to disable logging any cookies in your stream.
- Select **Log all cookies** to enable logging all cookies in your stream
- Select **Log some cookies** to specify which cookies to log in your stream. This option reveals the **Cookies** field where you can specify the cookie format that you can capture, either **sessionid** or **user_local**. You can also click the field and manually input the name of an applicable cookie format you want to use.

a. Switch **Include Custom Log Field** to **On**. This option reveals the **Custom Log Field** where you can specify an additional data field to append to each log line.

This field supports variables. See *Variable Support*. Note that the system shortens the value from the **Custom Log Field** to no more than 40 bytes of data.

Next steps

Once you've made your changes, activate your property in the production environment.

Informational and error messages

The following tables provide all informational and error messages that you may encounter when using DataStream 1.

Informational messages

Message	Description
Sent for activation stream: *stream ID*. Note: Other streams sharing properties with stream *stream ID* would be uneditable until the action is complete.	Shown when you've pushed a stream for activation. It also informs you that the action has locked the property that this stream shares with other streams, and that you can't edit this property until the action is complete.
Sent for deactivation stream: *stream ID*. Note: Other streams sharing properties with stream *stream ID* would be uneditable until the action is complete.	Shown when you've pushed a stream for deactivation. It also informs you that the action has locked the property that this stream shares with other streams, and that you can't edit this property until the action is complete.
Stream saved! Note: Other streams sharing properties with this stream would be uneditable until the action is complete.	Shown when you've successfully created or updated a stream. It also informs you that the action has locked the property that this stream shares with other streams, and that you can't edit this property until the action is complete.
Successfully deleted the stream: *stream ID*.	Shown when you've successfully deleted a stream.

Error Messages

Message	Description
Access denied. If you think you are authorized for this action, please contact support.	Shown when you don't have access to the DataStream application, or if you've performed an unauthorized action on a stream.
The stream has changed. Please refresh the page to check the latest version.	Shown when you're performing an action on a stream that other users have changed in the meantime.
An aggregated stream already exists for this account.	Shown when you already have an aggregated stream in your account, but you're creating another one through the DataStream API. ⓘ **Note:** This error only appears when you've performed the action through the DataStream API.

Message	Description
One or more properties in this stream are already in progress. Please try the action after some time. In case of persistent failure, please contact support.	Shown when you're trying to activate, deactivate, or update a stream in which properties are already undergoing any of these operations. ⓘ **Note:** This error only appears when you've performed any of the actions through the DataStream API.
Failed to save secrets. Please try again after some time. In case of persistent failure, please contact support.	Shown when your call to save connector secrets has failed. This means that your stream hasn't been properly saved.
No properties found in this stream. Please assign properties to stream and retry.	Shown when you're trying to activate or deactivate a stream that has no defined properties.
Failed to update stream: *stream ID*. `A link to retry.`	Shown when a stream update has failed. It also provides a retry link.
Failed to activate stream: *stream ID*. `A link to retry.`	Shown when a stream activation has failed. It also provides a retry link.
Failed to deactivate stream: *stream ID*. `A link to retry.`	Shown when a stream deactivation has failed. It also provides a retry link.
Failed to save destination(s) for stream: *stream ID*. Please contact support.	Shown when your secrets for connectors haven't been propagated across the Akamai network. This means that your data may fail to flow to the specified endpoint.
Something went wrong. Please try the action again. In case of persistent failure, please contact support.	Shown when an exceptional error in the ConfigAPI has occurred. For example, the service has been down and returns a 5xx error.

FAQ

The following are frequently asked questions (FAQs) about DataStream 2.

DataStream FAQs

Question	Answer
What is DataStream's output data format?	The output log file format for DataStream is Structured (CSV) or JSON. For sample schemas and data sets, see *DataStream 2 API*.
How many endpoints does DataStream send logs from?	DataStream sends logs to customer destinations from multiple source endpoints. Typically, the number of source endpoints range from 10s to 100s and can change without prior notice.
What happens if log files fail to upload to a destination?	Streaming logs to third-party destinations may temporarily fail due to issues such as latency or connection problems. To prevent data loss during temporary upload failures, DataStream retries data upload three times. After three failed upload retries, the data will not be saved. You can configure the *Alerts* feature to get notified about upload failures.
Should I use log data for billing?	We recommend using raw log data for basic traffic analysis and monitoring CDN health. In case of issues with third-party destinations, such as latency or connection problems, data will be lost after three failed retries to connect. You should take these limitations into account before using data served on your stream for audit, compliance and billing purposes. See *DataStream use cases*.
Does DataStream store log data in case of stream failures?	DataStream doesn't store data streamed to third-party destinations. In case of stream failure, the stream attempts to reconnect three times every 5 minutes. After three failures over 15 minutes, the log data is lost.
I don't use a log analytics tool. Can I still use DataStream?	You need a tool to parse and visualize the DataStream output. You could choose lower cost, open source stacks for data parsing and visualization in human readable formats.
Does DataStream support security event logs?	DataStream is a log delivery product for all transactional events and associated metrics. You can use the SIEM Integration product to deliver security logs. See *SIEM Integration*.

Question	Answer
Why does one or more custom fields return ^ in log files?	DataStream requires enabling custom fields, such as User-Agent, Accept-Language, Cookie, Referrer or X-Forwarded-For in the Log Delivery Service behavior in your property configuration. If log lines return ^, first enable the fields in Property Manager. See *Enable logging custom parameters*.

Notice

Akamai secures and delivers digital experiences for the world's largest companies. Akamai's Intelligent Edge Platform surrounds everything, from the enterprise to the cloud, so customers and their businesses can be fast, smart, and secure. Top brands globally rely on Akamai to help them realize competitive advantage through agile solutions that extend the power of their multi-cloud architectures. Akamai keeps decisions, apps, and experiences closer to users than anyone — and attacks and threats far away. Akamai's portfolio of edge security, web and mobile performance, enterprise access, and video delivery solutions is supported by unmatched customer service, analytics, and 24/7/365 monitoring. To learn why the world's top brands trust Akamai, visit *www.akamai.com*, *blogs.akamai.com*, or *@Akamai* on Twitter. You can find our global contact information at *www.akamai.com/locations*.

Akamai is headquartered in Cambridge, Massachusetts in the United States with operations in more than 57 offices around the world. Our services and renowned customer care are designed to enable businesses to provide an unparalleled Internet experience for their customers worldwide. Addresses, phone numbers, and contact information for all locations are listed on *www.akamai.com/locations*.

Published 1/2022