TIBCO FTL® - Enterprise Edition

Monitoring

Version 7.0.1 | December 2024

Contents

About this Product

TIBCO® is proud to announce the latest release of TIBCO FTL® software.

This release is the latest in a long history of TIBCO products that use the power of Information Bus® technology to enable truly event-driven IT environments. TIBCO FTL software is part of TIBCO Messaging®. To find out more about TIBCO Messaging software and other TIBCO products, please visit us at www.tibco.com.

Client Monitoring

Important: Use TIBCO® Messaging Monitor for TIBCO FTL® in place of the FTL monitoring component. As of FTL 6.10.0, the FTL Monitoring component (monitoring directory) including Grafana and tibmongateway are removed along with monitoring metric types. See the Release Notes, Deprecated and Removed Features for a complete list.

The TIBCO FTL client library monitors each client, sampling several values that reveal operating behavior and performance. The client library sends these monitoring samples to the FTL server, which publishes them, and displays the most recent values for administrators.

Scope

The TIBCO FTL library gathers monitoring data from the following entities:

- Application program clients
- FTL services, such as realm services, bridge services, persistence services, and eFTL services
- The FTL server itself

In the Administrative UI you can use the `Collect Subscription Statistics` setting to control whether clients and services collect fine-grained monitoring metrics about subscriptions and message substreams.

Sampling

Throughout the realm, every client samples data at a regular interval. The realm property `Client Monitoring Sample Interval` specifies that interval. Zero is a special value, instructing all clients to disable the monitoring feature.

Each client sends all its pending sample data to the FTL server in the next client heartbeat message. The first batch of data becomes available with the first heartbeat after the first sample interval. If the heartbeat interval is large, then the FTL server receives a batch of accumulated samples in each heartbeat).

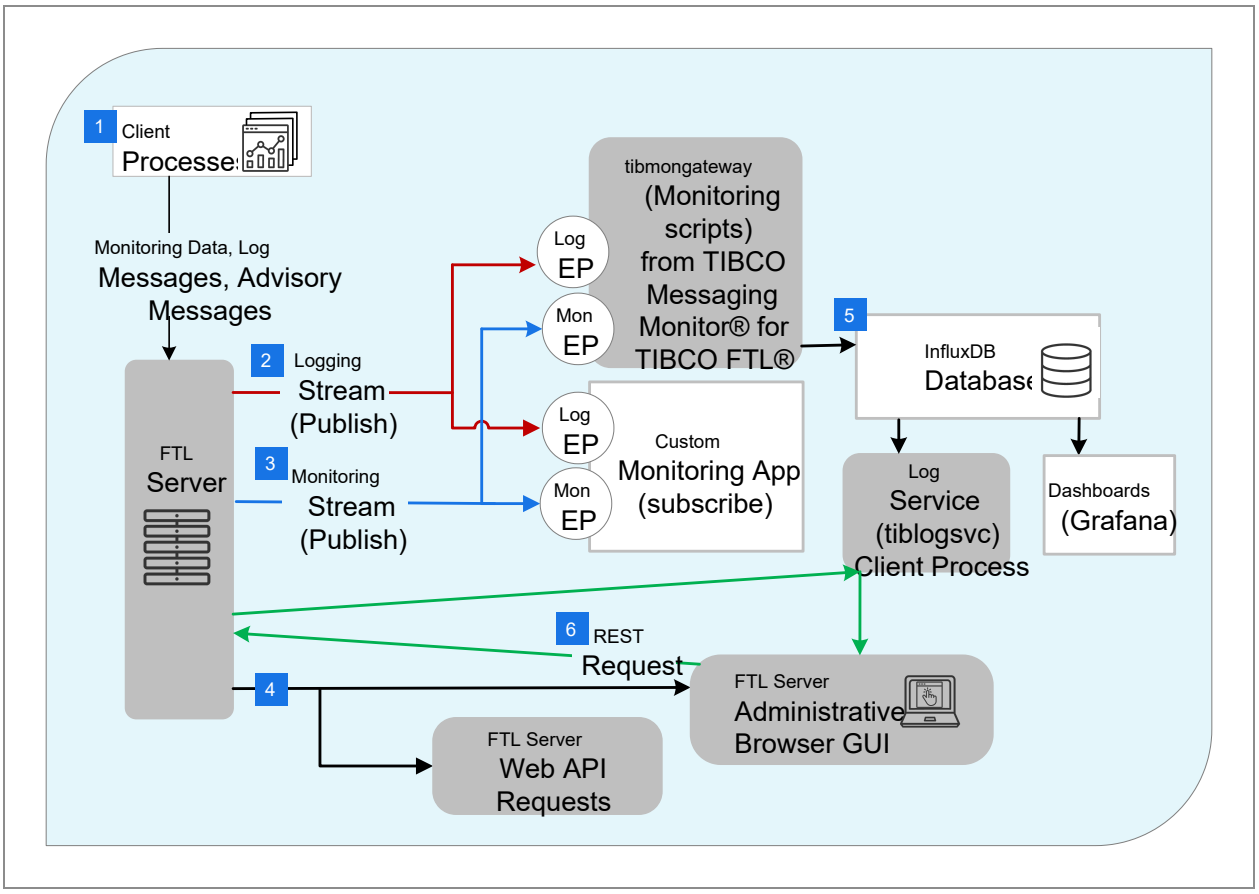
FTL Server Monitoring

The TIBCO FTL server also monitors its own operation, sampling several values that measure operating behavior and performance. The FTL server publishes these metrics, and displays the most recent values.

Data Flow of Monitoring Data and Log Messages

The following diagram illustrates the flow of FTL monitoring data and log messages.

Figure 1: Monitoring data and log flow



Explanation of the Diagram

1. All data begins in clients. The FTL client library monitors the metrics relevant to each client. It forwards monitoring data, log messages, and advisory messages to the

TIBCO FTL server.

2. When the FTL server receives log messages or advisory messages from any of its clients, the FTL server immediately publishes those messages on the *logging stream*. Any FTL application can subscribe to the built-in logging endpoint to receive the logging stream.
3. When the FTL server receives monitoring data from any of its clients, the FTL server immediately publishes that data on the *monitoring stream*. Any FTL application can subscribe to the built-in monitoring endpoint to receive the monitoring stream.
4. The FTL server retains only the most recent value of each metric and the most recent log from each client. These values are available from the FTL server through its web API and administrative GUI. You can develop custom monitoring applications that subscribe to these streams and use their data.
5. tibmongateway (or Monitoring scripts) from TIBCO® Messaging Monitor for TIBCO FTL® subscribes to both the monitoring stream and the logging stream, and stores the data in an InfluxDB database. InfluxDB is an open-source database. You can access the data directly from InfluxDB, or using other tools that manipulate and display the data, for example, Grafana.
6. REST requests can be sent from the web browser, via the FTL Server, to retrieve log entries.

The Monitoring Endpoint

The stream of monitoring metric data is available to any client program through the monitoring endpoint.

The FTL server publishes a stream of messages that contain monitoring metrics data.

Application programs can subscribe to this stream on the monitoring endpoint. The name of this endpoint is available as a constant value in the TIBCO FTL API library.

Subscribing to the Monitoring Stream

To receive monitoring data messages, application programs can subscribe to the built-in monitoring endpoint. An application can use a content matcher to narrow the message stream.

Procedure

1. Optional. Compose a content matcher.
    - To receive the entire stream, omit any content matcher.
    - To receive a sub-stream, construct an appropriate content matcher.

      For example, an application could subscribe to all monitoring messages, or to a sub-stream such as the following:

        - Only event messages.
        - Only client metric messages with a specific client label.
        - Only messages that report a specific client status.
        - Only messages about a specific host computer.

      See Fields of Monitoring Data Messages.
2. Create a subscriber object.
    a. Specify the API constant that denotes the monitoring endpoint.
    b. Optional. Specify the content matcher from step 1.

3. Ensure that the application program includes the other tasks required in any subscribing program.

   For example, the program must define appropriate callbacks, create an event queue, add the subscriber to the event queue, and start a dispatch loop.

   See "Structuring Programs" in TIBCO FTL Development.

Structure of Monitoring Data Messages

The FTL server publishes a stream of monitoring data messages. Each message can contain a group of samples, that is, data values. Each monitoring message also contains fields that identify and describe those samples.

Types of Monitoring Messages

A monitoring message can contain metric samples from a client, metric samples from the FTL server, or data about an event. The `msg_type` field distinguishes among these three types of message.

Top Level Fields

Many of the top level fields of a monitoring message contain descriptive information about the samples or event. For example, they describe the application program, client process, and the sampling time.

This descriptive information applies to all of the data samples in the message. For example, all the samples in the message come from the same client, and report data from the same sampling interval.

Data Sample Submessage

A monitoring message can contain data samples of multiple metrics. The `metrics` field contains an array of data sample submessages.

Data sample submessages can include the following fields:

Context

The object that this sample measures. For example, the context value could be the human-readable name of a specific endpoint.

A set of data sample submessages with the same value in the context field all contain information about the same monitored object.

Type of the metric

The type of counter, that is, the kind of measurement. For example, type 11 denotes the bytes sent metric, which counts the number of bytes sent outbound through a transport. For descriptions of the metric types, see .

Value

The numeric value of the counter at the end of the sampling interval.

ID

The unique internal identifier of the counter. This ID unambiguously identifies the specific counter, even though the context and type together are usually sufficient to describe the counter.

Fields of Monitoring Data Messages

The format of monitoring data messages includes many fields that are common to all monitoring messages, and some fields that are present only in certain types of monitoring messages. The following tables describe the fields and their values.

Top Level Fields

Field Name	Data Type	Description
msg_type	long	The type of a monitoring message indicates the variable portion of its contents. For a table of possible values, see Message Type Values. Each monitoring message type number is defined as a constant value in the TIBCO FTL API library. See also, Catalog of Application Metrics.
client_label	string	When connecting to the FTL server, a client program can supply a label that denotes that particular client. Unlike a client ID, which denotes a specific client *process*, this label remains unchanged even when the client restarts as a new process.

Field Name	Data Type	Description
		You can use this label to identify clients by their role within your enterprise.
client_status	string	This string denotes the status of the client reporting the metrics. For details, see the Status field in "Client Status Reference" in TIBCO FTL Administration.
metrics	message array	Each message in this array contains one data sample of one metric. For a table of possible fields of these sample messages, see Data Sample Submessage. This field contains a value only if the msg_type is *client metric* or *server metric*.
event_type	long	If the msg_type is *event*, this value indicates the type of event. For a table of possible values, see Event Type Values. Each event type number is defined as a constant value in the TIBCO FTL API library.
timestamp	long	The sampling time or event time (in microseconds after the epoch) for the metrics in this message.
application	string	The application name of the client. This field contains a value only if the msg_type is *client metric*.
application_instance	string	The application instance name of the client. This field contains a value only if the msg_type is *client metric*.
client_id	string	The client ID of the client process. The FTL server assigns each client process a unique client ID. This field contains a value only if the msg_type is *client metric*.

Field Name	Data Type	Description
server_id	string	The UUID of the FTL server. This field contains a value only if the msg_type is *server metric*.
pid	long	The process ID of the client process, as assigned by the operating system of the client's host computer. This field contains a value only if the msg_type is *client metric*.
ftl_user	string	The client supplied this value for the `username` property in its realm connect call.
effective_user	string	The username that started the client process, according its host computer's operating system. The client process inherits permissions from this user.
version	string	The version of the TIBCO FTL client library that the client or FTL server process uses.
host	string	The host name of the computer where the client process is running.
ip	string	For a client process, this IP address denotes the network interface through which the client communicates with the FTL server. For an FTL server, this IP address denotes the primary interface of the host computer.
server_uptime	long	Time (in seconds) that the FTL server process has been running. This field contains a value only if the msg_type is *server metric*.

Message Type Values

These values occur in the msg_type field of monitoring messages.

Each monitoring message type number is defined as a constant value in the TIBCO FTL API library.

Value	Description
1	Client Metric The message contains client metrics in an array of data sample submessages.
2	Server Metric The message contains FTL server metrics in an array of data sample submessages.
3	Event The message contains information about an event, such as a new client connection or a state change. The value of event_type field denotes the event.

Data Sample Submessage

Each sample submessage contains all of these fields.

Required Fields

Field Name	Data Type	Description
context	string	This human-readable string identifies the context of the metric within the client application. For example, if the metric measures some aspect of a specific endpoint, transport, or queue, then the context is the endpoint name, transport name, or queue name.
id	long	This internal ID number uniquely identifies the context of the metric.
context_type	long	This number categorizes the context of the metric, for example, as an endpoint or as a transport. Values are API constants. Programs can test these values (for example, in a switch clause).

Field Name	Data Type	Description
type	long	This number identifies the type of metric. For a table of possible metrics, see Catalog of Application Metrics. Each type number is defined as a constant value in the TIBCO FTL API library.
value	long	This field contains the data value of the metric. (Note: Use `value_long` or `value_string` instead, when possible.)
long_value	long	This field contains the data value of the metric, for data type long (preferred over the `value` field above).
string_value	string	This field contains the data value of the metric, for data type string.

A sample submessage can also contain a subset of these fields.

Additional Fields

Field Name	Data Type	Description
start_time	long	An individual monitored object stores its creation timestamp.
end_time	long	When FTL destroys a monitored object, it reports this destruction timestamp in the final monitoring messages for that object.
local_transport	string	Transport objects and sub-objects only. The name of the transport definition local to the monitored application.
remote_transport	string	Transport and connection objects only. The name of the transport definition at the opposite end of the transport bus.

Field Name	Data Type	Description
remote_client_id	string	Transport objects and sub-objects only. The client ID of the FTL client process at the opposite end of the transport bus.
connection_id	long	Transport connection objects only. The identifier that represents a monitored object or sub-object.
endpoint_name	string	The name of the endpoint in the application definition.
endpoint_mon_id	long	An identifier that represents the endpoint. Within a client process, a one-to-one mapping pairs endpoint names with endpoint identifiers.
subscriber_label	string	The application can supply this label when it creates a subscriber.
publisher_label	string	The application can supply this label when it creates a publisher.
matcher_string	string	The content matcher (in JSON representation) of a subscriber.
mcast_send_id	long	Within a monitored sub-object, this identifier represents its super-object, which is a multicast publisher. (It is the connection identifier of the super-object.)
mcast_recv_id	long	Within a monitored sub-object, this identifier represents its super-object, which is a multicast subscriber. (It is the connection identifier of the super-object.)
mcast_group	string	Sub-objects of multicast transports only. A multicast send or listen group specified in the transport definition.

Field Name	Data Type	Description
mcast_ interface	string	Sub-objects of multicast transports only. Local interface that a multicast transport uses for communication.
remote_ip	string	Transport objects and their sub-objects only. IP address of the host at the opposite end of the transport bus
remote_port	string	Transport objects and their sub-objects only. Port number at the opposite of the transport bus.
local_ip	string	Transport objects and their sub-objects only. IP address of the host of the monitored application.
local_port	string	Transport objects and their sub-objects only. Port number at the local end of the transport bus.

Event Type Values

If the msg_type is *event*, these values can occur in the event_type field to indicate the type of event.

Value	Description
1	Connected A client process has connected to the FTL server.
2	Disconnected A client process has disconnected from the FTL server.
3	Status Change

Value	Description
	The status of a client process has changed. The client_status field indicates the new status.

Connections Unravel Substreams

Connections capture fine-grained monitoring metrics about message substreams within a transport. You can use connection metrics to analyze communication patterns and to diagnose network problems.

Programs and administrators cannot access or manipulate connections.

Conceptually, you can think of a connection as if it were one side of a miniature TCP connection within a transport, either the sending or receiving side.

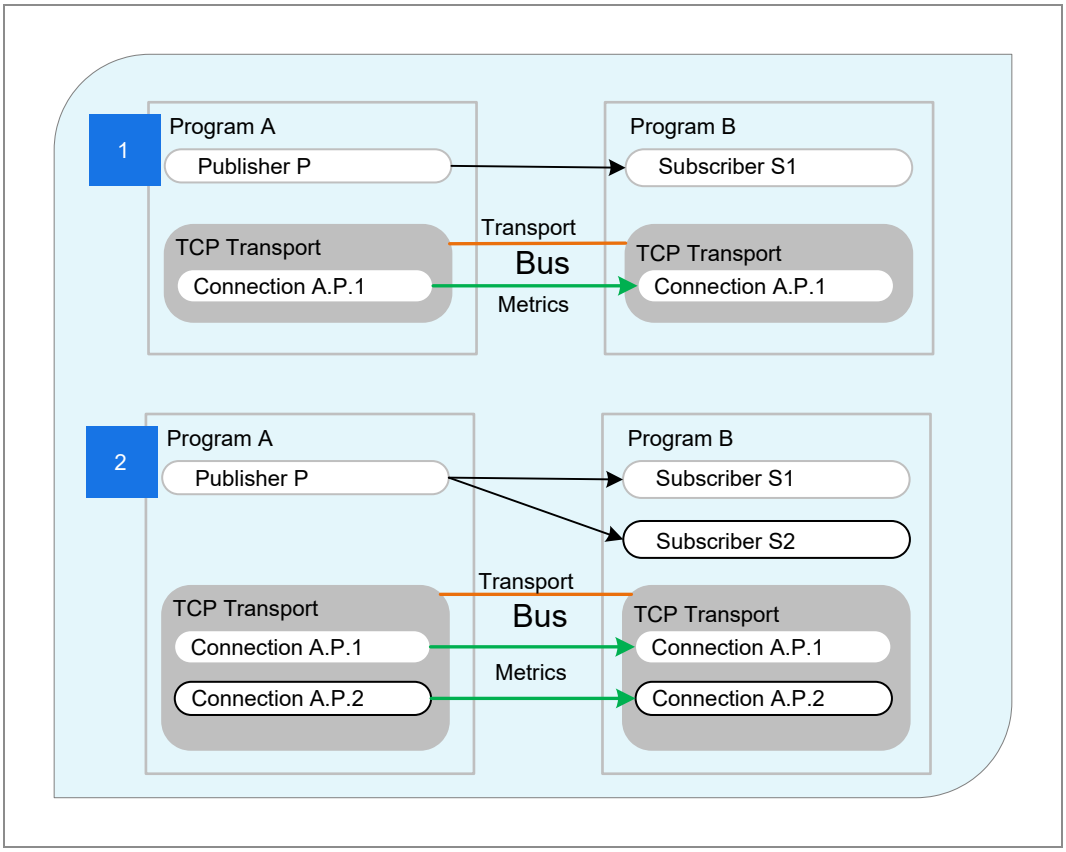
As publishers and subscribers establish a transport bus, or join an existing bus, FTL software creates connection objects to gather operational metrics. Each connection object represents a message substream.

See Catalog of Application Metrics, Catalog of Application MetricsCatalog of Application Metrics.

TCP Connections Example

To understand the concepts, consider this example of two programs, each with one endpoint, and a TCP transport configured between those endpoints.

Figure 2: Connections in a TCP Transport



1. Program A creates a publisher, P, and program B creates a subscriber, S1, and together they establish a TCP transport bus.

   FTL's monitoring layer opens a connection object at each side to gather operational metrics. Connection A.P.1 represents the sending side, and connection B.S.1 represents the receiving side.

2. Program B creates another subscriber, S2, with a different content matcher, and it joins the existing TCP bus.

   The monitoring layer opens another set of connection objects. Connection A.P.2 represents the sending side, and connection B.S.2 represents the receiving side. Now one connection represents each subscriber within the transport in program B. However, two connections represent publisher P, one for each subscriber that

receives messages from it (S1 and S2).

Even though a single TCP transport bus (orange line) carries the message stream between the two programs, the connection objects A.P.1 and A.P.2 separate the metrics for the two substreams (green arrows) outbound to the two subscribers. Moreover you can examine those substream metrics at either side of the network.

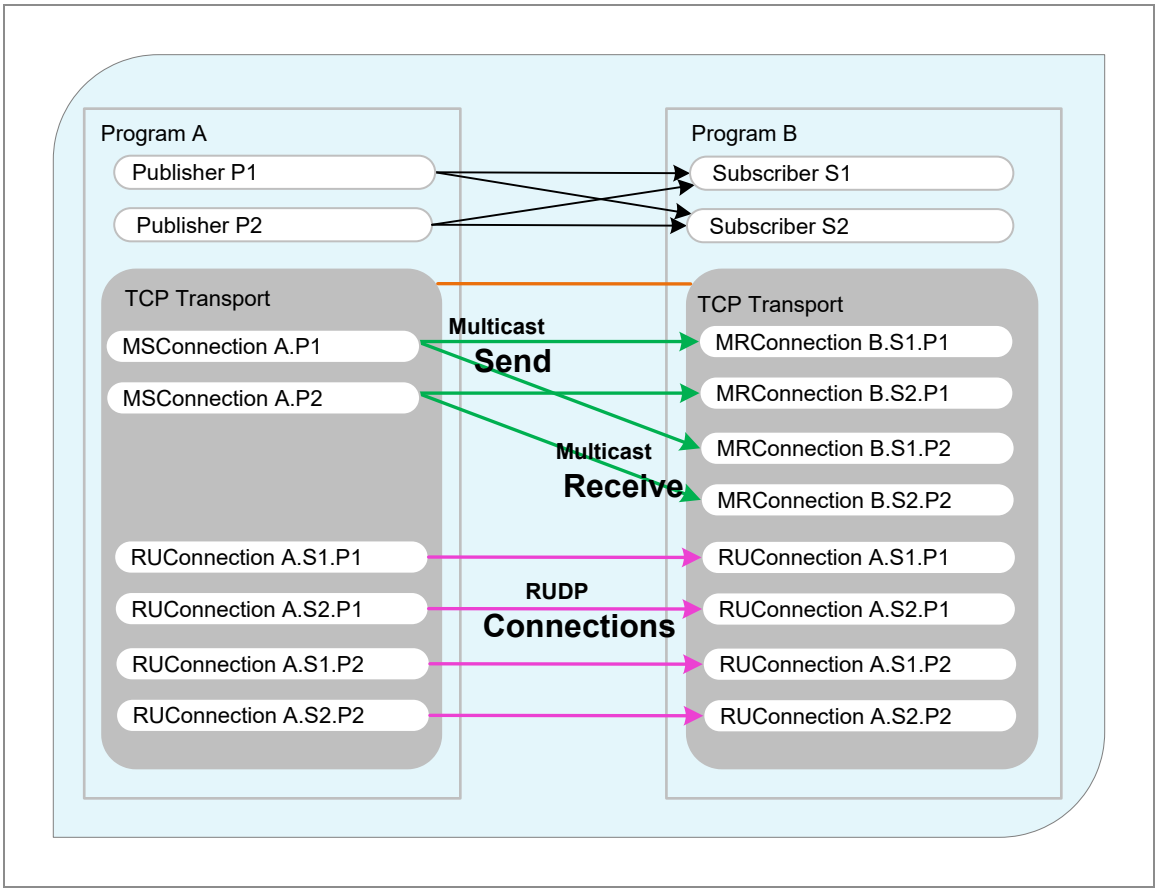
For simplicity, the diagram illustrates message flow in only one direction, even though connections monitor flow in both directions.

The same concepts apply to other transport protocols based on pair connections.

Multicast Connections Example

Even within a multicast transport bus this analogy still helpful, though the model is more complex.

Figure 3: Connections in a Multicast Transport



- ***Multicast Send*** *A multicast send connection represents the message substream from a single publisher on a multicast transport.*

  Unlike the TCP example, a multicast send connection cannot separately measure its outbound substreams to different subscribers. However, if a sending program creates multiple publishers on a multicast transport, the monitoring layer measures the substream from each publisher with a separate multicast send connection (green MSConn connections).

- ***Multicast Receive*** *A multicast receive connection represents the message substream into a subscriber from one specific publisher on a multicast transport (that is, one multicast send connection).*

  A subscriber that joins a multicast transport bus could receive many substreams from multiple publishers on that transport. A separate multicast receive connection gathers metrics corresponding to each substream (green MRConn connections).

- ***RUDP*** *When a multicast transport enables the send inbox ability or receive inbox ability, RUDP connections represent the inbox substreams between each multicast sender and each of its receivers (pink RUConn connections).*

  As in the TCP example, a pair of unique connections measures each side of a substream.

Catalog of Application Metrics

These tables describe metrics that you can use to assess the operation of a client application or service.

TIBCO FTL software gathers metrics for clients (applications), event queues, transports, and endpoints. It can also gather metrics for subscribers, publishers, and connections.

Unless otherwise specified, a metric reports a counter's value at the end of each sample interval, and the counter resets to zero for the next interval.

Each data sample submessage contains a `type` field. Its value is one of the type ID numbers in these tables, denoting a metric type.

Application Metrics

These metrics measure the operation of a client.

ID	Metric Type	Description
51	Dynamic Formats	Number of distinct dynamic formats registered within the client. Creating a message with a dynamic format can increment this counter. Receiving an inbound message with a dynamic format can increment this counter. This counter can decrease as the client library automatically unregisters unused formats.
70	Resident Memory Set Size	Current size in kilobytes of an application's working memory set.
71	Peak Resident Memory Set Size	Maximum size in kilobytes, since the client started, of an application's working memory set.
72	Process VM	Current size in kilobytes of an application's virtual memory set.
73	User CPU Time	Total user CPU time in microseconds, that is, time executing the client's object code since the client started.
74	System CPU Time	Total system CPU time in microseconds, that is, time executing operating system calls from the client since the client started.
75	Total CPU Time	Total CPU time in microseconds, that is, the sum of time executing the client's object code and its operating system calls since the client started.

Endpoint Metrics

These metrics report activity on an endpoint of a client.

Endpoint

ID	Metric Type	Description	Tracking
67	Store Mismatch Messages	Number of message flows that result from persistence store mismatch. Any non-zero value indicates a store mismatch misconfiguration. *Store mismatch* is a misconfiguration in which a direct path transport connects two endpoints that are associated with two *different* persistence stores.	Per endpoint

Transport

ID	Metric Type	Description	Tracking
12	Bytes Received	Inbound data bytes on a transport in the client.	Per transport
11	Bytes Sent	Outbound data bytes on a transport in the client.	Per transport
31	Data Lost	Inbound data loss events on a transport in the client. It is not possible to measure the magnitude of the events in bytes.	Per transport
32	Format Unavailable	Messages with an unrecognized format carried on a transport in the client.	Per transport

Note: Bytes sent and received need not correlate exactly with messages sent and received on corresponding endpoints. For example, filtering and optimizations can decouple these metrics.

Queues

These metrics report activity in each individual event queue in a client.

ID	Metric Type	Description
41	Queue Backlog	Maximum number of messages in an event queue in the client during the sample interval.
42	Queue Discards	Inbound messages discarded by a queue in the client.

Metrics for Multicast and RUDP

TIBCO FTL software gathers additional metrics for transports and connections that use UDP-based protocols, such as multicast and RUDP transports.

Unless otherwise specified, a metric reports a counter's value at the end of each sample interval. Transport metrics are cumulative since the transport established its bus. Connection metrics are cumulative since the connection's start time.

Multicast and RUDP Transports

ID	Metric Type	Description	Tracking
61	Packets Sent	Outbound data packets on a multicast or RUDP transport. This counter includes *all* outbound data packets, including original data packets and retransmitted data packets.	Per transport
62	Packets Received	Inbound data packets on a multicast or RUDP transport. This counter includes *all* inbound data packets, including original data packets and retransmitted data packets.	Per transport
63	Packets Retransmitted	Outbound packets retransmitted on a multicast or RUDP transport. If the transport retransmits a packet several times, it increments this counter for each retransmission.	Per transport

ID	Metric Type	Description	Tracking
64	Packets Missed	Number of missed inbound packets on a multicast or RUDP transport.	Per transport
65	Packets Lost Outbound	Number of outbound packets discarded on a multicast or RUDP transport.	Per transport
66	Packets Lost Inbound	Number of inbound packets deemed lost on a multicast or RUDP transport. The transport accounts a packet as lost after its reliability constraints expire.	Per transport

Multicast Connection

ID	Metric Type	Description	Tracking
90040	Send Backlog Maximum Size	The largest data backlog (in bytes) during this monitoring interval (not cumulative). For background information see "Send: Blocking versus Non-Blocking" in the TIBCO FTL documentation.	Per multicast sender

Catalog of Persistence Metrics

This table describes metrics that you can use to assess the operation of persistence stores.

Each persistence service is itself an application client, and tracks application metrics; see Catalog of Application Metrics.

The sampling interval for persistence metrics is fixed at 2 seconds. You cannot change this interval.

See the API Documentation, `monitor.h` and other metric sections for counters that are useful to you.

Store Metrics

ID	Metric Type	Description	Tracking
1000	Message Count	Number of messages in a store at the end of the sample interval.	Per Store
1001	Message Size	Storage (in bytes) that messages occupy in a store at the end of the sample interval.	Per Store
1006	Swap Message Count	When `disk_swap` is true, this is the number of messages in memory, else 0.	Per Store
1007	Swap Message Size	When `disk_swap` is true, this is the byte count of all messages in memory, else 0.	Per Store
1008	Store Byte Limit	Configured store memory byte limit.	Per Store
1009	Store Message Limit	Configured store message count limit.	Per Store
1010	Expiration Count	Counts the number of	Per Store

ID	Metric Type	Description	Tracking
		message expiration events in a store for the lifetime of the current leader. This includes messages discarded due to message TTL (Time-To-Live), durable message limit, durable byte limit, or durable max delivery. The count is incremented when any message is expired by any durable. The count will increment multiple times if a given message is expired by multiple durables.	
2000	Durable Count	Number of durables in a store at the end of the sample interval.	Per Store
3013	Disk Size	Allocated disk space for this	Per Persistence Service

ID	Metric Type	Description	Tracking
		server's persistence data. This is a high-water mark for disk space in use.	
3014	Backlog Limit	Soft limit on memory for pending messages.	Per Persistence Service
3015	Backlog Size	Memory in use for pending messages.	Per Persistence Service
3016	Disk Backlog Size	Memory in use for pending disk writes.	Per Persistence Service
3017	Inbound Byte Count	Byte count of messages received from client applications or other persistence clusters.	Per Store
3018	Outbound Byte Count	Byte count of messages delivered to client applications or other persistence clusters.	Per Store

ID	Metric Type	Description	Tracking
3019	Disk In Use Size	Disk space currently in use for this server's persistence data.	Per Persistence Service
3020	Disk Capacity	Capacity of the disk volume containing the persistence data directory.	Per Persistence Service
3021	Disk Usage Limit	Attempt to limit total space used (including non-FTL data) to this value, based on disk capacity and max disk fraction.	Per Persistence Service
3022	Disk Available	Available space on the disk volume containing the persistence data directory.	Per Persistence Service
3023	Disk Used	Total space used on the disk volume containing the persistence data directory.	Per Persistence Service
3024	Disk Size (Swap) .	Allocated disk space for this server's swap	Per Persistence Service

ID	Metric Type	Description	Tracking
		data	
3025	Disk In Use Size (Swap) .	Disk space currently in use for this server's swap data	Per Persistence Service
3026	Disk Capacity (Swap) .	Capacity of the disk volume containing the swap data directory.	Per Persistence Service
3027	Disk Usage Limit (Swap)	Attempt to limit total space used for swap (including non-FTL data) to this value, based on disk capacity and max disk fraction.	Per Persistence Service
3028	Disk Available (Swap)	Available space on the disk volume containing the swap data directory.	Per Persistence Service
3029	Disk Used (Swap)	Total space used on the disk volume containing the swap data directory.	Per Persistence Service

ID	Metric Type	Description	Tracking
3030	Disk Write Count	Total number of write calls issued to the persistence data file.	Per Persistence Service
3031	Disk Bytes Written	Total bytes written to the persistence data file.	Per Persistence Service
3032	Disk Read Count	Total number of read calls issued to the persistence data file.	Per Persistence Service
3033	Disk Bytes Read	Total bytes read from the persistence data file.	Per Persistence Service
3034	Disk Fdatasync Count	Total number of fdatasync calls issued to the persistence data file.	Per Persistence Service
3035	Replica Round-Trip Time: Count	Sample count of all round-trip times to a replica. The metric context identifies the replica.	Per Persistence Service
3036	Replica Round-Trip	Maximum round-	Per Persistence

ID	Metric Type	Description	Tracking
	Time: Max	trip time to a replica over the last sample interval, in microseconds. The metric context identifies the replica. If disk persistence is configured, the round-trip time includes the sync write to disk at the replica.	Service
3037	Replica Round-Trip Time: Total	Total of all round-trip times to a replica, in microseconds. The metric context identifies the replica. If disk persistence is configured, the round-trip time includes the sync write to disk at the replica.	Per Persistence Service
3038	Replica Round-Trip Bytes Per Op: Count	Sample count of	Per Persistence Service

ID	Metric Type	Description	Tracking
		all round-trips to a replica. The metric context identifies the replica.	
3039	Replica Round-Trip Bytes Per Op: Max	Maximum bytes per round-trip to a replica over the last sample interval. The metric context identifies the replica.	Per Persistence Service
3040	Replica Round-Trip Bytes Per Op: Total	Total byte count for all round-trips to a replica. The metric context identifies the replica.	Per Persistence Service
3041	Replica Round-Trip Time (Async Disk): Count	Sample count of all round-trip times to a replica. The metric context identifies the replica. For async disk persistence only.	Per Persistence Service
3042	Replica Round-Trip Time (Async Disk): Max	Maximum round-trip time to a replica over the	Per Persistence Service

ID	Metric Type	Description	Tracking
		last sample interval, in microseconds. The metric context identifies the replica. For async disk persistence only.	
3043	Replica Round-Trip Time (Async Disk): Total	Total of all round-trip times to a replica, in microseconds. The metric context identifies the replica. For async disk persistence only. The round-trip time only includes an async write to disk at the replica.	Per Persistence Service
3044	Replica Round-Trip Bytes Per Op (Async Disk): Count	Sample count of all round-trips to a replica. The metric context identifies the replica.	Per Persistence Service
3045	Replica Round-Trip Bytes Per Op (Async Disk): Max	Maximum bytes per round-trip to a replica over	Per Persistence Service

ID	Metric Type	Description	Tracking
		the last sample interval. The metric context identifies the replica. For async disk persistence only.	
3046	Replica Round-Trip Bytes Per Op (Async Disk): Total	Total byte count for all round-trips to a replica. The metric context identifies the replica. For async disk persistence only	Per Persistence Service
3047	Disk Write Latency: Count	Sample count of all sync writes to disk.	Per Persistence Service
3048	Disk Write Latency: Max	Maximum latency of a sync disk write over the last sample interval, in microseconds.	Per Persistence Service
3049	Disk Write Latency: Total	Total latency of all sync disk writes, in microseconds.	Per Persistence Service
3050	Disk Write Bytes Per Batch: Count	Sample count of all sync writes to	Per Persistence Service

ID	Metric Type	Description	Tracking
		disk.	
3051	Disk Write Bytes Per Batch: Max	Maximum bytes in a sync disk write over the last sample interval.	Per Persistence Service
3052	Disk Write Bytes Per Batch: Total	Total bytes in all sync disk writes.	Per Persistence Service

Catalog of eFTL Metrics

This table describes metrics that you can use to assess the operation of eFTL channels.

Each TIBCO eFTL service is itself an application client, and tracks application metrics: for example, metrics for the server's endpoints, and metrics for the transports that implement those endpoints. See Catalog of Application Metrics.

The sampling interval for eFTL metrics is fixed at 2 seconds. You cannot change this interval.

eFTL Metrics

ID	Metric Type	Description	Tracking
4000	Connection Count	Number of eFTL client connections on a channel at the end of the sample interval.	Per channel
4001	Subscription Count	Open subscriptions from eFTL clients on a channel at the end of the sample interval.	Per channel
4002	Inbound eFTL Message Count	Messages inbound from eFTL clients on a channel.	Per channel
4003	Outbound	Messages outbound to eFTL clients on a channel.	Per channel

ID	Metric Type	Description	Tracking
	eFTL Message Count		
4010	Discarded Message Count	Data loss from publishers: the number of messages that the server discarded on a channel. Discards can occur because a publisher sent a message that exceeded the maximum message size.	Per channel
4011	Dataloss Message Count	Data loss to subscribers: the number of messages that the server discarded on a channel. Discards can occur because a subscriber's queue exceeded the maximum outbound backlog.	Per channel
4012	Connection Pending Count	Number of new eFTL client connections on a channel that are not yet fully initialized at the end of the sample interval.	Per channel
4013	Connection Suspended Count	Number of eFTL client connections disconnected by a temporary network outage that have not yet automatically reconnected nor fully disconnected at the end of the sample interval.	Per channel
4018	Cumulative Connection Count	Counts all eFTL client or REST API successful connections to an eFTL channel for the given eFTL server.	Per channel

Catalog of FTL Server and Service Metrics

This table describes metrics that you can use to assess the operation of the FTL server and services.

FTL Server Metrics

The sampling interval for FTL server metrics is fixed at 2 seconds. You cannot change this interval.

ID	Metric Type	Description
5009	Client Connect Count	Cumulative number of client connections since the FTL server process started.
5010	Client Reconnect Count	Number of clients that reconnected to the FTL server during the sample interval. A spike in this metric could indicate a network connectivity issue between the FTL server and its clients.
5011	Send to Inbox Failures	Number of undelivered protocol messages that the FTL server sent to a client. A spike in this metric usually indicates that several clients abruptly exited.

Transport Bridge Metrics

Each bridge is itself an application client, and tracks application metrics; see Catalog of Application Metrics.

The sampling interval for bridge metrics is fixed at 2 seconds. You cannot change this interval.

ID	Metric Type	Description	Tracking
6000	Bridge Active	1 indicates that the bridge actively forwards messages. 0 indicates that the bridge is a backup in standby mode.	Per bridge

Estimating the Volume of Metric Data

You can estimate the size of client monitoring data to gauge its impact on network bandwidth and other resources. The number of counters depends on the number of clients, and on the number of endpoints, transports, and event queues in each client.

Procedure

1. For each application instance definition, determine the number of counters per client of that application instance. Combine information from the application instance definitions with information from the following table.

   Client Monitoring Resource Usage

11 counters per client
3 counters per endpoint
2 counters per event queue
3 counters per transport (other than multicast transports)
9 counters per multicast transport (6 specific to multicast, in addition to the 3 counters common to all transports)

2. For each application instance definition, multiply the number of counters per client (as determined in the previous step) by the expected number of client instances.
3. Sum that number of counters over all the application instance definitions to obtain the enterprise total number of data samples generated in each sample interval.

   When estimating bandwidth requirements, remember that each data sample travels between multiple components. Depending on the network architecture, the contents of a specific data sample may traverse the network several times, possibly across the same network segments or links. See Data Flow of Monitoring Data and Log Messages.
4. Multiply the enterprise total by the number of samples you plan to retain in your monitoring data repository.

This product estimates the total volume of metric data.

Security of Monitoring Data

TIBCO FTL components and clients can secure monitoring data and log data using TLS.

If your enterprise uses secure FTL servers, then data is secure as it travels from client to FTL server, and then to subscribers.

However, TIBCO makes no assertions about the security of third-party components.

FTL Server

A secure FTL server uses a secure TCP transport to communicate with its clients. TLS secures the heartbeat messages that carry monitoring data and log data from clients to the FTL server.

A secure FTL server can secure the monitoring stream and logging stream using TLS.

See "Secure FTL Servers" in TIBCO FTL Administration.

Monitoring Gateway

The monitoring gateway (`tibmongateway`) supports TLS to secure its client connection to a secure FTL server. Administrators must supply appropriate command line arguments when starting the gateway. For details, see TIBCO® Messaging Monitor for TIBCO FTL®.

InfluxDB

The monitoring gateway can secure its connection InfluxDB. For details, see TIBCO® Messaging Monitor for TIBCO FTL®.

Grafana

Grafana can use a secure connection to query InfluxDB. Grafana can secure client connections from browsers using HTTPS. For details, see TIBCO® Messaging Monitor for TIBCO FTL®.

Log Service

The log service can secure its connection to InfluxDB. The log service can secure client connections using HTTPS. For details, see "Securing Log Services" in *TIBCO FTL Security*.

Monitoring Data: Timing and Interruptions

Monitoring data flows through several segments on the path from client to database. Timing interactions along this path can affect the data. Some segments of this path retain data across interruptions, while others do not retain data.

Best Practice for Intervals

Default values for the monitoring interval and heartbeat interval are a good choice for many applications. If your enterprise needs different values, then for best results set the client monitoring sample interval and the client-server heartbeat interval to identical values.

Client

Most monitoring data begins in a client. Each client gathers data and transfers it to the FTL server in its regular heartbeat messages.

For the most complete results, ensure that the client monitoring sample interval is identical to the client-server heartbeat interval. (You can configure these intervals as global realm properties of the FTL server. However, the monitoring sample interval for persistence servers and TIBCO eFTL servers is fixed.)

If the heartbeat interval is longer than the sample interval, the client accumulates data from several sample intervals, and sends the accumulated samples to the FTL server in the next heartbeat.

If the client cannot communicate with the FTL server, the client retains its monitoring data until it successfully reconnects to the server, then it transfers the accumulated samples. Memory availability could limit the amount of data that a client can accumulate.

When a client stops or restarts, it does not preserve monitoring data it has not yet sent to the FTL server.

FTL Server

The FTL server receives data from clients, and also generates monitoring data that measures its own operations. The FTL server publishes a stream of monitoring data

messages on the built-in monitoring endpoint.

As each monitoring sample set arrives from a client, the FTL server immediately publishes it on the monitoring stream. The FTL server also stores the new data samples, overwriting previous data samples for that client.

The FTL server publishes each sample in a separate message.

The monitoring message stream is available only to current subscribers. Subscribers cannot receive any messages that the FTL server sends while a subscribing client is not running, or while the communications link is interrupted. Monitoring data in those messages is no longer available to the subscriber.

You cannot associate a persistence store with the monitoring endpoint.

Gateway

The TIBCO® Messaging Monitor for TIBCO FTL® subscribes to the monitoring message stream.

As each message arrives, the adapter immediately stores the data in the database and then discards the message.

The gateway cannot receive any messages that the FTL server sends while the gateway is not running, or while the communications link from the FTL server is interrupted. Data in those messages is no longer available to the adapter.

If the gateway service stops, or if communications from the FTL server are interrupted, then the gateway cannot receive any messages that the FTL server sent in the interim. Data in those messages is lost.

Central Logging

When application clients are distributed over many host computers, accessing their individual log files could be cumbersome. However, application and component clients can forward copies of their advisory and log messages to the FTL server, so you can access those messages through a central facility.

Note: The initial behavior for each client is to *disable* forwarding logs to the FTL server. You can enable log forwarding for individual clients using the FTL server web API or the GUI's client status table.

Motivation

Traditionally, each process would output its log messages to either the console, a log file on its host computer, or both. Traditionally, the logging behavior of each process would be determined by the process itself. This model is disadvantageous when an administrator must track many processes, and especially when application clients run in cloud containers.

With central logging you can easily access log messages from any subset of clients, search a consolidated database of log messages, view and analyze log activity in a graphic display, and apply log management products such as TIBCO LogLogic.® Administrators can also control logging behavior dynamically.

Central Logging Features

- You can manipulate the logging behavior of clients by using the FTL server GUI or web API.
- Clients can forward advisory and log messages to the FTL server.
- The FTL server can publish those messages on a dedicated log stream, which is available to every client.
- A monitoring gateway component subscribes to those messages and stores them in a database.

- A log service responds to REST requests by querying the database.
- A sample Grafana dashboard also displays those messages from the database.
- You can build custom applications that subscribe to the log stream.

Clients Forward Log Messages

When log forwarding is enabled, clients forward advisory messages and log messages to the FTL server, but the two types of messages have different forwarding behavior.

Clients forward advisory messages immediately.

Clients accumulate log messages during a heartbeat interval, and forward them in a batch at the end of the interval. The FTL library limits the number of messages in each batch, to protect against programs that emit logs while stuck in a loop. After exceeding the batch size, the library discards new messages. Each interval begins with an empty batch.

Log Stream

The FTL server receives advisory messages and log messages from its clients and services, and publishes them in a consolidated log stream.

After publishing the messages, the FTL server discards them. It retains only the most recent messages from each client, so it can display them in the FTL server GUI and respond to web API requests.

Clients can subscribe to the log stream using a special logging endpoint. The FTL API defines this endpoint as the value of a constant.

Log Message Structure

Each message in the log stream contains an *array* of individual log messages, plus fields that describe them. Your programs can access fields of the individual log messages and the outer container message by using field names that the API defines.

Fields of the Log Stream Message (Outer Container)

Field Content
Client ID
Timestamp
Client Label
Array of Logs
FTL Server ID
Application Name
Application Instance Name
PID
username
Host Name
Host IP Address

Fields of Individual Log Messages (Inside the Array)

Field Content
Client ID
Timestamp
Log Level
FTL Component
FTL Context (for example, the name of an endpoint, transport, or event queue)

Field Content
Statement (the text string of the message)
Exception
Advisory Flag

Note: Both the outer container and the inner log messages can contain fields with the client ID and timestamp.

Log Service

The log service queries the database for log messages and advisories.

The log service is an FTL service component. To enable its functionality, you must configure and run this service.

Before starting the log service, ensure that the FTL server and InfluxDB server are already running.

Security

Security of log data requires an unbroken chain of secure connections:

- FTL clients to the FTL server
- FTL server to monitoring gateway
- TIBCO® Messaging Monitor for TIBCO FTL®
- Log service to its HTTPS clients

Log Service Web API

You can use the log service web API to query the database of logs and advisory messages.

The FTL server is the central contact point for the log service. That is, send log service REST requests to the FTL server.

When sending requests using the log service web API, address them to URIs with one of these prefixes:

```
http://<host>:<port>/logsvc/v1
```

```
https://<host>:<port>/logsvc/v1
```

- <host>:<port> is the FTL server's location.
- v1 denotes the version of the web API. This version number could change in subsquent releases.

GET realms

The web method `GET realms` retrieves a list of realms from which the log service collects logs. Use these realm IDs in other log service `GET` calls.

Example Requests

```
curl -X GET http://<host>:<port>/logsvc/v1/realms
```

Example JSON Representation

This request returns a response with this form.

```
{
"realms":
  [
    {"realm_uuid": "5bf95cd4-7035-4842-894a-49438d6da9e0"},
    {"realm_uuid": "743362b7-e56c-43b0-b056-dd8824dc71fa"}
  ]
}
```

GET <realm_UUID>/logs

The web method `GET <realm_UUID>/logs` retrieves the list of available log files for clients in the realm, including metadata that describes each file.

To obtain realm UUIDs, see `GET realms`.

You can include pagination parameters to retrieve long lists in smaller batches. For details, see "Pagination" in TIBCO FTL Administration.

Example Requests

```
curl -X GET http://<host>:<port>/logsvc/v1/<realmUUID>/logs
```

Example JSON Representation

This request returns a response with this form.

```
{
  “logfile_entries”:
  [
    {“client_id”: 1234,
     “realm_uuid”: “8a9707d13bb446feab42aa751c715a5f”,
     “row_count”: 1189,
     “creation_date”: “June 26, 2017 2:38:54 PM”,
     “last_modified”: “June 27, 2017 2:54:05 PM”,
     “bytes”: 23192},
    {client_id”: 5678,
     “realm_uuid”: “9b9707d13bb446feab42aa751c715a5f”,
     “row_count”: 2249,
     “creation_date”: “June 26, 2017 2:38:54 PM”,
     “last_modified”: “June 27, 2017 2:54:05 PM”,
     “bytes”: 45643}
  ]
}
```

GET <realm_UUID>/logs/<client_ID>

This web method retrieves the log file for a specific client, along with metadata that describes the file.

- <realm_UUID> in the URI specifies the realm.

  To obtain realm UUIDs, see `GET realms`.

- <client_ID> in the URI specifies the client.

Parameters

You can include pagination parameters to retrieve the rows of long log files in smaller batches. For details, see "Pagination" in TIBCO FTL Administration.

You can include limit parameters to truncate long files to only the most recent logs. When neither limit parameter is present, the default limit is 10MB.

Syntax	Description
`max_size=<byte_limit>`	Truncate the results to this size. Supply the limit as a numerical value with units. For example, 100MB.
`max_rows=<row_limit>`	Truncate the results to this length (in rows).

Example Requests

```
curl -X GET http://<host>:<port>/logsvc/v1/<realmUUID>/logs/<client_
ID>?max_size=100MB
```

Example JSON Representation

This request returns a response with this form.

```
{
    "client_id": 106025,
    "realm_uuid": "743362b7-e56c-43b0-b056-dd8824dc71fa",
    "row_count": 13,
    "creation_date": "2017-10-10T14:22:05-05:00",
    "last_modified": "2017-10-11T11:32:39-05:00",
    "rows":
     [
      {
        "timestamp": "2017-10-10T14:22:05-05:00",
        "level": "info",
        "component": "ftl",
        "context": "ftl_log",
        "statement": "starting primary realmserver" },
      {
        "timestamp": "2017-10-10T14:22:05-05:00",
        "level": "info",
        "component": "ftl",
        "context": "ftl_log",
        "statement": "Server is now running." }
     ]
}
```

GET <realm_UUID>/advisories

This web method retrieves advisories for a specific realm.

<realm_UUID> in the URI specifies the realm.

Parameters

You can include pagination parameters to retrieve advisories in smaller batches. For details, see "Pagination" in TIBCO FTL Administration.

You can use parameters to retrieve advisories only from FTL clients or services of a specific type. For example, only from bridge services.

Syntax	Description
`record_type=<type>`	Retrieve advisories from clients or services of this type. Valid types include: • "Client" • "Satellite" • "Persistence Server" • "Bridge" • "Group Server" • "Group Client" • "eFTL Cluster"

Example Requests

```
curl -X GET
http://<host>:<port>/logsvc/v1/<realmUUID>/advisories?record_
type="<type>"
```

Example JSON Representation

This request returns a response with this form.

```
{
  "advisories":
  [
    {
      "client_id": 1234,
      "name": "DATALOSS",
      "reason": "QUEUE_LIMIT_EXCEEDED",
      "severity": "WARNING",
      "module": "BASE",
      "aggregation_count": 0,
      "aggregation_time" : 1498244496256,
      "timestamp": 1498244496256,
      "queue_name": "my_queue",
      "subscriber_name": ""
```

```
      "endpoints": [
        "endpoint": "ep1",
        "endpoint": "ep2"
      ],
      "lock_name": ""
    },
    {
      "client_id": 5678,
      "name": "DATALOSS",
      "reason": "TPORT_DATALOSS",
      "severity": "WARNING",
      "module": "BASE",
      "aggregation_count": 0,
      "aggregation_time" : 1498244496256,
      "timestamp": 1498244496256,
      "queue_name": "my_queue",
      "subscriber_name": ""
      "endpoints": [
        "endpoint": "ep3",
        "endpoint": "ep4"
      ],
      "lock_name": ""
    }
  ]
}
```

POST shutdown

The web method `POST shutdown` requests that the log service terminate gracefully.

Example Requests

```
curl -X POST http://<host>:<port>/logsvc/v1/shutdown
```

TIBCO Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the Product Documentation website, mainly in HTML and PDF formats.

The Product Documentation website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

Documentation for TIBCO FTL® - Enterprise Edition is available on the TIBCO FTL® - Enterprise Edition Product Documentation page.

TIBCO eFTL™ Documentation Set

TIBCO eFTL software is documented separately. Administrators use the FTL server GUI to configure and monitor the eFTL service. For information about these GUI pages, see the documentation set for TIBCO eFTL software.

How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our product Support website.
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the product Support website. If you do not have a username, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the TIBCO Ideas Portal. For a free registration, go to TIBCO Community.

Legal and Third-Party Notices

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.