TIBCO eFTL™ Concepts

Software Release 3.3
October 2017

Two-Second Advantage®

TIBCO®

Important Information

Contents

About this Product

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

This release is the latest in a long history of TIBCO products that leverage the power of Information Bus® technology to enable truly event-driven IT environments. To find out more about how TIBCO eFTL software and other TIBCO products are powered by TIB® technology, please visit us at www.tibco.com.

Product Editions

TIBCO eFTL and related products are now available in a community edition and an enterprise edition.

TIBCO eFTL - Community Edition is ideal for getting started with eFTL, for implementing application projects, including proof of concept projects, for testing, and for deploying applications in a production environment. Although the community license limits the number of production processes, you can easily upgrade to the enterprise edition as your use of TIBCO eFTL expands.

The community edition is available free of charge. It is a full installation of the TIBCO eFTL product, with the following limitations and exclusions:

- Users may run up to 1000 mobile clients in a production environment, per corporate entity.
- Users do not have access to TIBCO Support. However, you can use TIBCO Community as a resource (https://community.tibco.com).

TIBCO eFTL - Community Edition is compatible with both the enterprise and community editions of TIBCO FTL®.

TIBCO eFTL - Enterprise Edition is ideal for all application development projects, and for deploying and managing applications in an enterprise production environment. It includes all features presented in this documentation set, and access to TIBCO Support. Choose the enterprise edition for production deployments with more than 1000 mobile clients, and for enterprise monitoring using customizable status dashboards.

TIBCO eFTL - Enterprise Edition uses the enterprise edition of TIBCO FTL and includes a license for it.

TIBCO Documentation and Support Services

How to Access TIBCO Documentation

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

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit https://docs.tibco.com.

Product-Specific Documentation

The following documents for this product can be found on the TIBCO Documentation site:

- *TIBCO eFTL Concepts*
- *TIBCO eFTL Administration*
- *TIBCO eFTL Development*
- *TIBCO eFTL Installation*
- *TIBCO eFTL API Reference* (HTML only)
- *TIBCO eFTL Release Notes*

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit http://www.tibco.com/services/support.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at https://support.tibco.com.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to https://support.tibco.com. If you do not have a user name, 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 https://community.tibco.com.

Purpose and Features

TIBCO eFTL software extends TIBCO messaging to platforms such as web browsers and mobile phones. Applications on these devices can use TIBCO eFTL software to communicate with one another, and with back-end applications that use infrastructure based on TIBCO FTL® software or TIBCO Enterprise Message Service™ software.

TIBCO eFTL software features high scalability at human-scale performance: that is, throughput and latency adequate for cell phones and browsers.

TIBCO eFTL software consists of a compact messaging API and a TIBCO eFTL server executable program.

TIBCO eFTL software cleanly separates the responsibilities of application developers from those of administrators, just as TIBCO FTL software does. Developers use the TIBCO eFTL API in programs. Administrators configure and run TIBCO eFTL servers, and monitor message load and performance.

TIBCO eFTL software requires the TIBCO FTL realm server for configuration and monitoring.

Basic Definitions

Understanding the basic definitions in this topic will help you understand the more detailed scenarios in the following topics.

eFTL Server

The *TIBCO eFTL server* is a software routing component. All message communications that involve eFTL applications flow through a TIBCO eFTL server.

Channel

Each server operates a set of independent channels. A *channel* carries a message stream, and keeps it separate from the message streams of other channels. That is, the server does not forward messages from one channel to another.

Sides

Each channel has two sides, and conducts messages between them. On the *eFTL side*, eFTL clients connect to the channel. The name of the other side depends on message infrastructure with which it interfaces. If it interfaces with TIBCO FTL infrastructure, it is called the *FTL side*. If it interfaces with TIBCO Enterprise Message Service infrastructure, it is called the *EMS side*.

Cluster

Within a TIBCO FTL realm, the administrator can define TIBCO eFTL clusters. A *cluster* is a set of TIBCO eFTL server processes that cooperate for scalability and redundancy. Within a cluster, each server operates the same set of channels.

Use Cases

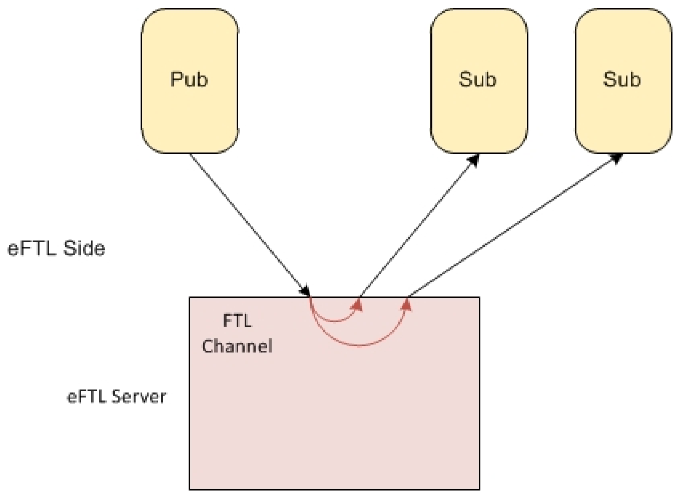
The following examples illustrate possible use cases involving TIBCO eFTL software to mediate communication among applications.

Messages among eFTL Clients

In the simplest use case, an FTL channel can forward messages among eFTL applications. (Communication with FTL applications is not a requirement for using an FTL channel.)

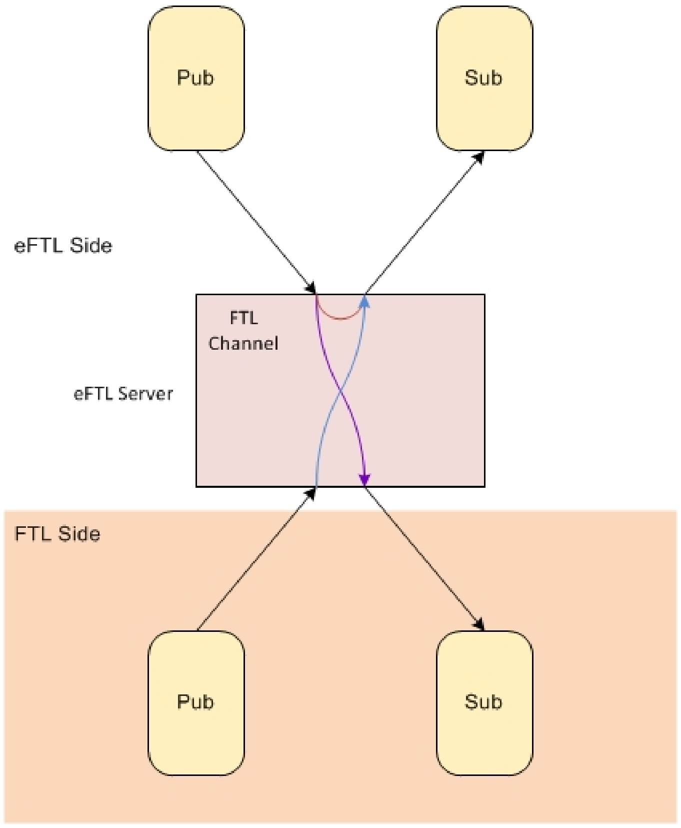
The diagram shows three eFTL applications: one that publishes and two that subscribe. These eFTL applications connect to a TIBCO eFTL server with one channel, which forwards messages among the applications. The red arrows indicate *forwarding within the eFTL side*: that is, *from eFTL* publishers *to eFTL* subscribers, through an FTL channel.

If an eFTL application both publishes and subscribes, it does not receive messages from itself through the FTL channel.



Messages between eFTL Clients and FTL Clients

An FTL channel can forward messages between eFTL applications and FTL applications.



The diagram shows applications that publish and subscribe. Applications at the top connect to the FTL channel using the TIBCO eFTL API, while those at the bottom communicate using TIBCO FTL transports. The channel forwards messages among them in three ways:

- The purple downward arrow indicates *forwarding into the FTL side*: that is, *from eFTL* publishers *to FTL* subscribers.
- The blue upward arrow indicates *forwarding into the eFTL side*: that is, *from FTL* publishers *to eFTL* subscribers.
- The red arrow indicates *forwarding within the eFTL side*: that is, *from eFTL* publishers *to eFTL* subscribers, through the FTL channel.

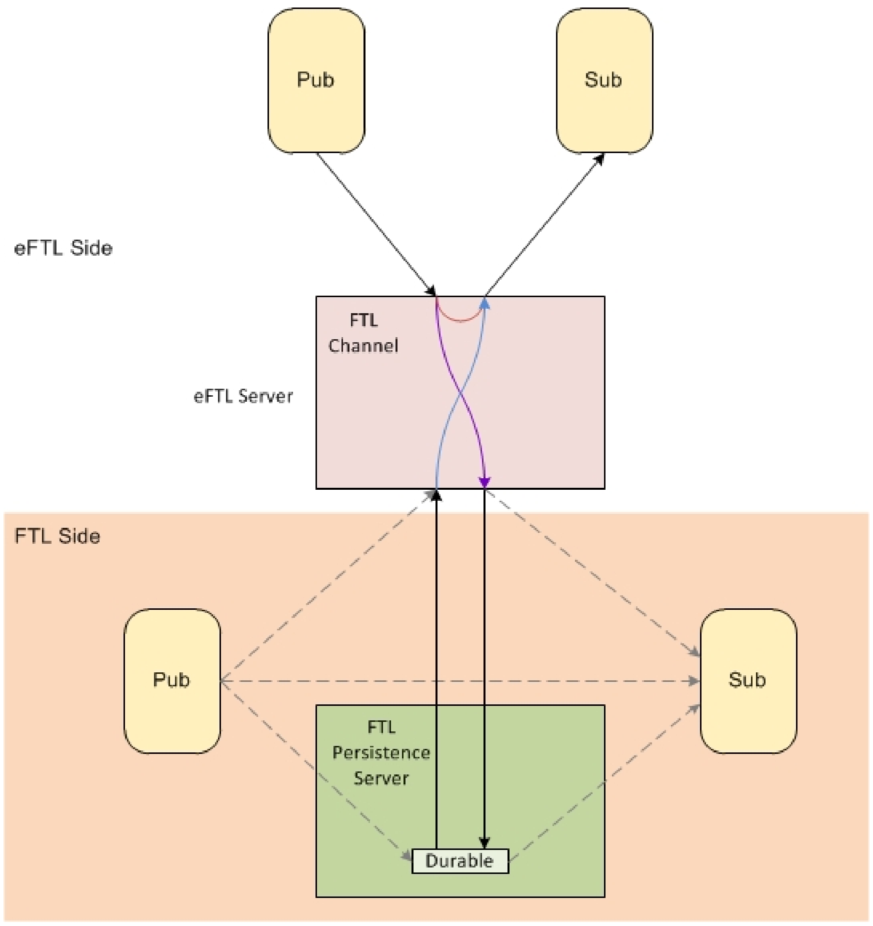
When forwarding a message, the eFTL server translates the message format appropriately; for details, see Message Format TIBCO eFTL and TIBCO FTL.

Notice that when forwarding within the eFTL side, the channel translates the message twice: as it enters and leaves the server.

FTL Persistence

An FTL channel can interact with an FTL persistence server.

The diagram adds an FTL persistence server to the previous diagram.



Gray dashed arrows in the diagram indicate that this arrangement does not depend on FTL publishers and subscribers. They need not be present. That is, eFTL applications can use the FTL persistence feature even if they do not communicate with other FTL applications.

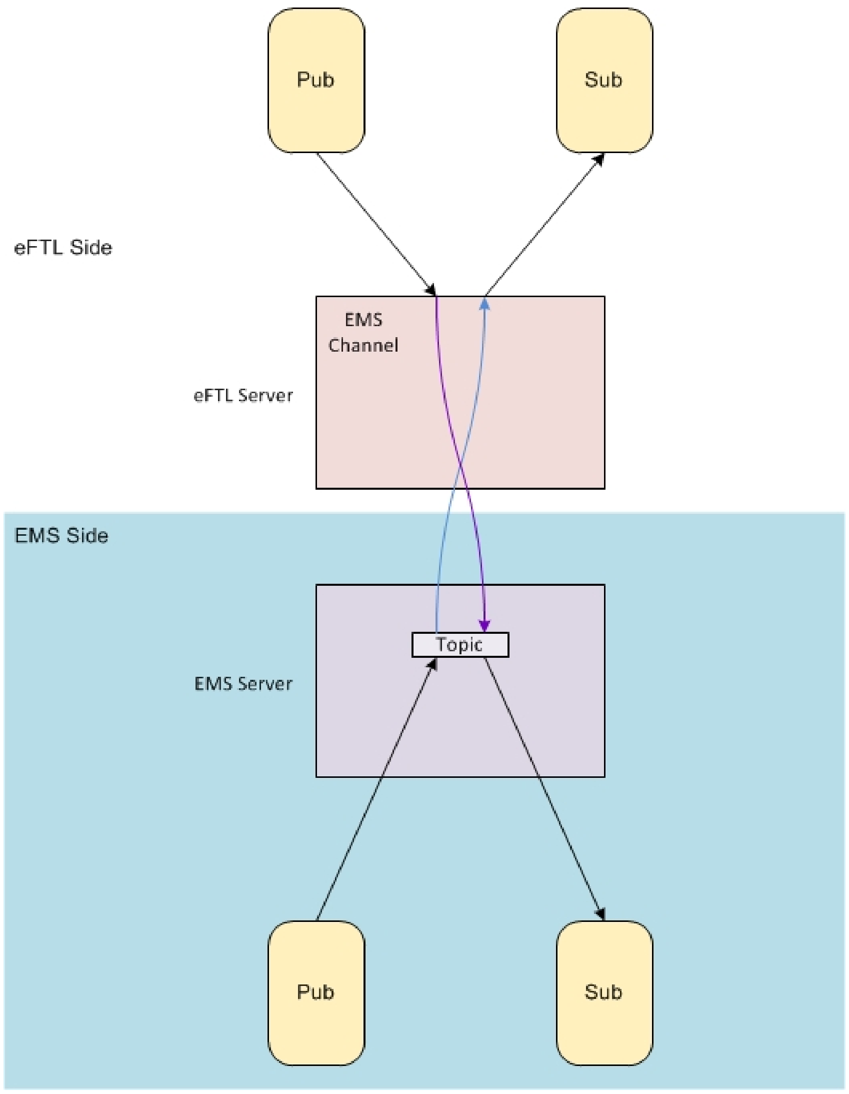
Administrators cooperate to configure persistence.

- The FTL administrator configures persistence and runs the persistence server processes.
- The eFTL administrator associates the channel's application-facing endpoint with an FTL persistence store. (For information about application-facing endpoints, see FTL Channels.)

Messages between eFTL Clients and EMS Clients

An EMS channel can forward messages between eFTL applications and a TIBCO Enterprise Message Service server, which in turn provides access to EMS applications.

The diagram shows applications that publish and subscribe. Applications at the top connect to the eFTL server, while those at the bottom connect to the EMS server. The eFTL server connects to the EMS server as a client.



The EMS channel forwards messages in two ways:

- The purple downward arrow indicates *forwarding into the EMS side*: that is, *from eFTL* publishers *to EMS* topics in the EMS server.

  EMS applications that subscribe to a topic receive its messages. The EMS server can also store messages for persistence (optional) using durables.

- The blue upward arrow indicates *forwarding into the eFTL side*: that is, *from EMS* topics *to eFTL* subscribers.

  The EMS channel subscribes to EMS topics on behalf of eFTL subscribers.

Topics Only

The eFTL server can access EMS topics, but *not* EMS queues.

Translation

When an EMS channel forwards a message, the eFTL server translates the message format appropriately; for details, see Message Translation TIBCO eFTL and TIBCO EMS.

Round-Trip Forwarding

All messages that flow through an EMS channel must pass through the EMS server, even messages that travel *from eFTL* publishers *to eFTL* subscribers. The EMS channel forwards messages to the EMS server, which delivers them back to the channel, which in turn forwards them back to the eFTL side.

The channel translates such messages twice: once in each direction, to and from the EMS server.

Channels

Within a TIBCO eFTL server, administrators can define several named *channels*, each of which carries a separate message stream. Administrators can use channels to isolate message streams from one another.

An eFTL application specifies a specific channel name when it connects to a TIBCO eFTL server.

Channel Type

A channel can use one of two communication infrastructures: either TIBCO FTL or TIBCO Enterprise Message Service, but not both. The *type* of the channel denotes this infrastructure: each channel is either an FTL channel or an EMS channel.

If eFTL applications on the channel must communicate with either FTL or EMS applications, then this constraint forces the choice of channel type. In contrast, if the channel carries messages *only* among eFTL clients, then the administrator can choose either channel type based on other considerations.

eFTL Side

Each channel has an eFTL side.

Each channel also has either an FTL side or an EMS side, depending on the channel's type.

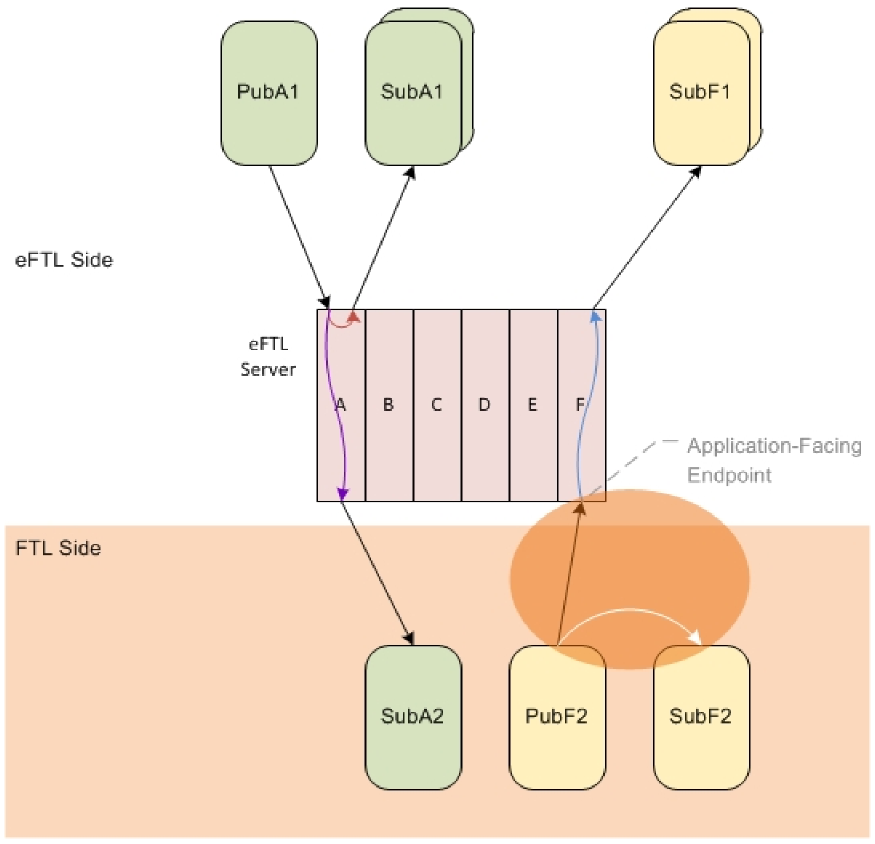
FTL Channels

An FTL channel provides publish-subscribe messaging among eFTL applications, *and* between eFTL applications and FTL applications.

Each FTL channel has an *application-facing endpoint*, which is its interface to the FTL messaging infrastructure. The channel uses this endpoint to forward messages to and from FTL application processes, including other eFTL servers within a cluster.

Administrators can implement each channel's application-facing endpoint with transports, which tie it to the FTL infrastructure.

In the diagram, channels A and F are FTL channels.



Messages from eFTL Publishers

When PubA1 sends a message, the server forwards it in two ways. Red arrows indicate forwarding within the eFTL side to all the other eFTL subscribers on channel A. The purple arrow within channel A indicates forwarding into the FTL side.

Messages from FTL Publishers

In the opposite direction, the blue arrow indicates forwarding into the eFTL side on channel F. When PubF2 sends a message, channel F forwards the message to all its eFTL subscribers.

However, channel F does not forward the message downward again over any transports attached to its application-facing endpoint: that responsibility belongs to transports within the FTL application infrastructure. SubF2 receives the message only if an FTL transport carries messages from PubF2 to SubF2. In this example, the orange ellipse indicates a dynamic TCP transport implementing the application-facing endpoint of channel F, and PubF2 and SubF2 both share that transport. The white arrow indicates that the shared transport carries the message to SubF2.

Isolation

Administrators can use separate channels to isolate message streams from one another. When FTL transports are properly configured, messages in channel A remain within channel A. Similarly for channel F, and all the other channels. Each channel gives rise to a separate message network. The diagram shows channel A's network in green, and channel F's network in yellow.

Nonetheless, if you configure FTL transports that carry messages between the application-facing endpoints of two channels, you can create opportunities for crosstalk between their message streams.

Types of Forwarding

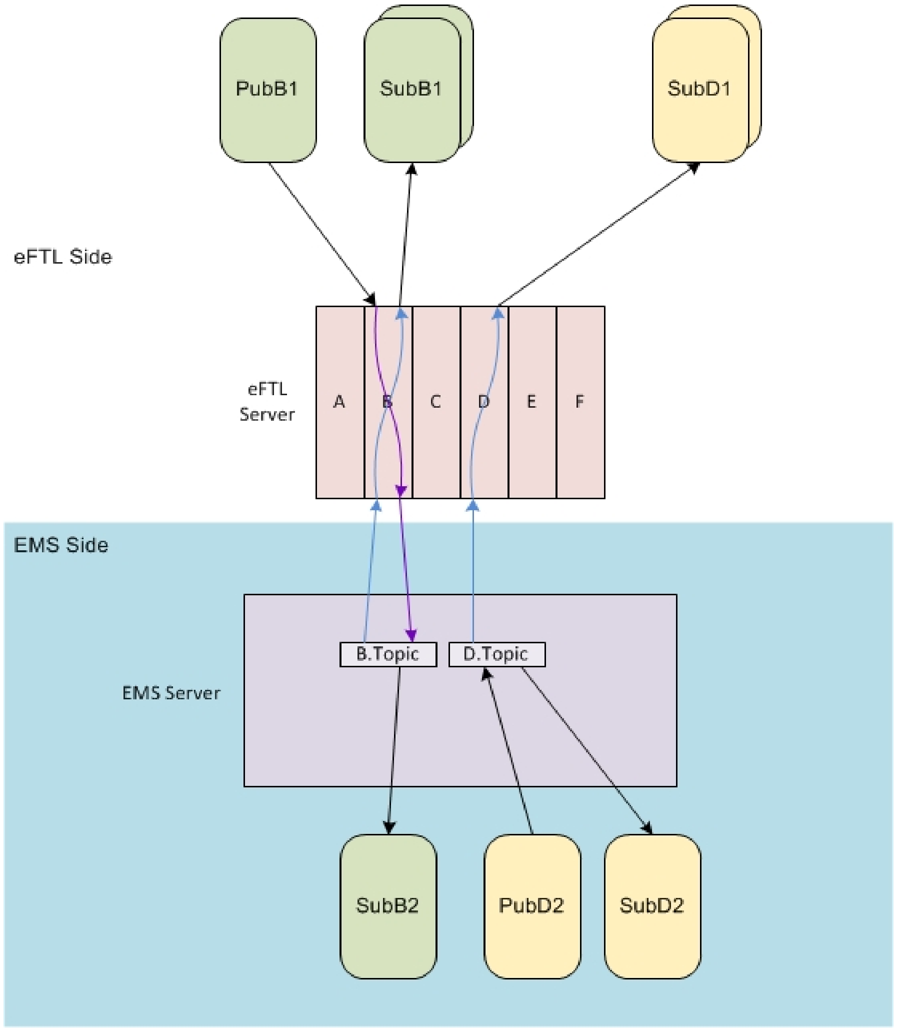
Although the diagram specifically highlights particular types of forwarding for channels A and F, every FTL channel is capable of all three types of forwarding. That is, channel A can forward into the eFTL side, even though the diagram does not indicate this path. Similarly, channel F can forward within the eFTL side and into the FTL side, even though the diagram does not indicate these paths.

EMS Channels

An EMS channel provides publish-subscribe messaging among eFTL applications, *and* between eFTL applications and EMS applications.

Each EMS channel is a client of an EMS server, which acts as a store-and-forward intermediary for all messages through the channel.

In the diagram, channels B and D are EMS channels.



Messages from eFTL Publishers

The purple arrow within channel B indicates forwarding into the EMS side. When PubB1 sends a message, channel B translates it and publishes it to a topic within the EMS server.

The EMS server forwards the message in two ways:

- The downward black arrow indicates ordinary EMS delivery to subscriber SubB2.

- The upward blue arrow indicates the two phases that carry the message into the eFTL side:

  1. The EMS server delivers the message to channel B.
  2. Channel B retranslates the message and forwards it to all its eFTL subscribers, represented by SubB1 in the diagram.

Messages from EMS Publishers

The diagram also traces the path of a message from EMS publisher PubD2 to its topic in the EMS server.

- The downward black arrow indicates ordinary EMS delivery to subscriber SubD2.
- The upward blue arrow indicates the two phases that carry the message into the eFTL side:

  1. The EMS server forwards the message to channel D.
  2. Channel D translates the message, and delivers it to all the eFTL subscribers on channel D, represented by SubD1 in the diagram.

Isolation

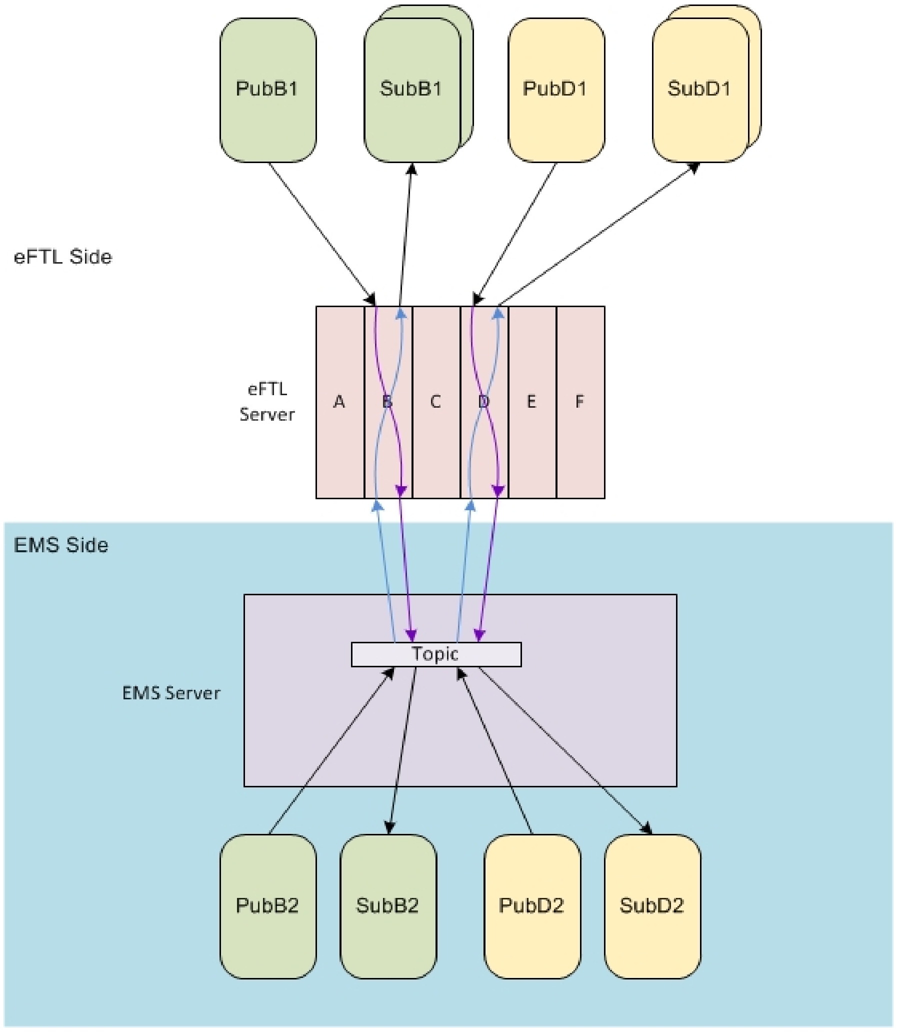
Administrators can use separate channels to isolate message streams from one another. With proper configuration, messages in channel B remain within channel B. Similarly for channel D, and all the other channels. Each channel gives rise to a separate message network. The diagram shows channel B's network in green, and channel D's network in yellow.

Nonetheless, if subscribers on two channels access the same EMS topic, the EMS server can merge the two message streams. For more information about EMS crosstalk and an administrative remedy, see EMS Topic Prefix.

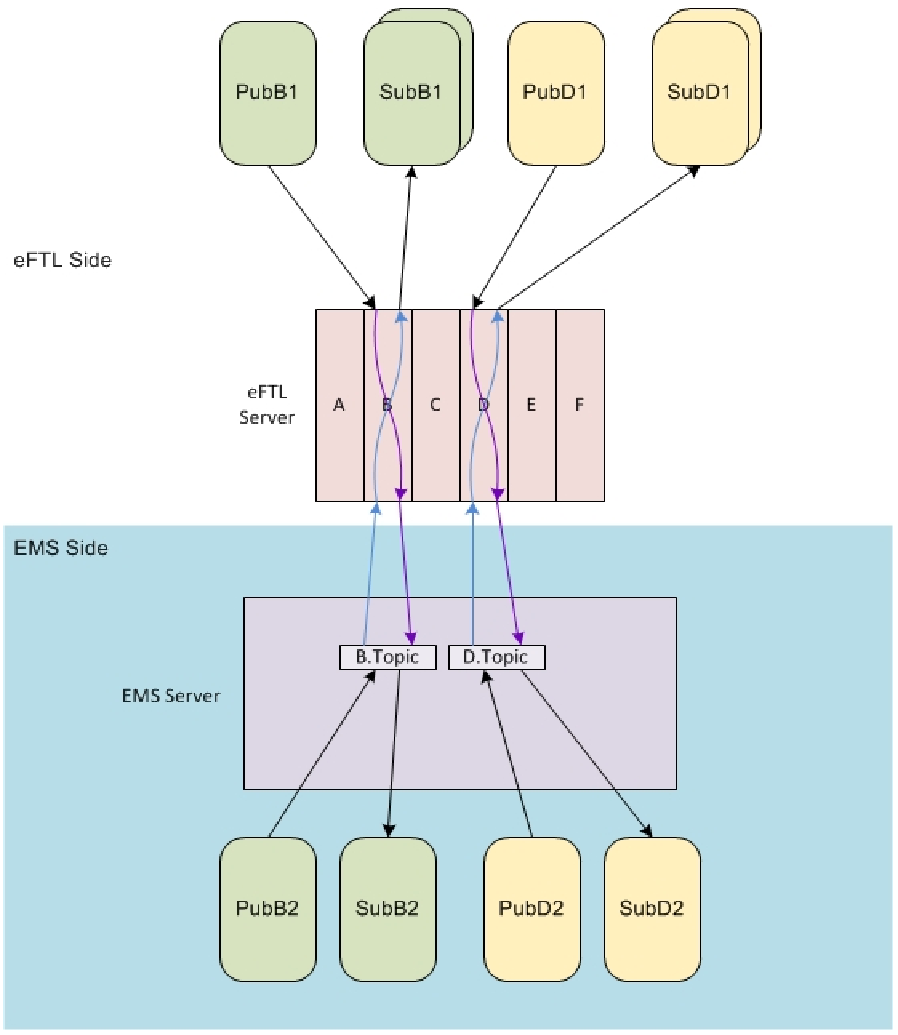
EMS Topic Prefix

Channels isolate message streams. However, the EMS server merges message streams that share a topic name. Configuring a distinct topic prefix on each EMS channel can prevent crosstalk among channels through the EMS server.

The first diagram illustrates the undesirable crosstalk effect. The intent is that channels B and D carry separate message streams, isolating the green applications on channel B from the yellow applications on channel D. However, because applications on the two channels use the same topic name, the EMS server forwards messages on both channels, resulting in crosstalk. (The effect is the same even in the absence of EMS applications that use the shared topic name.)



To prevent crosstalk, the eFTL administrator can supply a distinct prefix string for each EMS channel. Each EMS channel prepends its prefix string to EMS topic names at each publish and subscribe operation. The second diagram illustrates the result: each channel communicates with a separate topic in the EMS server, effectively isolating their message streams.



Topic prefix strings are transparent to eFTL client applications. That is, applications on the eFTL side still publish and subscribe using a topic name *without* the prefix: the channel prepends it only on the EMS side.

However, topic prefix strings are *not* transparent to EMS client applications. That is, applications on the EMS side must explicitly include the prefix when they publish and subscribe.

Furthermore, EMS administrators must explicitly allow these prefix strings in topic names.

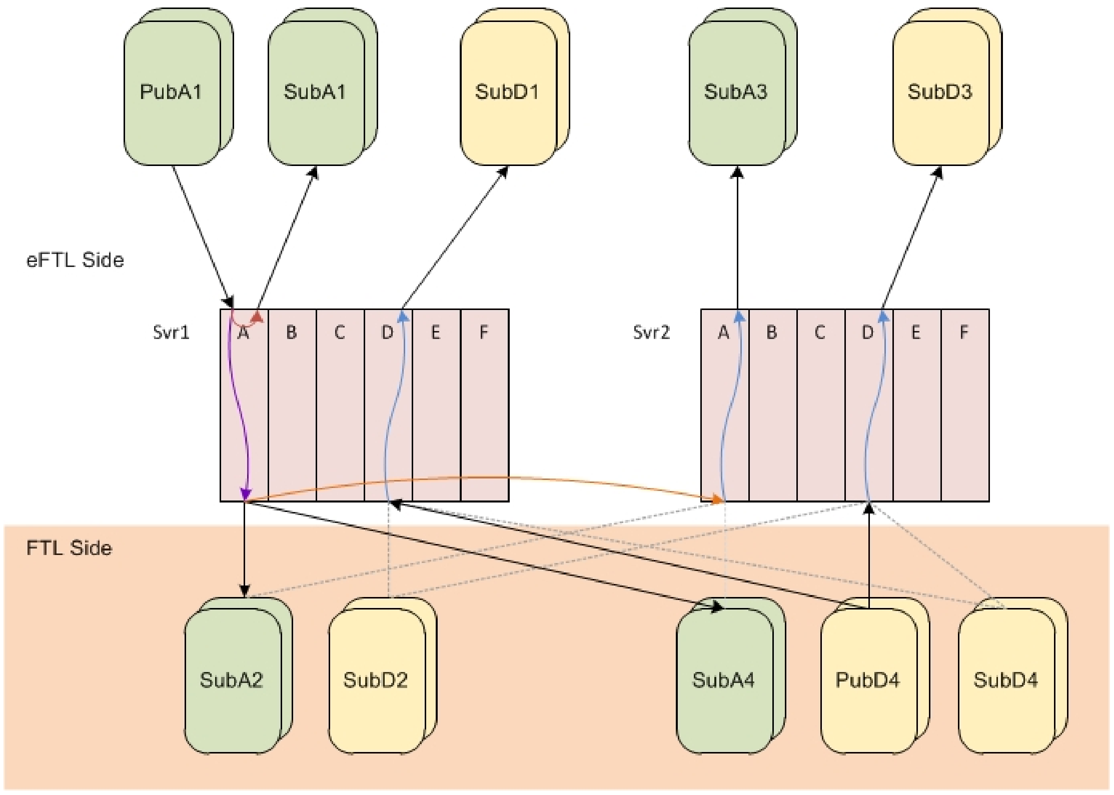
eFTL Cluster

Administrators can arrange an *eFTL cluster*: several TIBCO eFTL server processes that cooperate as a single unified service. With a cluster you can scale the service's capacity for client connections and bandwidth beyond the capacity of a single server process on current hardware.

All the server processes in the cluster share the same configuration, which they receive from the realm server.

FTL Channels in a Cluster

When eFTL servers cooperate in a cluster, each FTL channel ensures that all the eFTL clients of each channel can access the entire message stream of the channel. A message travels one of two possible paths, depending on its origin.



Messages Originating in the eFTL Side

The diagram illustrates a cluster of two eFTL server processes. When PubA1 sends a message, Svr1 forwards the message in three ways. The first two types of forwarding are already familiar from preceding topics:

- The red arrow indicates forwarding within the eFTL side: that is, to all the other eFTL clients of Svr1 that subscribe on channel A, represented by SubA1.
- The purple arrow indicates forwarding into the FTL side: more precisely, to the application-facing endpoint of channel A. From there, the FTL transports that implement that endpoint carry the message stream to SubA2 and SubA4.

The third type of forwarding is unique to FTL channels in an eFTL cluster. The orange arrow indicates *forwarding across a cluster*: this type of forwarding carries the eFTL message stream through the channel's application-facing endpoint, to the corresponding channel of Svr2, which represents all the other server processes. When these messages arrive at Svr2, it forwards them into its eFTL side (blue arrow within channel A), to SubA3, which represents all the eFTL clients of Svr2 that subscribe on channel A.

Forwarding across a cluster carries only messages that originate in eFTL clients: it does *not* carry messages that originate in FTL applications.

Each message flows into the FTL side through *only one* server within the cluster: the server that receives the message from the originating eFTL client. For example, notice that Svr2 does *not* forward messages from Svr1 downward to SubA4, which would result in duplicate delivery paths.

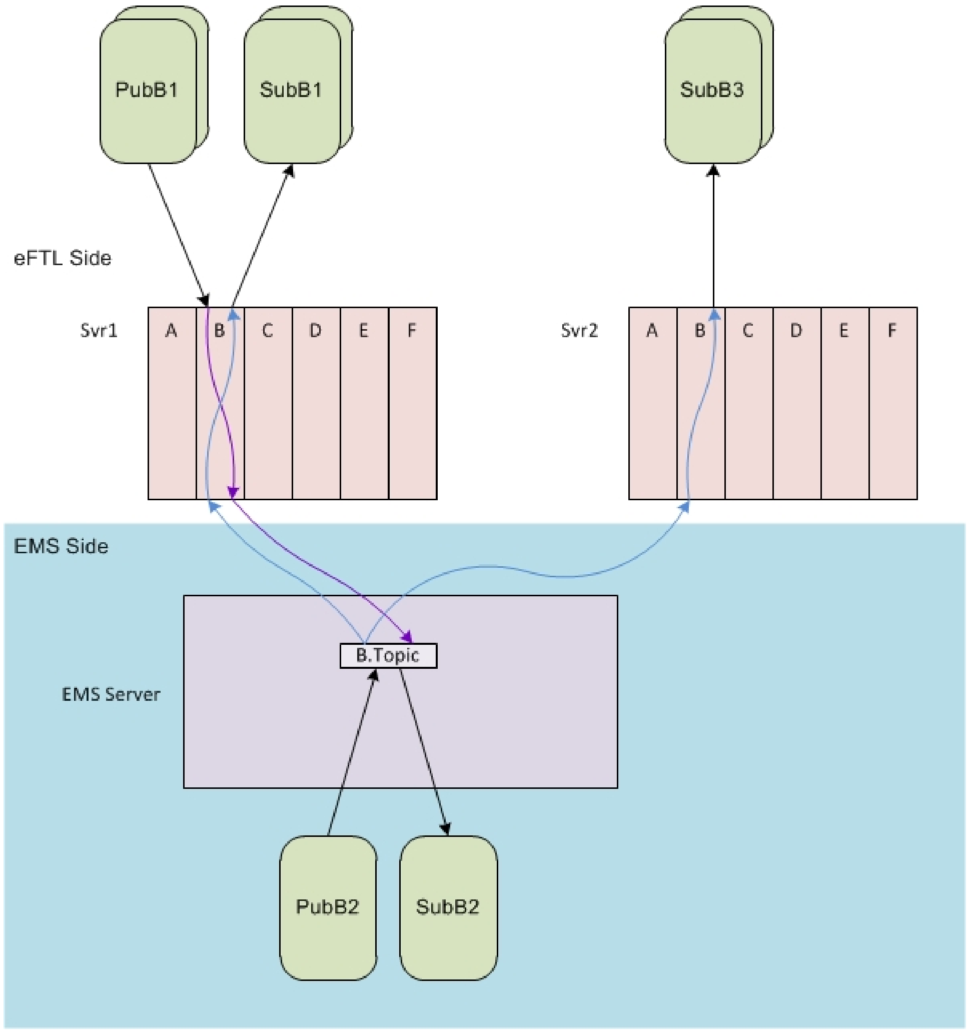
Messages Originating in the FTL Side

In the opposite direction, each eFTL server in the cluster applies only one type of forwarding. When FTL publisher PubD4 sends a message, FTL transports carry the message to FTL subscribers, and to all the servers in the cluster. Blue arrows in channel D indicate forwarding into the eFTL side: through Svr1 to SubD1, and through Svr2 to SubD3, that is, to all of the subscribing eFTL clients on channel D of the two servers.

Notice that FTL transports connect each FTL application to the appropriate application-facing endpoint at *every* eFTL server. In the lower part of the diagram, solid arrows indicate messages flowing over transports, while gray dashed lines indicate transports that are present, even though they did not carry messages in this example.

EMS Channels in a Cluster

When eFTL servers cooperate in a cluster, their corresponding EMS channels do not forward messages across the cluster. Instead the EMS server forwards messages to the corresponding channels in each server.



The diagram illustrates a cluster of two eFTL server processes. The purple arrow indicates that when PubB1 sends a message, Svr1 translates it and forwards it the EMS server. The EMS server forwards it in two directions:

- The downward black arrow indicates delivery to EMS subscribers, such as SubB2.
- The upward blue arrows indicate delivery to channel B of *both* eFTL servers, which translate the message again, and forward it to eFTL subscribers, such as SubB1 and SubB3.

When EMS publisher PubB2 sends a message, the EMS server forwards it over same upward paths.

Message Format: TIBCO eFTL and TIBCO FTL

Although TIBCO eFTL and TIBCO FTL message formats are similar, these products encode messages for transmission using different wire formats. When forwarding a message in either direction between these frameworks, the TIBCO eFTL server converts each message appropriately.

For each FTL channel, administrators can configure an optional *FTL exchange format*, which governs conversion when accepting a message from the eFTL side:

- When an exchange format is present, an FTL channel converts eFTL messages to the exchange format.

  Furthermore, it forwards an eFTL message *only* if the message *matches* the exchange format: that is, the data fields of the message are a subset of the fields defined in the exchange format.

  If a message does not match the exchange format, the channel discards the message. The message does not travel through the channel at all: neither to the FTL side, nor back to the eFTL side.

  However, exchange format does *not* affect forwarding into the eFTL side. That is, the TIBCO eFTL server forwards *all* messages from FTL publishers to eFTL subscribers.
- When an exchange format is *not* present, an FTL channel converts eFTL messages to self-describing, that is, dynamic format, FTL messages.

  Furthermore, it forwards *all* messages from eFTL publishers to FTL subscribers.

See Also

To configure the FTL Exchange Format parameter, see "eFTL Clusters Grid” in *TIBCO eFTL Administration*.

Message Translation: TIBCO eFTL and TIBCO EMS

TIBCO eFTL software and TIBCO Enterprise Message Service software differ in the form of messages and in their subscription paradigm. To account for these differences, the eFTL server translates messages as they pass through its EMS side. These differences can affect developers of eFTL applications.

Content Matchers, Topic Names, and the _dest Field

TIBCO Enterprise Message Service software delivers messages based on topic names: publishers send each message to a specific topic, and subscribers receive the messages sent to a topic.

In contrast, TIBCO eFTL software delivers messages based on the content of named fields: a subscriber specifies a content matcher, and receives messages with matching fields and values.

To resolve this difference, the eFTL server translates according to the following table.

Translation
When an EMS channel forwards an EMS message into the eFTL side, it sets the `_dest` field of the eFTL message to the topic name of the EMS message.
When an eFTL client creates a subscription with a content matcher that matches against the `_dest` field, the EMS channel creates an EMS subscription to the corresponding topic. EMS subscriptions can use the asterisk (*) and greater than (>) characters as wildcard elements. If an eFTL subscriber includes these elements in a content matcher, they become part of the EMS subscription, and the EMS server interprets them correctly.
When an EMS channel forwards an eFTL message into the EMS side, and the message contains a `_dest` field, the channel publishes the EMS message to the topic corresponding to value of the `_dest` field.
When an EMS channel attempts to forward an eFTL message into the EMS side, but the message does not contain a `_dest` field, or its `_dest` value is not a valid topic name, the channel discards the message, and does not publish it into the EMS side. The eFTL publisher receives an error callback.

Translation from the EMS Side into the eFTL Side

The eFTL server translates JMS message types according to the following body type table.

JMS Body Type	Translation
Message	Translates to an empty eFTL message: that is, its only field is `_dest`.
TextMessage	The `_text` field contains the text string data.
MapMessage	Fields and their values translate according to the following field type table.
Any other body type.	The eFTL server discards messages with any other JMS body type.

The eFTL server translates JMS fields according to the following field type table.

JMS Field Type	Translation
char, byte, int, long	long
float	double
char*	string
Array of fixed point numbers	Array of long
Array of floating point numbers	Array of double
bytes	The eFTL server discards byte array fields, omitting them from the translated message.

Translation from the eFTL Side into the EMS Side

All eFTL messages translate into EMS map messages. The eFTL server translates eFTL fields according to the following field type table.

EMS administrators do not configure translation in the EMS server.

eFTL Field Type	Translation
long	tibems_long
double	tibems_double
string	char*
long[]	Array of tibems_long
double[]	Array of tibems_double
message	MapMessage
datetime	A DateTime field translates into a nested MapMessage with two fields: • `s` represents whole seconds in a signed 64-bit integer. Zero denotes the UNIX epoch: midnight entering January 1, 1970, UTC. • `n` represents nanoseconds after the time that the s field denotes. Although the data structure stores this value in a signed 64-bit integer, this component is always non-negative, between zero and 999,999,999 (inclusive). The EMS field name is the prefix `_dateTime:` concatenated with the FTL field name.

eFTL Field Type	Translation
Array of string values	A field containing an array of these types translates into a nested MapMessage. The EMS field name is the prefix `_stringArray:` concatenated with the FTL field name. The fields of the nested MapMessage contain the values of the array, and each field name is a string that denotes the corresponding array index: for example, 0, 1, 2, 3, and so on.
Array of message values	A field containing an array of these types translates into a nested MapMessage. The EMS field name is the prefix `_msgArray:` concatenated with the FTL field name. The fields of the nested MapMessage contain the values of the array, and each field name is a string that denotes the corresponding array index: for example, 0, 1, 2, 3, and so on.
Array of datetime values	A field containing an array of these types translates into a nested MapMessage. The EMS field name is the prefix `_dateTimeArray:` concatenated with the FTL field name. The fields of the nested MapMessage contain the values of the array, and each field name is a string that denotes the corresponding array index: for example, 0, 1, 2, 3, and so on.

Message Fields

Client ID Field

The TIBCO eFTL server can automatically append the client ID field to the messages that clients publish.

The value of the `_client_id` field is the client identifier supplied or assigned when the client connected to the eFTL server. See the `CLIENT_ID` property in the API documentation.

Back-end FTL and EMS applications can use this value to direct responses to that specific client. For example, the content matcher of the eFTL client's subscription could match for this value.

To enable the server to append this field, see "Messages" in the topic "Server Command Line Reference" in *TIBCO eFTL Administration*.

User Field

The TIBCO eFTL server can automatically append the user field to the messages that clients publish.

The value of the `_user` field is the user name that the client supplied to authenticate itself to the eFTL server. See the `USERNAME` property in the API documentation.

Back-end FTL and EMS applications can use this authenticated user name to authorize requests. (Back-end request authorization is separate from eFTL server authorization for publish and subscribe groups.)

To enable the server to append this field, see "Messages" in the topic "Server Command Line Reference" in *TIBCO eFTL Administration*.

Automatic Reconnect

When a temporary network outage disconnects an eFTL client from the server, the client library automatically attempts to reconnect to the server.

This feature is transparent to the client application and to the developer.

During the disconnect, the client library buffers messages that the application publishes. Upon reconnect, the client library transmits those messages to the server.

Similarly, during the disconnect, the server buffers outbound messages to the client's subscribers. Upon reconnect, the server transmits those messages.

Reconnection attempts continue for approximately half a minute. If automatic reconnect fails, then both client and server discard buffered messages, and the application's disconnect callback determines its response to the longer network outage.