TIBCO FTL®

Release Notes

Version 6.10.1 | July 2023

Contents

About this Product

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

The 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. 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.

Product Editions

TIBCO Messaging is available in a community edition and an enterprise edition.

TIBCO Messaging - Community Edition is ideal for getting started with TIBCO Messaging, for implementing application projects (including proof of concept efforts), 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 Messaging expands.

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

- Users may run up to 100 application instances or 1000 web/mobile instances in a production environment.
- Users do not have access to TIBCO Support, but you can use TIBCO Community as a resource (community.tibco.com).

TIBCO FTL in the Community Edition has the following additional limitations and exclusions:

- Excludes transport bridges
- Excludes the RDMA transport protocol
- Excludes disaster recovery features
- Excludes customizable dashboards and monitoring gateway

TIBCO Messaging - 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, as well as access to TIBCO Support.

New Features

The following new features were added to recent releases of TIBCO FTL® software:

6.10.1

OpenSSL 3.0.9 Support

TIBCO FTL® now supports OpenSSL 3.0.9.

6.10.0

max.disk.fraction Available on the Windows Platform

The FTL Server configuration parameter, max.disk.fraction, is available on the Windows platform. It monitors disk capacity to prevent a disk full state.

For details, see the FTL Administration guide, Persistence Architecture, Handling Persistence Service Disk Capacity and Persistence Service Configuration Parameters. Also see the FTL Monitoring guide, Catalog of Persistence Metrics.

Disk Compaction with Persistence Service Running (Online)

The persistence service can compact its disk persistence files while running without interruption to active publishers or subscribers. Compaction can be started manually or automatically.

For details, see the FTL Administration guide:

- Persistence Service Disk Capacity
- Compact Disk Persistence Files with Persistence Service Online

Also see compact_online in the FTL Administration guide, FTL Administration Utility.

Disk Compaction with Persistence Service Not Running (Offline)

The FTL administration utility , tibftladmin, may be used to compact disk persistence files for a given persistence service when that persistence service is not running.
For details, see the FTL Administration guide, Compact Disk Persistence Files with Persistence Service Offline.

Also see compact_offline in the FTL Administration guide, FTL Administration Utility.

Backup and Restore with the FTL Administration Utility (tibftladmin)

Administrators can now use the FTL administration utility, tibftladmin, to perform disk persistence backup and realm backup. Both backups can be run while the servers are running without interruption to clients. See the following topics in the FTL Administration guide:

- Disk Persistence Backup and Restore
- Realm Administration Tools
- FTL Administration Utility

New Realm Property

You can set a new realm property enable_format_serialization_optimization. Set the value on when all the clients and servers are FTL 6.10 or greater. The default is off. See POST realm/properties.

Additions to Go

The Go API includes new message access methods (Get/Set), a new message iterator, and a way to read/write messages with byte arrays.

See the Go API. To open the Go documentation, see the readme.txt file in /samples/src/golang.

Log Message from Failed or Successful Logins

FTL now logs a message from failed or successful logins at the tls:verbose level.

Improved Logging for Message TTL

The persistence service now logs message expiration counts per durable, for up to 16 durables at once.

Show the Count of Expired Messages

The number of message expiration events in a given persistence store is reported by a monitoring metric and the REST API. The number of messages expired in a given durable is reported by the REST API.

For details, see the FTL Administration guide:

- Catalog of Persistence Metrics
- GET persistence/clusters/<clus_name>/stores/<stor_name>

- GET persistence/clusters/<clus_name>/stores/<stor_name>/durables/<dur_name>

Message Limit is added to the Cluster Details Panel

The parameter Message Limit limits the number of messages that can be held by the cluster. It is on the Cluster Details Panel.
For details, see the FTL Administration guide, Cluster Details Panel.

Non-Inline Send Policy for Persistent Messages

Applications can now use the non-inline send policy for publishers created on an endpoint with a store. This allows the publisher to send messages to the persistence service in the background, potentially improving throughput.

See the following FTL guides:

- Administration guide, Publisher Mode, section "Publisher Mode and Send Policy (Administrators)"
- Development guide, Publisher Mode and Send Policy

Purging System Stores and Durables in System Stores

You can purge system stores (such as the monitoring and logging stores) and durables in system stores, such as the monitoring or logging stores. For details, see the FTL Administration guide, Persistence Stores Status Table and Durables List.

Timeout Configuration for Remote Authentication Service

Added the ability to configure a timeout when FTL server makes a call to a remote authentication service by setting auth.timeout in the FTL server YAML file, globals section. See the FTL Administration guide, FTL Server Configuration Parameters.

Persistent Cluster Properties

You can set inter-cluster heartbeat and timeout from the Cluster Details Panel.

See the FTL Administration guide, Cluster Details Panel.

C and Go: API Detail String Retrieval

For C and Go, an API is available to retrieve the detail string from an FTL exception.

In the Go API, calling Error() on the FTL exception now returns the detail string, if available, rather than the summary string.

TLS 1.3 Support

FTL supports TLS 1.3. See the FTL Administration guide, Securing FTL Servers.

Persistence Service Warning

With store forwarding enabled, the persistence service issues potential data loss warnings after reestablishing a route. The warnings have been reduced to meaningful warnings only.

OpenSSL 3.0.8 Support

FTL now supports OpenSSL 3.0.8. For details, see the FTL 6.10.0 Release Notes, OpenSSL V1.1.1 to 3.0 Migration.

6.9.1

OpenSSL 3.0.7 Support

FTL now supports OpenSSL 3.0.7. See OpenSSL V1.1.1 to 3.0 Migration.

6.9.0

OpenSSL 3.0.5 Support

FTL now supports OpenSSL 3.0.5. See OpenSSL V1.1.1 to 3.0 Migration.

Browse Messages in a Shared Durable

Use a client API to browse messages stored in a shared durable, for example, to run analytics on the data or delete a message. Messages are browsed from oldest to newest. A matcher can be used to browse a subset of the messages.

For details, see the FTL Administration guide:

- Shared Durables
- Browsers List

Also see the FTL API Documentation.

Weights are Supported in the FTL Bridge Service

Weights can be specified in the FTL bridge service to coordinate its fault-tolerant operations to set a precedence order among all the bridge service instances.

For details, see the FTL Administration guide, Bridge Service Configuration Parameters.

Handling Persistence Service Disk Capacity

The FTL Server configuration parameter, max.disk.fraction, monitors disk capacity to prevent a disk full state. This keeps the persistence cluster running in the event that disk space is not available. The max.disk.fraction parameter is not a replacement for

proper storage limits (byte limit and message limit). It is instead a safeguard against an unplanned lack of disk space.

For FTL 6.9.0, this feature is only available on Linux platforms.

For details, see the FTL Administration guide, Persistence Architecture, Handling Persistence Service Disk Capacity and Persistence Service Configuration Parameters. Also see the FTL Monitoring guide, Catalog of Persistence Metrics.

Async Disk Persistence is Supported for Maps

Async disk persistence is now supported for maps. Calls to set or remove keys from a map may return before the set or remove has been written to disk by majority of the FTL servers.

For details, see the FTL Concepts guide, Disk-Based Persistence and Message Swapping. Also the FTL Administration guide, Clusters Grid and Cluster Details Panel.

Persistence Improvement to Avoid Discards When Sending to the Store

If a store's publisher mode is store_confirm_send, the FTL client library automatically prevents discards when sending extremely large message batches to the persistence service.

The default backlog buffer size of the persistence client transport is increased to 256 MB.

For details, see the FTL Administration guide, Buffer and Performance Settings for Transports.

FTL Persistence Delivers Messages Without Causing Discards

Before FTL 6.9.0, the FTL persistence service always attempted to deliver messages to a consumer until the prefetch count was reached. If the total message size was large enough, sender discard would occur, forcing the consumer to reconnect.

Now the persistence service delivers fewer messages when sender discard might otherwise occur.

For details, see the FTL Administration guide, Durable Prefetch Count. Also see the FTL Development guide, Maximum Message Size.

REST Commands to Manage Disaster Recovery Servers

REST commands can be used to automate some portions of the DR procedure.

For details, see Securing FTL Servers, Trust File, POST cluster, and GET cluster. Also see the revised Disaster Recovery section in the Administration guide.

User Interface Identifies Subscribers and Browsers

The user interface differentiates between subscribers on a shared durable and browsers on a shared durable. For details, see the FTL Administration guide, Durables List.

User Interface Supports Permissions

The user interface supports creating permissions for users and roles on different persistence related objects (such as stores and clusters). For details, see the FTL Administration guide, Configuring Permissions.

FTL Cumulative Connection Count

Each FTL server now accumulates a count of all successful FTL client or REST API connections over the lifetime of the FTL server process. This is reported by these GET server and GET cluster fields:

- cumulative_client_count
- cumulative_total_client_count (includes internal services as well as user clients)

For details, see GET server and GET cluster.

Log Service Adds Parameter for Database Name

tiblogsvc takes a parameter for the database name (such as -influx-database). For details, see FTL Monitoring guide, Log Service Command Line Reference (tiblogsvc).

Log Improvements

When multiple services in an FTL server are configured to use the same log file, log messages from each service are identified by prefixing the service's name to the message. In addition, log rotation is now implemented for this file.

For details, see the FTL Administration guide, Realm Service Configuration Parameters.

Quorum Member Log

The persistence service now logs whenever it times out a member due to missing heartbeats.

6.8.0

Logical OR Content Matcher

In addition to supporting logical AND content matching, FTL now also supports logical OR content matching, whereby a subscriber can express an interest in multiple values. This eliminates the need to, in some cases, create multiple subscribers. For information

on using logical OR matchers, see the FTL Development guide, "Content Matchers" and "Match String Syntax" topics.

Weights for Group Members in FTL Group API

An application can assign a weight to a fault tolerance group member process. The weight influences the FTL group service's assignment of the ordinal, which determines the operating role, such as 1 (active). Use weights to make decisions based on hardware speed, hardware reliability, load factors, or other operating environment factors. Programs can adjust weights based on system conditions.

This feature is similar to the Fault Tolerance Groups feature in TIBCO Rendezvous.

For information on assigning weights, see the FTL Development guide, "Weights for Group Members".

User Permissions

Administrators can secure the messaging infrastructure by setting permissions at the cluster and store level to restrict what actions users can take and what data users can receive. To enable permissions, set the enable_permissions flag in the realm properties to true.

TLS without Authentication is deprecated and is going to be removed in a subsequent FTL release.

For information, including migration preferences, see the FTL Administration guide, topics: "Permissions", "Required Permissions for API Calls", and "Migrating to FTL 6.8.0 when Using Permissions".

Service Connection Policy for Reduced Latency and Cost

An administrator can choose a Service Connection Policy to tune connections to services that use the auto transport. For information on the Service Connection Policy configuration, see the FTL Administration guide, "Realm Properties Details Panel", "Service Connection Policy".

Endpoint Store Inboxes

By default in FTL 6.8.0 for new realm configurations, inbox subscribers create a durable in the endpoint's configured store, provided the endpoint has no transport configured. All inbox traffic goes through the durable in the store. Routing inbox data to one of two system inbox stores (ftl.system.inbox.store or ftl.routing.inbox.store) is still available but is not the default behavior. For information, see the FTL Administration guide, "Endpoint Store Inboxes".

New Client API Call to Get Multiple Key/Value Pairs from a Map

For all APIs (C, Java, Go, and .NET), use the client API calls tibMap_GetMultiple or tibMap_GetMultipleWithLock to get multiple values from a map. See the FTL API Documentation for details. tibRealm_RemoveMap (and equivalents for all APIs) now accepts TIB_MAP_PROPERTY_DOUBLE_PERSISTENCE_RETRY_DURATION via tibProperties. The call fails after the retry duration expires.

New Client API Call to Specify the Persistence Retry Duration for Unsubscribe

For all APIs (C, Java, Go, and .NET), use the client API call tibRealm_UnsubscribeEx to allow the caller to specify TIB_SUBSCRIBER_PROPERTY_DOUBLE_PERSISTENCE_RETRY_DURATION via tibProperties . The call fails after the retry duration expires. See the FTL API Documentation for details.

New Client API Call to Get the Number of Fields of a Message

For all APIs (C, Java, Go, and .NET), use the client API call GetFieldCount to allow the caller to return the number of fields. For static format messages, the call returns a count of the fields in the static format. For dynamic format messages, the call returns the number of fields set. See the FTL API Documentation for details.

Bytes In and Out Monitoring Counters from Persistence Service

The bytes in and out monitoring counters from a persistence service are available. For information, see the FTL Monitoring guide, "Catalog of Persistence Metrics".

Configure Byte Limit for Durables

A byte limit can be configured for a static durable, a durable template or in zone settings. See the FTL Administration guide, "Persistence Limits".

Specify a Group Name While Running tibmongateway in Active-Standby

You can specify a group name for tibmongateway, while running in an active-standby setup.

Customizable Certificate SAN

Administrators can specify custom subject alternative names (SAN) for certificates generated by FTL server. See the FTL Administration guide, "FTL Server Configuration Parameters".

Improved API Logging while Sending To and Receiving From Persistence Service

API debug logging is improved. FTL now prints log messages while sending and receiving messages to and from the persistence service.

Improved Performance in Java While Getting Fields by Name

When using Java to get fields by name, performance is improved.

Improved FTL Server Start Up Time

The FTL Server startup time is now significantly improved.

Persistence Service Log Message

Clarified persistence service log message for suspected reconnect without prior disconnect.

Improved Monitoring Scripts and Added Log Rotation

Improved monitoring scripts to reduce InfluxDB logging and added log rotation to `tibmongateway` logs.

OpenSSL 3.0.2 Support

FTL now supports OpenSSL 3.0.2.

6.7.0

Disk-Based Persistence

You can now store FTL messages and metadata to disk when disk is more readily available and cost effective than using memory. These messages and metadata can also then be automatically recovered on a full restart of the persistence cluster.

New GUI Disk Persistence Settings

The administrative GUI now has the following new options, related to the new disk-based persistence feature:

- Message Swapping Mode: Allow the persistence service to store more message data than available memory.
- Disk Persistence Mode: Enable disk-based persistence and select one of two modes, depending on performance requirements.
- Force Quorum Delay: Force a quorum after this amount of delay, even if all members are not present.

The modes are also included in monitoring statistics.

New GUI Cluster Backup Command

The administrative GUI now has the following new cluster backup command, available from the Cluster status display:

- Back up the cluster: Save the state of all persistence servers in this cluster. This option available only if disk persistence is enabled, and can be used while the cluster is running. The backup is intended for disaster recovery, and does not contain the most recent state.

New Log Level Parameter for tibftlserver

You can now specify the log level for `tibftlserver` as a command-line option.

New Group Client API Calls

In the client API, the group facility now has the following calls:

- `tibGroup_GetOrdinal()`, to get the current ordinal for a member of a group.
- `tibGroup_GetMembers()`, to get a list of all members in a group.

Format IDs Preserved

When downloading and then uploading a realm configuration, format IDs are now included in the upload, for more reliable migrations/backups.

Improved Deployment Description Format

In the Administrative GUI, deployment descriptions use a more readable format.

Improved Performance of Message Serialization and De-serialization

The FTL client API library now has a number of performance improvements during message serialization and de-serialization.

6.6.0

Swagger UI support for FTLserver web API

FTL now supports and includes Swagger UI for access to the FTL server web API REST commands.

Password Masking

Password masking by base64 encoding is now available for server and client connections.

In the client API, the tibRealm_Connect call accepts masked passwords in the appropriate password property. The password is unmasked before being sent to the realm service.

Where plain-text passwords appear in command line options or configuration files, a masked version of the password is now also accepted. Configuration files like the tibftlserver YAML file accept masked passwords where the following options are available:

- pass:<password>
- file:<password file>
- env:<password env>

New command line options for tibftladmin let you perform password masking encoding explicitly or on passwords contained within a configuration file.

Robust Monitoring Gateway

The FTL monitoring gateway (tibmongateway) now runs witn increased convenience and reliability through the following improvements:

- The tibmongateway process now runs as a service under the FTL server.
- Multiple tibmongateway processes can now collaborate in an active/standby configuration for increased fault tolerance.

Easier Auto Transport DR Configuration

Externally reachable addresses are no longer required when configuring auto transports for disaster recovery.

For Go API map Function, Simpler No-Lock Option

When using a Go-language API map function without a lock, for better code clarity you can now use a constant, ftl.NoLock, for the lock value.

FTL Client Prefetch Limit

Now client applications can set a prefetch limit. If the client sets its prefetch to a value greater than the configured value, the client's prefetch is automatically set to, and overrides, the configured value.

Full Satellite Display in GUI

In the GUI, Realm Service Status page, all satellite server URLs are now shown (leader and non-leaders).

Automatic Setup of Default Zone Address

The FTL server can now automatically set an externally reachable address for default zones in the following additional scenarios:

- where a load balancer is placed between the satellite and the primary cluster.
- where there is only one-way connectivity from the satellite to the primary.

Expired Dynamic Durable Logging Improved

When a dynamic durable expires, the log entry now includes the durable name.

Map Remove All Operation

You can now clear a map with one round-trip between client and server (rather than one round-trip per key) (tibMap_RemoveAll() in C).

Configuration Parameters for Persistence Service in the Default Cluster

In the FTL server YAML configuration file, you can now specify parameters for the default persistence service.

6.5.0

Save Persistence Cluster State Upon Shutdown

Sometimes a persistence cluster must save its state to disk before shutting down. To do this, you can use tibftladmin with a new parameter: --saveonexit <true or false>.

This new parameter works only with main commands -xc --shutdown_cluster or -x --shutdown, and when present, it appends the save state request to the main shutdown command being sent to the FTL server.

Administrative GUI Tooltips

The administrative GUI grid views now include cursor-hover tooltips.

Monitor Gateway Retention Policy Per Measurement

In the tibmongateway command you can now specify the InfluxDB retention policy to be used for each measurement (metric, event, log) measurement with the following new options:

- --influx-log-retention-policy name
- --influx-event-retention-policy name

- --influx-metric-retention-policy name

New Client API Call to Explicitly acknowledge a batch of messages

You can now use a durable subscriber call (e.g., tibSubscriber_AcknowledgeMessages in C; similar API calls have been added to java, .NET and Go clients) to acknowledge batches of messages.

New Monitoring Counters

The following new persistence monitoring counters (C) are added:

- TIB_MONITORING_TYPE_STORE_BYTE_LIMIT 1008 - the configured memory limit of a store
- TIB_MONITORING_TYPE_PERSISTENCE_SERVER 1100 - the persistence service name
- TIB_MONITORING_TYPE_PERSISTENCE_CLUSTER 1101 - the cluster name

New Client API Call to Get the FTL Server Version

You can now use a client API call (tibRealm_GetServerVersion in C) to return the version of the FTL server to which the client is connected.

New Client API Call to Get Event Queue and Map Names

You can now use a client API call (tibEventQueue_GetName, tibMap_GetName in C) to return the names of the event queue and key/value map.

New Client API Call to Get a Store-Local Message ID

You can now use a client API call (tibMessage_GetStoreLocalMessageId in C) to return a message monotonically increasing ID as assigned by the local persistence service.

New Message Dispatch API for Go Language Clients

For Go language applications, you can now invoke a dispatch-style call for messages in a event queue.

Check Realm Availability via Command Line

You can now use the `tibftladmin` command to check whether the realm service is available. (This is equivalent to the REST API call `/api/v1/available`.)

New Persistence Dashboard

The monitoring dashboards for Grafana now include a dashboard for persistence service monitoring.

Highlighting of Stores and Durables Approaching Limits

In the administrative GUI, durables and/or persistence stores that are approaching their configured limits are now highlighted.

Inbox Routing via Default Cluster

Inboxes are now able to be reached with routing durables using the default cluster.

6.4.0

Message Swapping Configuration Enhancements

For the message swapping persistence feature, you can now control additional parameters to determine when store message swapping to disk takes place, on a store, zone, or durable basis. This allows for finer tuning of the balance between performance and memory conservation.

Set Maximum Message Size in a Store

You can now configure, per store, a maximum message size, via the Administration GUI or web API. Messages exceeding this size generate an error when Store Publisher Mode is set to Store - Confirm - Send.

Custom Log Level Support for Client in Administration GUI

You can now set client log level in the Administration GUI. For example:

```
transports:debug;api:debug;msg:off
```

Wide-Area Forwarding of Last-Value Durables

Wide-area stores (routed stores) now support last-value durables.

Additional Samples

New samples (in C, Java, .NET and Go), offer different combinations of multiple subscribers and rate control.

Default Store Forwarding Between the Primary Realm Service and Satellites.

To simplify configuration, a default store-forwarding cluster routes messages between primary and satellite realm services.

Java System Property Log Level.

You can now pass property com.tibco.ftl.log.level as a Java system property.

Port Ranges for Dynamic TCP Transports

You can now set a range of ports on a dynamic TCP transport. When the transport is instantiated, it selects an unused port from the given range, and binds to that port. This applies to the listen side in a client-server dynamic TCP transport, or if the mode is set to mesh.

6.3.0

Targeted Deployments

For the Targeted Deployment feature introduced in FTL 6.2.0, FTL now has a more refined ability to skip deployments for clients that do not need it.

Throughput Improvements

Publishers now batch messages for improved throughput. A new publisher property is available to disable this behavior in favor of latency.

New Send Request and Send Reply APIs

You can now use simplified send-request and send-reply APIs to send a request message and receive a corresponding reply message .

Password Hashing

For user authentication based on text-file user databases and URLs of type file://, security is enhanced via UNIX-style password hashing. Hash modes are added to auth.url configuration entry.

Enhanced Administration and Monitoring of Durables

The REST API and FTL Server GUI have new functionality:

- Purge and delete multiple durables with a single GUI operation or REST request.
- Query/get durables by type (static or dynamic).

Migration of Stores from One Cluster to Another

The persistence service now supports migrating a store from one cluster to the other using the dump-state-to-disk file.

Pass FTL Properties as Java System Properties

For Java clients, you can now override application settings that are passed during realm connect. Realm properties like connect retries, client label, etc., can be passed to a Java application via -D<property_name>=<property_value>.

Filtering for Durables in Monitoring Request

When requesting durables for a store via a REST call, you can now filter the returned durables by durable name for realm services in all languages.

Support for eFTL-Channel Firebase Cloud Messaging Push Notifications

For eFTL Firebase Cloud Messaging (FCM) push notifications you can now set the server/api key via GUI or JSON configuration.

Administration Web API FTL Server Running Status Check

The new administration web REST API, /api/v1/available, returns a status code to quickly tell the health and availability of an FTL Server or groupserver.

Realm Service Initial Configuration Option startup option to load an initial JSON

A new FTL server parameter, initial.realm.config lets you specify a JSON-configuration filename containing an initial realm configuration. The first time the FTL server is run, the FTL server is populated with this configuration. Subsequent runs of the FTL server use the last deployed realm configuration.

Persistence State File Management

A persistence service now moves a state file automatically after successfully loading the state file.

Linux Concurrent Installations

You can now install side-by-side versions of FTL on a Linux computer.

Field for timed out clients in component counts of the ftlserver cluster

The REST API api/v1/server now returns the count of timed out clients within the server_cluster.component_count section with field name timed_out_client_count.

New Persistence Monitoring Statistics

Persistence monitoring has new additional statistics values for the number of messages added to and removed from a store.

New Persistence Quorum Status via Web API

Now, via web APIs, you can get persistence quorum information regarding status and statistics.

6.2.0

Message Swapping

You can now enable persistence services to swap messages to disk when needed. This can be activated via a switch in the GUI or via web API.

.NET Core support

.NET Core is now supported, in addition to .NET Framework.

Targeted Deployments

During a deployment, clients are no longer targeted if there are no changes that would affect those clients.

Multiple Different Realm Connects From a Single Application

An application can now connect to different realm servers or services and be connected to multiple realms at the same time. Administrative changes made at one realm are not going to affect the application's connection to the other realms.

Map Iterator Matcher

For persistent stores, you can now create a map iterator that specifies a matcher, thus limiting the scope of the iteration. In addition, you can get the number of key/value pairs present in the map.

Dual-Use http Connection

For persistent http connections, you can now use the same http connection to make eFTL publish/subscribe web API calls and also realm web API calls.

OpenSSL 1.1.1c Support

FTL now supports OpenSSL 1.1.1c.

6.1.0

Performance Enhancements

Improved throughput and scalability of FTL servers.

Wide-Area Forwarding Zones

Enhanced GUI and REST API for zones.

Client Monitoring and Logging Enhancements

Clients that FTL creates internally now set their client labels for improved readability of log and monitoring data.

Log messages incorporate this information.

Grafana Dashboards

Updated Grafana dashboards, and added a new dashboard to monitor eFTL channels.

Server-Defined Transport for Message Broker Client Applications

An application definition can specify a server-defined transport, `Server`, which automatically connects with the persistence services of the message broker. The GUI specifies this behavior by default for new application definitions.

Map Label Property

Map create calls now accept a label property. Monitoring data incorporates the label.

Persistence Enhancement

Standard durables and templates no longer require a direct path.

FTL Server Web API

Added a new web API REST command to shut down only core servers.

Application-Defined Locking for Persistence Methods

A new API property governs retries for methods that request locks.

Docker Image for Log Service

A new Docker image runs `tiblogsvc`.

Go Language API

Go language message objects now support the Marshal and Unmarshal interfaces.

Documentation for Go Language API

Enhanced documentation for Go language API is now available in the book in TIBCO FTL Development, and through GoDoc.

6.0.0

FTL Server

A new FTL server executable consolidates several services into one convenient executable. The FTL server incorporates the realm server, persistence server, group server, transport bridge process, and eFTL server.

See *TIBCO FTL Administration* guide.

Wide-Area Durables and Stores

Client applications can publish and subscribe to wide-area durables across network boundaries.

GUI Grid Search

You can search for objects in a GUI grid, or in any column of a grid. You can search using simple strings or regular expressions.

Message Broker

The FTL server can operate as a message broker.

Updated Resources on TIBCO Community

Supplemental resources are now distributed at the TIBCO FTL Community Wiki in the Reference Info tab. You can always find the latest versions of these resources in that location.

Those resources include *TIBCO FTL Quick Start Guide* and *TIBCO FTL Tutorials*. They also include sample FTL server configuration files and sample realm definition files.

Changes in Functionality

The following changes in functionality were introduced in recent releases of TIBCO FTL® software:

6.10.1

When disaster recovery is configured, an administrator may now make configuration changes at the primary site even if the disaster recovery site is not reachable. Although it is recommended that disaster recovery site should be reachable when administrator makes configuration changes at the primary site.

6.9.0

In previous releases, some affiliated FTL servers would function correctly in secure realms with only the FTL trust file, despite what the documentation stated. Now, both the keystore file and trust file must be distributed to all FTL servers, namely core servers and auxiliary servers at all sites (including primary, satellite, and DR sites).

6.7.0

tibmongateway as a Service

When tibmongateway is run as a service under the FTL server, it no longer requires FTL trust parameters to be specified.

Realm Database Filenames

The realm database maintained by the core servers now uses a different file name (*filename*.persist rather than *filename*.dat). When migrating to FTL 6.7.x, the old *filename*.dat file is imported and renamed.

6.6.0

Log Level Changes

Changes to a client or service log level are now always logged.

Monitoring Gateway Startup

Startup options are now printed when tibmongateway starts.

Grafana Persistence Server Count

In the monitoring Grafana page **FTL Services: Overview** or **FTL Applications: Overview**, the **Persistence Server Count** value no longer includes the internal config cluster's persistence services.

Grafana Custom Dashboards

If your 6.5.x (or earlier) FTL monitoring uses FTL-supplied Grafana dashboards, you must delete and re-deploy those dashboards using the new versions supplied with FTL 6.6.0. If those dashboards were customer-modified, you must re-apply those modifications.

Realm Service Monitoring Message Re-publishing

The realm service no longer automatically creates a subscription on the monitoring endpoint to re-publish monitoring messages for the benefit of pre-6.0 clients. For the realm server to re-publish monitoring messages to pre-6.0 clients you must set the `legacy.monitoring` configuration property in the realm service YAML configuration file.

6.5.0

Expanded Persistence Monitoring Statistics

Store monitoring REST API call responses now include the configured memory size limit (bytelimit). Also Durable monitoring REST API call responses now include message limits.

Realm Configuration Deployment

The FTL server now deploys a realm configuration even if there are no changes.

TIBCO Hawk Microagent

TIBCO Hawk Microagent for TIBCO FTL/TIBCO eFTL Software is now available under TIBCO Hawk at eDelivery.tibco.com.

ftlstart Samples Script

The `ftlstart` command in the samples directory no longer loads the `tibrealm.json` configuration file. To use the advanced sample features, load `tibrealm.json` after starting the FTL server.

6.4.0

Port Ranges for Dynamic TCP Transports

In the administrative GUI Transports grid, for a DTCP Transport you can now specify a range of ports.

6.3.0

YAML File Syntax for Windows

For Microsoft Windows-based installations, in the YAML configuration file, the auth.url value now requires forward slashes instead of backslashes.

6.2.0

Authentication-Only Security

Running the FTL Server in authentication-only security mode now requires running tibftlserver --init-auth-only before launching the server. This command generates security files ftl-trust.pem and ftl-tport.p12. Copy these files into the data directory of all the core FTL servers.

6.1.0

Persistence Stores Grid

GUI persistence stores grid has changed to simplify configuration of wide-area stores.

The columns *Persistence Cluster* and *Zone* have been replaced by new columns *Scope* and *Cluster/Zone*.

The Scope column indicates the scope of a persistence store, that is, whether the durables of the store are available only within a specific cluster of persistence services, or throughout the clusters of a zone.

The Cluster/Zone column names the cluster or zone where the durables of the store are available.

Persistence Clusters Status Web API

REST API calls to locations that start with

```
http://<host>:<port>/api/v1/persistence
```

now begin instead with

```
http://<host>:<port>/api/v1/persistence/clusters
```

Calls to old locations are deprecated, and will become obsolete in the November 2019 release. Until then, the old locations automatically map to the new calls.

Note: Update your code appropriately.

Inter-Cluster Transport

The GUI parameter formerly named "Inter-Cluster Transport" is now renamed "Zone Transport." You can set this parameter in the persistence service details panel.

Go Language API

The Go language API (in Releases 6.0.1 and earlier) misspelled the following constants:

- PublisherPropertyStringLabel was misspelled as PublisterPropertyStringLabel.
- MonitoringTypeRecordsToCatchUp was misspelled as MonitoringTypeRecordsToCatcUp.
- MonitoringTypePendingConnectionCount was misspelled as MonitoringTypePendingMessageCount.
- MonitoringTypeClientDestroyedCount was misspelled as MonitoringTypeClientDestrouedCount.

If you copied the misspelled names of these constants into your Go program code, correct them before migrating to Release 6.1 or later.

Map Property Constants

The following API constant values were incorrect. If your code uses the old literal values of these constants, you must update your code. It is good practice to code using constants rather than values.

Language	Constant	Old Value	New Value
.NET	FTL.TIBMAP_PROPERTY_STRING_LABEL	com.tibco.ftl.client.**publisher**.label	com.tibco.ftl.client.**map**.label
Java	TibMap.PROPERTY_STRING_LABEL	com.tibco.ftl.client.**publisher**.label	com.tibco.ftl.client.**map**.label

6.0.0

FTL Server

A new FTL server executable consolidates several services that were formerly separate executables. The FTL server incorporates the realm server, persistence server, group server, transport bridge process, and eFTL server.

Those executable processes are now obsolete. However, to facilitate migration to Release 6.0, the new services that replace them (within the FTL server) can interoperate with the older server processes. It is good practice to migrate your enterprise expeditiously.

Docker images for those executables are also obsolete. The release package now includes exactly one docker image, for the FTL server.

Parameter names that previously referred to the realm server now refer to the FTL server instead.

See TIBCO FTL Administration.

File-Based Configuration

The FTL server executable accepts a dramatically reduced set of command line parameters. Configure all parameters for the FTL server and its services in a consolidated configuration file. See TIBCO FTL Administration.

Realm Service Fault Tolerance Based on Clusters

Release 6.0 features a new fault tolerance mechanism for realm services within FTL core servers. This new mechanism, based on clusters of active FTL servers, completely replaces the old arrangement of backup realm servers.

Backup realm servers are obsolete, along with all configuration parameters related to them, including parameters of the realm service and also of its clients. The authorization group ftl-backup is also obsolete.

Realm Server Administration Utility

The new FTL server administration utility (tibftladmin) completely replaces the old realm server administration utility (tibrealmadmin). The new name indicates additional functionality.

Use only command line parameters to guide the behavior of the FTL administration utility, as it does not support reading parameters from a configuration file.

Persistence Retry Behavior

The default behavior is now Unlimited. That is, persistence calls retry the interaction indefinitely. Calls return only upon success. In earlier releases the default behavior was None.

Persistence Cluster Transport

It is now illegal to modify the cluster transport of a persistence cluster.

Authorization Groups

The authorization groups ftl-primary, ftl-satellite, ftl-backup, ftl-dr are obsolete. For each of these, use ftl-internal instead.

GET server REST API

The response output of the REST request GET api/v1/server has changed to reflect the new organization of the FTL server providing services. In particular, this request gets information pertaining only to responding FTL server.

New REST requests get information about each FTL core server and the core servers cluster as a whole.

For example, to get the client counts for all core servers, use GET api/v1/cluster (rather than GET api/v1/server).

Default Application Definition

The behavior of the default application definition has changed. In earlier releases the default application specified peer-to-peer message delivery. In Release 6.0 it specifies delivery using a message broker model, using the default non-persistent store and the dynamic standard durable template named pubsub.

For the previous behavior, use the default-peer-to-peer application definition.

Built-In Dynamic Durable Templates

The names of the built-in dynamic durable templates have changed.

Built-In Clusters

The name of the default cluster has changed.

Agent

The agent component is obsolete. FTL clients and servers running in Docker containers no longer require the agent.

Deprecated and Removed Features

The following tables list any features that have been deprecated or removed as of Release 6.10.1 of TIBCO FTL® software:

For deprecated features, if relevant, useful alternatives to the deprecated features are listed. Any use of a deprecated feature should be discontinued as it may be removed in a future release. You should avoid becoming dependent on deprecated features and become familiar with the suggested alternative features.

Platforms

Affected Platform	Migration	Affected Release
Windows Server 2016	Migrate to Windows Server 2019.	6.7.1
Windows Server 2012	Migrate to Windows Server 2016.	6.1.0
Windows Server 2008	Migrate to Windows Server 2016.	Deprecated in Release 5.3.0
Apple macOS 10.14 64-bit, x86-64	Migrate to 10.15.x.	6.6.1
Apple macOS 10.13 64-bit, x86-64	Migrate to 10.14.x, 10.15.x.	6.6.0
Apple macOS 10.12 64-bit, x86-64	Migrate to 10.14.x, 10.15.x.	6.1.0
Apple Mac OS X 10.11 64-bit, x86-64	Migrate to 10.14.x, 10.15.x	6.0.1

Affected Platform	Migration	Affected Release
Red Hat Enterprise Linux Server 5.x 64-bit, x86-64	Migrate to 6.x or 7.x.	4.3.0
SUSE Linux Enterprise Server 11.x 64-bit, x86-64	Migrate to 12.x or 15.	6.7.0
SUSE Linux Enterprise Server 11.0 64-bit, x86-64	Migrate to 11.4 or 12.	4.2.0

Deprecated and Removed Features

Affected Component	Description	Deprecated Release	Removed Release
Go API constant	The Go API constant PublisherSendPolicyBatching is deprecated. Use PublisherSendPolicyNonInline instead.	6.10.0	
RDMA support on Windows	RDMA support on Windows is deprecated.	6.10.0	
Display of Client Application Statistics	The display of client application statistics is removed.	6.10.0	6.10.0
Realm Configuration Matcher	The ability to change the matcher of an existing static durable in the realm configuration is deprecated. In future releases changes to the matcher of an existing static	6.10.0	

Affected Component	Description	Deprecated Release	Removed Release
	durable will result in an exception when deployed.		
FTL Monitoring Components	The FTL Monitoring component (monitoring directory) including Grafana and tibmongateway are removed. Use TIBCO® Messaging Monitor for TIBCO FTL®.	6.9.0	6.10.0
Monitoring Metrics	The following monitoring metric types are removed and data will not be returned. Use TIBCO® Messaging Monitor for TIBCO FTL®. #define TIB_MONITORING_ TYPE_CONN_INFO 90001 #define TIB_MONITORING_ TYPE_CONN_INFO_NAME "connection_definition" /** Transport connection - Number of matches performed at receiving side only*/ #define TIB_MONITORING_ TYPE_CONN_RCVR_SIDE_ MATCHES 90002 #define TIB_MONITORING_ TYPE_CONN_RCVR_SIDE_ MATCHES_NAME "receive_ side_matches" /** Transport connection - Number of received messages	6.9.0	6.10.0

Affected Component	Description	Deprecated Release	Removed Release
	that failed to match*/ #define TIB_MONITORING_ TYPE_CONN_RCVR_SIDE_ DISCARDS 90003 #define TIB_MONITORING_ TYPE_CONN_RCVR_SIDE_ DISCARDS_NAME "failed_ matches" /** Transport connection - Number of received messages matched the NULL matcher*/ #define TIB_MONITORING_ TYPE_CONN_REC_NULL_ MATCHES 90004 #define TIB_MONITORING_ TYPE_CONN_REC_NULL_ MATCHES_NAME "null_ matches" /** Bytes sent. */ #define TIB_MONITORING_ TYPE_CONN_SENT_BYTES 90005 #define TIB_MONITORING_ TYPE_CONN_SENT_BYTES_ NAME "connection_sent_ bytes" /** Bytes received. */ #define TIB_MONITORING_ TYPE_CONN_REC_BYTES 90006 #define TIB_MONITORING_		

Affected Component	Description	Deprecated Release	Removed Release
	TYPE_CONN_REC_BYTES_ NAME "connection_received_ bytes" /** Messages sent using normal matching procedure */ #define TIB_MONITORING_ TYPE_CONN_SENT_MSGS_ MATCHING 90007 #define TIB_MONITORING_ TYPE_CONN_SENT_MSGS_ MATCHING_NAME "connection_sent_msgs_ match" /** Messages sent using without matching*/ #define TIB_MONITORING_ TYPE_CONN_SENT_MSGS_ OPAQUE 90008 #define TIB_MONITORING_ TYPE_CONN_SENT_MSGS_ OPAQUE_NAME "connection_ sent_msgs_exp" /** Received messages that matched a certain matcher*/ #define TIB_MONITORING_ TYPE_CONN_REC_MATCHING 90009 #define TIB_MONITORING_ TYPE_CONN_REC_MATCHING_ NAME "receive_matcher_ matches"		

Affected Component	Description	Deprecated Release	Removed Release
	/** Sent messages that matched a certain matcher*/ #define TIB_MONITORING_ TYPE_CONN_SENT_MATCHING 90010 #define TIB_MONITORING_ TYPE_CONN_SENT_ MATCHING_NAME "send_ matcher_matches" /** End Point Description */ #define TIB_MONITORING_ TYPE_EP_INFO 90011 #define TIB_MONITORING_ TYPE_EP_INFO_NAME "endpoint_description" /** End Point Delivered Messages */ #define TIB_MONITORING_ TYPE_EP_DELIVERED_MSGS 90012 #define TIB_MONITORING_ TYPE_EP_DELIVERED_MSGS_ NAME "endpoint_delivered" /** End Point Published Messages */ #define TIB_MONITORING_ TYPE_EP_PUBLISHED_MSGS 90013 #define TIB_MONITORING_ TYPE_EP_PUBLISHED_MSGS_		

Affected Component	Description	Deprecated Release	Removed Release
	NAME "endpoint_published" /** Outgoing Messages that were discarded*/ #define TIB_MONITORING_ TYPE_EP_SENT_ NONMATCHING_MSGS 90014 #define TIB_MONITORING_ TYPE_EP_SENT_ NONMATCHING_MSGS_NAME "endpoint_out_discarded" /** Subscriber Delivered Messages */ #define TIB_MONITORING_ TYPE_SUB_DELIVERED_MSGS 90015 #define TIB_MONITORING_ TYPE_SUB_DELIVERED_MSGS_ NAME "subscriber_delivered" /** Publisher Published Messages */ #define TIB_MONITORING_ TYPE_PUB_PUBLISHED_MSGS 90016 #define TIB_MONITORING_ TYPE_PUB_PUBLISHED_MSGS_ NAME "publisher_published" /** Outgoing Messages that were discarded*/ #define TIB_MONITORING_ TYPE_PUB_SENT_		

Affected Component	Description	Deprecated Release	Removed Release
	NONMATCHING_MSGS 90017 #define TIB_MONITORING_ TYPE_PUB_SENT_ NONMATCHING_MSGS_NAME "publisher_discarded" /** Packets sent by connection. */ #define TIB_MONITORING_ TYPE_CONN_PACKETS_SENT 90018 #define TIB_MONITORING_ TYPE_CONN_PACKETS_SENT_ NAME "connection_packets_ sent" /** Packets received by connection. */ #define TIB_MONITORING_ TYPE_CONN_PACKETS_ RECEIVED 90019 #define TIB_MONITORING_ TYPE_CONN_PACKETS_ RECEIVED_NAME "connection_ packets_received" /** Packets retransmitted by connection. */ #define TIB_MONITORING_ TYPE_CONN_PACKETS_ RETRANSMITTED 90020 #define TIB_MONITORING_ TYPE_CONN_PACKETS_		

Affected Component	Description	Deprecated Release	Removed Release
	RETRANSMITTED_NAME "connection_packets_ retransmitted" /** Packets missed by connection. */ #define TIB_MONITORING_ TYPE_CONN_PACKETS_MISSED 90021 #define TIB_MONITORING_ TYPE_CONN_PACKETS_ MISSED_NAME "connection_ packets_missed" /** Packets lost outbound by connection. */ #define TIB_MONITORING_ TYPE_CONN_PACKETS_LOST_ OUTBOUND 90022 #define TIB_MONITORING_ TYPE_CONN_PACKETS_LOST_ OUTBOUND_NAME "connection_packets_lost_ outbound" /** Packets lost inbound by connection. */ #define TIB_MONITORING_ TYPE_CONN_PACKETS_LOST_ INBOUND 90023 #define TIB_MONITORING_ TYPE_CONN_PACKETS_LOST_ INBOUND_NAME "connection_ packets_lost_inbound"		

Affected Component	Description	Deprecated Release	Removed Release
	/** Multicast Receiver Stream Definition */ #define TIB_MONITORING_ TYPE_MCAST_REC_INFO 90024 #define TIB_MONITORING_ TYPE_MCAST_REC_INFO_NAME "mcast_receiver_stream_info" /** Multicast Sender Stream Definition */ #define TIB_MONITORING_ TYPE_MCAST_SND_INFO 90025 #define TIB_MONITORING_ TYPE_MCAST_SND_INFO_ NAME "mcast_sender_stream_ info" /** Multicast Receiver received bytes */ #define TIB_MONITORING_ TYPE_MCAST_REC_BYTES 90026 #define TIB_MONITORING_ TYPE_MCAST_REC_BYTES_ NAME "mcast_receiver_bytes_ received" /** Multicast Receiver delivered bytes */ #define TIB_MONITORING_ TYPE_MCAST_DEL_BYTES 90027 #define TIB_MONITORING_		

Affected Component	Description	Deprecated Release	Removed Release
	TYPE_MCAST_DEL_BYTES_ NAME "mcast_receiver_bytes_ delivered" /** Multicast Receiver received packets */ #define TIB_MONITORING_ TYPE_MCAST_REC_PACKETS 90028 #define TIB_MONITORING_ TYPE_MCAST_REC_PACKETS_ NAME "mcast_receiver_ packets_received" /** Multicast Receiver delivered packets */ #define TIB_MONITORING_ TYPE_MCAST_DEL_PACKETS 90029 #define TIB_MONITORING_ TYPE_MCAST_DEL_PACKETS_ NAME "mcast_receiver_ packets_delivered" /** Multicast receiver sent protocol packets*/ #define TIB_MONITORING_ TYPE_MCAST_RCVR_SENT_ PACKETS 90030 #define TIB_MONITORING_ TYPE_MCAST_RCVR_SENT_ PACKETS_NAME "mcast_ receiver_packets_sent"		

Affected Component	Description	Deprecated Release	Removed Release
	/** Multicast sent NAK requests */ #define TIB_MONITORING_ TYPE_MCAST_SENT_NAK_ REQS 90031 #define TIB_MONITORING_ TYPE_MCAST_SENT_NAK_ REQS_NAME "mcast_receiver_ nak_requests_sent" /** Multicast sent NAKs requested */ #define TIB_MONITORING_ TYPE_MCAST_SENT_NAKS 90032 #define TIB_MONITORING_ TYPE_MCAST_SENT_NAKS_ NAME "mcast_receiver_naks_ requested" /** Multicast Receiver lost packets */ #define TIB_MONITORING_ TYPE_MCAST_LOST_PACKETS 90033 #define TIB_MONITORING_ TYPE_MCAST_LOST_PACKETS_ NAME "mcast_receiver_lost_ packets" /** Multicast Receiver duplicate packets received*/ #define TIB_MONITORING_		

Affected Component	Description	Deprecated Release	Removed Release
	TYPE_MCAST_DUP_REC_ PACKETS 90034 #define TIB_MONITORING_ TYPE_MCAST_DUP_REC_ PACKETS_NAME "mcast_ receiver_duplicates_received" /** Multicast Receiver data errors*/ #define TIB_MONITORING_ TYPE_MCAST_DATA_ERRORS 90035 #define TIB_MONITORING_ TYPE_MCAST_DATA_ERRORS_ NAME "mcast_receiver_data_ errors" /** Packets sent by multicast sender. */ #define TIB_MONITORING_ TYPE_MCAST_SENDER_ PACKETS_SENT 90036 #define TIB_MONITORING_ TYPE_MCAST_SENDER_ PACKETS_SENT_NAME "mcast_ sender_packets_sent" /** Bytes sent by multicast sender. */ #define TIB_MONITORING_ TYPE_MCAST_SENDER_BYTES_ SENT 90037 #define TIB_MONITORING_		

Affected Component	Description	Deprecated Release	Removed Release
	TYPE_MCAST_SENDER_BYTES_SENT_NAME "mcast_sender_bytes_sent" /** Packets retransmitted by the multicast sender. */ #define TIB_MONITORING_TYPE_MCAST_SENDER_PACKETS_RETRANSMITTED 90038 #define TIB_MONITORING_TYPE_MCAST_SENDER_PACKETS_RETRANSMITTED_NAME "mcast_sender_packets_retransmitted" /** Packets lost outbound by multicast sender. */ #define TIB_MONITORING_TYPE_MCAST_SENDER_PACKETS_LOST_OUTBOUND 90039 #define TIB_MONITORING_TYPE_MCAST_SENDER_PACKETS_LOST_OUTBOUND_NAME "mcast_sender_packets_lost" /** Backlog highest size during the previous period*/ #define TIB_MONITORING_TYPE_SEND_BACKLOG_MAX_SIZE 90040 #define TIB_MONITORING_		

Affected Component	Description	Deprecated Release	Removed Release
	TYPE_SEND_BACKLOG_MAX_SIZE_NAME "send_backlog_maximum_size"		
User Interface, Statistics Page (FTL-11958)	The display of client application statistics is now removed.	6.9.1	6.9.1
Server Clusters	A cluster of seven servers in a quorum is no longer supported. Three or five servers are recommended.	6.9.0	6.9.0
TLS without Authentication	TLS without Authentication is deprecated.	6.8.0	
Pre-Built Docker Images	Pre-built Docker images are no longer supplied with FTL software distributions.	6.7.1	6.7.1
Administrative GUI	The display of client application statistics is deprecated. It is planned for removal in the next minor release.	6.7.1	6.10.0
Persistence Service	The disk_mode configuration parameter for a persistence cluster in the web API is deprecated. Instead, use disk_swap and/or disk_persistence.	6.7.0	
RDMA Transport	On the Linux and Windows platforms, the RDMA transport is deprecated.	6.7.0	

Affected Component	Description	Deprecated Release	Removed Release
FTL Server Monitoring	For the web API call `api/v1/persistence/ftlservers/<ftl server name>/status` and in the administrative GUI, FTL Server status, the following fields are deprecated: • current_connection • max_connections • rejected_connections • lookup_failures Also in the administrative GUI, the **Connections at this server** section is deprecated.	6.7.0	

Persistence Monitoring Web API	REST API calls to locations that started with
	`api/v1/persistence`
	now begin instead with
	`api/v1/persistence/clusters`
	Until (but not including) the removal release, the old locations automatically map to the new calls.
	Note: Update your code appropriately.
		6.1.0	

Affected Component	Description	Deprecated Release	Removed Release
Go API	`MsgContent` is deprecated. Instead, use methods of the `Message` object to marshal and unmarshal data between messages and Go structs.	6.1.0	
FTL Server Web API	The REST API call `server {"cmd":"shutdown"}` is obsolete. Instead, use `ftlservers {"cmd":"shutdown"}`	6.1.0	6.1.0
FTL Server Administration Utility	Support for reading parameters from a configuration file is deprecated. Supply all commands and parameters on the command line.	6.0.0	6.0.0
Agent	The agent component is obsolete. FTL clients and servers running in Docker containers no longer require the agent.	6.0.0	6.0.0
Prometheus	Support for Prometheus is obsolete.	5.4.0	6.0.0
Monitoring Message Stream	New monitoring message types replace old types, which are deprecated: • 90012 replaces type 2. • 90013 replaces type 1. Use the new types. The old types remain in Release 5.4.0	5.4.0	

Affected Component	Description	Deprecated Release	Removed Release
	for backward compatibility, but will become obsolete in a future release.		
Bridge Setup Scripts	The Python scripts init_bridge.py and init_dtcp_bridge.py are obsolete. Use the realm server web API instead.	5.2.0	5.2.0
Realm Configuration Python Scripts	The realm configuration Python script, rs_script.py, and its supporting utility scripts are obsolete. Use the realm server web API instead.	5.2.0	5.2.0
Group Setup	The group facility is automatically enabled. The Python script init_groups.py is obsolete and no longer needed. The web API calls POST realm/groupserver and DELETE realm/groupserver are obsolete and no longer needed.	5.2.0	5.2.0
Realm Server Internal JAAS	The realm server now relies on a separate authentication service, rather than an internal JAAS component.	5.2.0	5.2.0
Realm Server Monitoring Interface	The realm server no longer stores historical monitoring data. For replacement functionality,	5.0.0	5.0.0

Affected Component	Description	Deprecated Release	Removed Release
	see *TIBCO FTL Monitoring*.		
Edit Transport Configuration Manually	The realm server GUI transport definition page no longer support manually editing a transport's JSON definition. To modify the transport definition, use either the GUI or the web API. See "PUT realm/transports/<name>" in TIBCO FTL Administration.	5.0.0	5.0.0
Adapter	The adapter converts and forwards messages between TIBCO FTL and TIBCO Rendezvous. This component is obsolete. This functionality is now part of TIBCO Rendezvous Network Server software.	4.2.0	4.3.0
API	API calls that facilitated request/reply interactions between TIBCO FTL and TIBCO eFTL are obsolete.	4.2.0	Deactivated in 4.2.0. Removed in 5.0.0.
FTL Server Configuration File	Support for eFTL Service configuration parameters server.cert, private.key, and private.key.password is deprecated. Instead, use FTL Server configuration parameters custom.cert, custom.cert.private.key , and custom.cert.private.key.password .	6.2.0	November 2019

Migration and Compatibility

The following information provides migration procedures for this release of TIBCO FTL® 6.10.1.

Migrating from a Previous Release of FTL

The section identifies compatibility issues among releases of TIBCO FTL® software. Instructions on how to migrate from a previous release to Release 6.10.1 of TIBCO FTL® software are included in the product documentation set.

If you are upgrading from 6.3.1 or earlier, and using auth-only or secure mode, you must first upgrade to TIBCO FTL® 6.3.1 HF 4 or TIBCO FTL® 6.4.0 HF1 (if you haven't already).

Update your applications to use the latest FTL client library.

See “Upgrade Migration to a New Release” in TIBCO FTL® Administration guide.

Note: If you are using the administrative GUI, after upgrading, clear your browser cache.

Compatibility with Earlier Releases of TIBCO FTL

Warning: TIBCO FTL® 6.10.1 data on disk representation is not compatible with an earlier version of TIBCO FTL®. Hence, before upgrading to TIBCO FTL® 6.10.1, see Downgrading to 6.10.0 Release of FTL or Downgrading to 6.9.0 Release of FTL section as applicable to take backups.

- If using insecure mode, components of TIBCO FTL® Release 6.7.1 can communicate with those of Release 5.4 or later.
- If using auth-only or secure mode, components of TIBCO FTL® Release 6.7.1 can communicate only with those of Release 6.3.1 HF4, or Release 6.4.0 HF1, or later.

- If using auth-only or secure mode, we recommend upgrading all clients to TIBCO FTL® Release 6.7.1 or later.
- If using auth-only or secure mode, clients developed at TIBCO FTL® Release 6.7.1 or later cannot communicate with FTL servers at any earlier release.
- If you plan to use disk persistence (6.7.0 or later), you must upgrade all clients to 6.7.x or later.

Satellite Default Routing

If a server or cluster is started with no realm database, and if the primary site servers are at 6.10 and the satellite site servers are at 6.5.x or earlier, default routing is not functional. If this issue occurs, migrate the satellite server(s) to 6.10.

Downgrading to 6.10.0 Release of FTL

TIBCO FTL® 6.10.1 data on disk representation is not compatible with an older version of TIBCO FTL®. Hence, before upgrading to TIBCO FTL® 6.10.1, use this procedure to back up the existing TIBCO FTL® 6.10.0 database. Use this procedure to downgrade to TIBCO FTL® 6.10.0.

Steps to be taken before upgrading to TIBCO FTL® 6.10.1

1. Back up the realm according to the documentation from TIBCO FTL® 6.10.0 release
    a. For example, `tibftladmin --backup_realm -ftls <host:port>` (`host:port` of one of the FTLServers from the FTLServer cluster)
2. Back up the persistence clusters according to the TIBCO FTL® 6.10.0 documentation. Use these steps when disk persistence is enabled for `ftl.default.cluster` and other user-defined clusters
    a. `tibftladmin --backup_persist --cluster ftl.default.cluster -ftls <host:port>` (host:port of one of the servers from the FTLServer cluster)

> **Note:** Check the progress of the backup by using the REST API, /api/v1/persistence/ftl.default.cluster/servers, look for the 'backup_in_ progress' field from the status JSON and if complete the field value is false

b. If you are running TIBCO Enterprise Message Service™ as a part of the FTLServer cluster, then run tibftladmin --backup_persist --cluster _embedded_tibemsd -ftls <host:port> (host:port of one of the FTLservers from the FTLServer cluster)

> **Note:** Check the progress of the back up by using the REST API, /api/v1/persistence/_embedded_tibemsd/servers, look for the 'backup_in_ progress' field from the status JSON and if complete the field value is false

c. Also if you have any user-defined persistence clusters, repeat this process for each user-defined persistence clusters

> **Note:** Check the progress of the backup using the REST API

d.

> **Note:** If disk persistence is not enabled, see TIBCO FTL® documentation for suspending and saving the state

3. Move the generated realm backups directory from the data directory of the realm as specified in the YAML file to some known location. For example, if the data directory for the realm is
   a. TIBCO_HOME/disk-persistence/ftlserver1/realm/data, then the backups directory would be in TIBCO_HOME/disk-persistence/ftlserver1/realm/data/backups, move this backups directory to some known location. For example, $HOME/realm/backups
4. Move the generated default cluster backup files from the persistence cluster's data directory to some well-defined location. For example, if the data directory for the persistence cluster is set to:
   a. TIBCO_HOME/disk-persistence/ftlserver1/persist/data, then the backup files are named something like this in the same directory
   b. default_ftlserver1_2023-07-28_10-24-40-567.persist.backup

c. Move this file to some well-defined location. For example, move it to $HOME/persist/backups

5. If you are running EMS as a part of the FTLServer cluster, similarly move the backup files generated in 2.1.2, related to the _embedded_tibemsd cluster to some known location. For example, move the files to $HOME/persist/backups. The files related to the _embedded_tibemsd cluster are in the path specified in the YAML file for tibemsd -store YAML option
6. If you have any user-defined persistence clusters, move the backup files to some known location
7. You can now upgrade from 6.10.0 to 6.10.1
8. You can downgrade to TIBCO FTL® 6.10.0, if required
    a. Gracefully shut down all the TIBCO FTL® 6.10.1 FTLServers
    b. Restore the state from the **backups** for the realm, the **backup files** associated with **ftl.default.cluster** and **_emedded_tibemsd** cluster. But before restoring, run the following steps:
        i. Delete the content of the realm data directory for all the three FTL servers in the cluster
        ii. Delete the content of the persistence cluster data directory for the ftl.default.cluster for all the three FTL servers in the cluster
        iii. Delete the content of the persistence cluster data directory for the _ emedded_tibemsd cluster for all the three FTL servers in the cluster
        iv. Delete the data directory content for any user-defined persistence clusters

        > **Note:** Any data that was sent or durables created after the upgrade would be lost at this point. The state is restored from the backups taken earlier than the upgrade. Similarly any deployments made after the upgrade would be lost

    c. Run the tibftladmin command to restore the realm data from the backups directory (assuming $HOME/realm/backups is where you saved off the realm backups earlier)
        i. tibftladmin --restore_realm --backupdir $HOME/realm/backups --datadir TIBCO_

HOME/disk-persistence/ftlserver1/realm/data --name ftlserver1

d. Run the tibftladmin command to restore the persistence cluster related files. Example for the default cluster related restore is

   i. tibftladmin --restore_persist --backupdir $HOME/persist/backups --datadir TIBCO_HOME/disk-persistence/ftlserver1/persist/data --name default_ftlserver1

e. Similarly run the -restore_persist command for the _embedded_tibemsd cluster. Example is

   i. tibftladmin --restore_persist --backupdir $HOME/persist/backups/ --datadir /opt/deployment/ftlserver1/ftlstore_data/ --name ftlserver1.

f. Also if you have any user-defined persistence clusters, repeat this process for each of the user-defined persistence clusters

> **Note:** If disk persistence is not enabled for ftl.default.cluster and other user-defined clusters, see TIBCO FTL® documentation on restoring state

9. Now restart all the FTLServers from previous version 6.10.0
   a. Ensure that the durables and messages earlier than the upgrade are preserved
   b. If you are running EMS, make sure that the EMS related topics, queues, and messages before the upgrade are preserved

Downgrading to 6.9.0 Release of FTL

TIBCO FTL® 6.10.1 data on disk representation is not compatible with an older version of TIBCO FTL®. Hence, before upgrading to TIBCO FTL® 6.10.1, use this procedure to back up the existing TIBCO FTL® 6.9.0 database. Use this procedure to downgrade to TIBCO FTL® 6.9.0.

Steps to be taken before upgrading to TIBCO FTL® 6.10.1

1. Back up the realm according to the documentation from TIBCO FTL® 6.9.0 release
   a. For example, tibftladmin --backup_database -ftls <host:port> (host:port of one of the

FTLServers from the FTLServer cluster)

2. Back up the persistence clusters according to the TIBCO FTL® 6.9.0 documentation. Use these steps when disk persistence is enabled for ftl.default.cluster and other user-defined clusters
    a. Go to the FTLServer UI that shows the running persistence clusters. Click the **backup** icon for
        i. ftl.default cluster
        ii. Any user-defined persistence clusters
        iii. _embedded_tibemsd cluster
        iv. Look for the FTLServer log message that indicates that the backup is complete. For example, look for 'Finished backup of'
    b. **Note:** If disk persistence is not enabled, see the TIBCO FTL® documentation for suspending and saving the state
3. Move the realm backups directory, from the realm data directory as specified in the YAML file, to some known location. For example, if the data directory for the realm is
    a. TIBCO_HOME/disk-persistence/ftlserver1/realm/data, then the backups directory would be in TIBCO_HOME/disk-persistence/ftlserver1/realm/data/backups, move this backups directory to some known location. For example, $HOME/realm/backups
4. Move the cluster backup files from the persistence cluster's data directory to a well-defined location. You can do it for one of the servers from the cluster. You do not need to save the files from all the servers from the FTLServer cluster. For example, if the data directory for the persistence cluster is set to
    a. TIBCO_HOME/disk-persistence/ftlserver1/persist/data, then the backup files are named something like this in the same directory
    b. default_ftlserver1_2023-07-28_10-24-40-567.persist.backup
    c. Move this file to some well-defined location. For example, move it to $HOME/persist/backups
5. If you are running EMS as part of the FTLServer cluster, move the embedded cluster related backup file to the known location. For example:
    a. ftlserver1_2023-07-31_15-29-15-448.persist.backup is the backup file of the embedded

tibemsd cluster. The FTLserver is named ftlserver1. Hence the backup is named according to the FTLserver name

6. If you are running user-defined persistence clusters, move the backup files related to the user-defined clusters to a known location
7. You can now upgrade from 6.9.0 to 6.10.1
8. You can downgrade to TIBCO FTL® 6.9.0, if required:
    a. Gracefully shut down all the TIBCO FTL® 6.10.1 FTLServers
    b. Before restoring the state from backup, run the following steps:
        i. Delete the content of the realm data directory for all the three FTL servers in the cluster
        ii. Delete the content of the persistence cluster data directory for the ftl.default.cluster for all the three FTL servers in the cluster
        iii. Delete the contents of the persistence cluster data directory for the _emedded_tibemsd cluster for all the three FTL servers in the cluster
        iv. Delete the content of the data directory of user-defined clusters
        v. **Note:** Any data that was sent or durables created after the upgrade would be lost at this point. The state is restored from the backups taken earlier than the upgrade. Similarly any deployments made after the upgrade would be lost
        vi. **Note:** If disk persistence is not enabled for ftl.default.cluster and other user-defined clusters, see TIBCO FTL® documentation on restoring state
    c. Restore the state from the backups for the realm using the procedure in the TIBCO FTL® 6.9.0 documentation. The procedure is a manual process of renaming the files from the backups directory and copying them to the realm ‘data’ directory. For example
        i. mv config_ftlserver1_2023-07-31_15-31-07-026.persist.backup config_ftlserver1.persist
        ii. mv rs_ftlserver1_2023-07-31_15-31-07-026.dat.backup rs_ftlserver1.dat

9. Rename the backup file of ftl.default.cluster from .backup to .persist. For example
    a. mv default_ftlserver1_2023-07-31_15-27-46-406.persist.backup default_ftlserver1.persist. This must be moved to the data directory of the ftl.default.cluster
10. Similarly move the backup of embedded_tibemsd from .backup to .persist
    a. mv ftlserver1_2023-07-31_15-29-15-448.persist.backup ftlserver1.persist
11. Similarly if you have any user-defined persistence clusters and their backup files, move them to the data directory of the user-defined persistence cluster
12. Restart all the FTLServers from the previous version of TIBCO FTL® 6.9.0
    a. Ensure that the durables and messages earlier than the upgrade are preserved
    b. If you are running EMS, make sure that EMS related topics, queues, and messages earlier than the upgrade are preserved

OpenSSL V1.1.1 to 3.0 Migration

Note: Some PKCS12 certificates created with OpenSSL versions prior to 3.0 must be updated. The default PKCS12 settings used by OpenSSL versions prior to 3.0 are considered insecure and not supported by default.

Industry security guidelines now recommend that certificates, ciphers and keys originally created using older protocols be upgraded to newer, stronger implementations as soon as possible to prevent unauthorized access to applications and systems. TIBCO Messaging customers should review and update current security configurations as soon as possible.

TIBCO FTL and TIBCO eFTL require strengthening ciphers and certificates, and remove older, exploitable protocols. There is a set of minimum requirements that affect the backwards compatibility of older certificates, ciphers and keys. Customers should review their current configurations and make updates. The “openssl” command line utility, version 1.1.1 and later, can be used to update existing installations.

In FTL/eFTL, clients and servers running in secure mode may be affected.

- Certificates, other than ftl-trust.pem, used in TIBCO FTL/eFTL may need to be updated.
- PKCS#12 files other than ftl-tport.p12 specified in TIBCO FTL/eFTL configurations may

need to be converted as specified below.

Changes in OpenSSL 3.0

As part of this strengthening of security, TIBCO has transitioned from OpenSSL 1.1.1 to OpenSSL 3.0. The new version attempts to simplify such things as cipher suite selection and key length choices using a security level setting (SECLEVEL) from 0 to 5. The default SECLEVEL is 1, and includes the following restrictions, as documented on the OpenSSL site:

- RSA, DSA and DH keys shorter than 1024 bits and ECC keys shorter than 160 bits are prohibited.
- All export cipher suites are prohibited.
- SSL version 2 is prohibited.
- Any cipher suite using MD5 for the MAC is also prohibited.
- Signatures using SHA1 and MD5 are also forbidden.

TIBCO products impose additional restrictions beyond these:

- SSL version 3 is disabled.
- TLS versions 1.0 and 1.1 are disabled.

Additional restrictions may be applied in the future as best practices evolve. Because of these restrictions, many of the cipher suites available in OpenSSL 1.1.1 are, by default, disabled. Certificates and keys that do not meet these criteria will fail.

The first consequence of this that may be noticed is that PKCS#12 files encrypted with older ciphers are no longer readable. This is because, by default, older utilities produced files that use RC2 encryption to protect the private key. RC2 is considered a legacy algorithm in OpenSSL 3.0. Not only does it not meet the criteria of SECLEVEL 1, it is not actually compiled into the main library. See the instructions below on how to convert older PKCS#12 files to a format acceptable to OpenSSL 3.0.

Converting the PKCS#12 file to newer ciphers introduces the requirement that all consumers of the file must support the new ciphers. Because TIBCO FTL Java clients use OpenSSL, the Java version is not relevant to compatibility.

Even after the file is converted, if the key algorithm or key size are not acceptable by modern standards, OpenSSL rejects any certificate based on that key. Customers or users need to replace existing certificates with new ones that meet the requirements of SECLEVEL 1.

There are other reasons that OpenSSL 3.0 may reject a customer generated certificate. OpenSSL 3.0 generally enforces the rules specified in the applicable RFCs much more strictly. For instance, the following errors may come up:

- Path length given without key usage
- Missing Authority Key Identifier
- Missing Subject Key Identifier
- Basic Constraints of CA cert not marked critical
- CA cert does not include key usage extension

This is not an exhaustive list, but represents a few of the errors seen in practice.

The restrictions on actual TLS cipher suite selection should be benign, since virtually all clients support at least one cipher mode that meets the criteria.

Converting PKCS#12 Files

To convert a legacy PKCS#12 file to newer algorithms using OpenSSL 1.1, use the following commands:

```
openssl pkcs12 -in sample.p12 -passin pass:password -nodes > tmp.txt
```

```
openssl pkcs12 -in tmp.txt -out fixed_sample.p12 -macalg SHA256 -keypbe AES-256-CBC -certpbe AES-256-CBC -export -passout pass:password
```

The corresponding commands for OpenSSL 3.0:

```
openssl pkcs12 -in sample.p12 -passin pass:password -noenc -legacy > tmp.txt
```

```
openssl pkcs12 -in tmp.txt -out fixed_sample.p12 -export -passout pass:password
```

The result can be verified with the following command:

```
openssl pkcs12 -in fixed_sample.p12 -info -noenc -noout -passin pass:password
```

If the output contains “RC2” in any of the text, then the p12 file is incompatible with OpenSSL 3.0.

Compatibility with Releases of Other TIBCO Products

TIBCO ActiveSpaces®

The enterprise edition of TIBCO ActiveSpaces® uses the enterprise edition of TIBCO Messaging and includes a license for it. The community edition of TIBCO ActiveSpaces is compatible with both the enterprise and community editions of TIBCO Messaging.

TIBCO Enterprise Message Service

TIBCO FTL® 6.10.1 is not compatible with TIBCO Enterprise Message Service (EMS) 8.5.0 or earlier releases. Upgrade to EMS 8.5.1 or later. When starting the TIBCO EMS server, ensure that the TIBCO EMS libraries precede the TIBCO FTL® libraries in the `module_path`.

Closed Issues

The following tables list closed issues in recent releases of TIBCO FTL® software:

Release 6.10.1

Key	Summary
FTL-14304	If tibMessage_SetMessage is used to set a field in the parent message (after a call to tibMessage_GetMessage of the same field) with different sub message, then subsequent calls to tibMessage_GetMessage of the same field return erroneous data. tibMessage_ToString prints erroneous data.
FTL-14299	The persistence service experiences a very long write stall or a general performance degradation, when many messages of varying sizes are stored and disk persistence is enabled.
FTL-14298	tibRealm_Close API may leak resources when called with an exception set.
FTL-14250	If the primary site goes down, it is not possible to restart the FTL servers on the disaster recovery site.
FTL-14221	A clarified error message is displayed when duplicate persistence services are created in the same persistence cluster.
FTL-14208	If an extremely large message is published to a replicated persistence store, and the transport layer security (TLS) is enabled, the persistence quorum might be disrupted.
FTL-14204	If the disk persistence is enabled and the FTL server is not shut down cleanly, the persistence cluster may take twice the expected time (pserver_timeout_pserver) or more to reform the quorum.

Release 6.10.0

Key	Summary
FTL-14170	Fixed the issue where, in secure mode, FTL server did not correctly infer the HTTPS protocol for the URLs supplied in the enable_dr REST command. In this case the FTL server did not attempt to connect to the DR site. (There was no issue if the URLs explicitly specified the HTTPS protocol.)
FTL-14136	Fixed an issue where the FTL server could take longer than expected to process a very large realm JSON.
FTL-14116	Fixed an issue where the application might not reconnect and recover. This occurred when the persistence service discards when delivering messages to a subscriber, due to insufficient backlog buffer for the persistence client transport.
FTL-13855	Fixed an issue where, in rare cases, the FTL library could crash when the TCP transport completes a connection.
FTL-14113	Resolved an issue where, in rare cases, connections to a satellite or DR site might cause FTL server to crash.
FTL-14111	Resolved an issue where, after restarting an FTL server, the ordinals of various group members might be re-assigned without notifying all affected members.
FTL-14064	The map API memory leak when reconnecting to the persistence service is resolved.
FTL-14060	Resolved an issue where, in Windows, the DTCP transport could fail to initialize if you are using TIB_DTCP_EXTERNAL.
FTL-14039	The FTL Monitoring component (monitoring directory) including Grafana and tibmongateway are removed. Use TIBCO Messaging Monitor for FTL.
FTL-14038	Resolved a issue where, in rare cases, acknowledgments could be lost if a subscriber is closed just as a persistence service rejoins the quorum.
FTL-14036	Fixed an issue where, if a non-default value is configured for client_timeout_pserver, the persistence service might wait longer than expected to enforce the timeout.

FTL-14025	In the user interface, under eFTL Cluster configuration, the column name has been corrected from **Authorization** to **Authentication**.
FTL-14018	The satelliteof, drfor, and drto parameters were incorrectly ignored if placed in the globals section of the FTL server YAML configuration file.
FTL-13988	There was a spelling error in the following Go API constant: AdvisoryReasonPeristenceStoreAvailableAfterClusterDataloss A following new constant has been added and the old constant has been deprecated: AdvisoryReasonPersistenceStoreAvailableAfterClusterDataloss
FTL-13968	Fixed a deadlock when creating a non-inline publisher object on Windows.
FTL-13952 and FTL-13930	Fixed various memory leaks in the TIBCO FTL .NET client library. The memory leaks happened from calls to ITibMap.Get, ITibMap.GetMultiple, IPublisher.SendRequest, and IMessage.MutableCopy.
FTL-13935	The Swagger user interface now shows Example Value and Model of POST commands without error.
FTL-13849	An error is now correctly logged for these conditions: • When a satellite FTL server fails to connect to a primary FTL server • When a primary FTL server fails to connect to a disaster recovery FTL server
FTL-13822	Resolved an issue where the FTL client library consumes an unexpectedly large amount of memory when the map API is used to store very large values.
FTL-13821	Resolved issues so messages are not lost in the following rare conditions: • If a subscriber has received messages but has not dispatched them, and the subscriber reconnects to the persistence service • A reconnect on a route between persistence services in a forwarding zone
FTL-13820	The FTL Server logs a warning message if it fails to authenticate a client because it's connection to the authentication service failed due to:

	• An invalid certificate or • Invalid authentication service related credentials
FTL-13816	The FTL server logs a warning message if authentication fails because the FTL server is unable to connect to the authentication service.
FTL-13815	The FTL server prints a warning log message when auth.url is specified and the protocol of auth.url is of type https and the auth.trust.file is not specified in the YAML file.
FTL-13813	Resolved an issue where the user interface and REST API incorrectly reported the persistence service as offline when dumping a very large message backlog to disk.
FTL-13775	Clarified the error message returned by the FTL server when a realm deployment fails because the DR site is not up to date.
FTL-13558	FTL/eFTL supports Istio service mesh using server based connection for clients connected to the persistence service using auto transport. Also supported are eFTL clients connected to eFTL service and clients using Group API connected to the Group Service.
FTL-13169	The user experience in the FTL server user interface when purging multiple durables is improved.
FTL-13033	FTL Servers status page loads when there is no quorum and shows the server that is down in red.
FTL-12253	The user interface now displays the label of the satellite server under FTL Server GUI: Monitoring, Realm Services Status Page.
FTL-11958	The display of client application statistics is removed. See Deprecated and Removed Features.

Release 6.9.1

Key	Summary

FTL-13821	In rare cases, if a subscriber has received messages but has not dispatched them, and the subscriber reconnects to the persistence service, messages could be lost. Similarly, a reconnect on a route between persistence services in a forwarding zone could cause messages to be lost. This has been fixed.

Release 6.9.0

Key	Summary
FTL-13771	In rare cases, a rolling upgrade of persistence services at the DR site could lead to data loss if the DR site was activated.
FTL-13768	Certain disk configurations in Kubernetes deployments could prevent the FTL server from starting successfully.
FTL-13749	There is improved performance of the FTL client library in the presence of a large number of subscribers with matchers.
FTL-13729	tibSubscriber_AcknowledgeMessages now throws an exception if the subscriber is not an explicit ACK subscriber.
FTL-13422	Fixed a defect that can cause the client to abruptly exit during a reconnect to the FTL server.
FTL-12944	Fixed a defect where the monitoring data was incorrect if the monitoring interval was smaller than the heartbeat interval.

Release 6.8.0

Key	Summary
FTL-13312	If a static format is created and then deleted, and an identical static format of the same name is created, a receiver that saw the original static format might not receive messages sent using the new static format.
FTL-13306	Multi-zone routing (where a single store is routed through multiple forwarding zones) was not functional.
FTL-	If a client reconnects to a durable whose underlying template has been modified in

Key	Summary
13295	the past, the client may experience further unnecessary reconnects.
FTL-13294	A lingering persistence leader could take over a route on its way out, preventing message delivery to the legitimate leader.
FTL-13290	The realm UI now correctly displays the units for ack batch time.
FTL-13279	Reduced the permissiveness of the backups directly created by default by FTL server.
FTL-13274	Persistence services in the same realm may not re-use transport names. This is a realm validation error.
FTL-13235	On failover of a non-replicated store, a durable TTL might be applied before subscribers have a chance to reconnect.
FTL-13180	The FTL server that manages the startup and shutdown of services can occasionally lose heartbeats with the services, causing the FTL server to restart the services.
FTL-13174	In rare cases, a DTCP mesh (for example, the cluster transport of a persistence cluster) fails to form all required connections.
FTL-13146	An FTL client might exit unexpectedly if it discarded messages while sending on a process transport.
FTL-13129	When using the send request API, the reply could occasionally be lost.
FTL-13124	The UI no longer displays overlapping text in the Client Information section of the 'Reachable Clusters of Stores'.
FTL-13104	A realm JSON upload might cause default routing to stop working.
FTL-	If a persistence quorum is resized, and some of the metadata files (*.dat) are

Key	Summary
13103	leftover, the now-larger quorum might fail to form.
FTL-13101	The quorum might become stuck if a partition is introduced between the leader and one follower.
FTL-13075	When many clients are connected, the persistence service might use excess CPU.
FTL-13057	If the realm heartbeat and timeout intervals are reduced, the realm service might log the warning `Disconnecting from realm service.`
FTL-13055	If a persistence store has 16 or more shared durables, and a failover occurs, the new leader might unexpectedly exit or exhibit unexpected behavior.
FTL-13047	The `tibEventQueue_RunDispatch` call has a small memory leak.
FTL-13045	When upgrading a routed store to FTL 6.7.x or later, the upgraded persistence service might exit unexpectedly.
FTL-13032	On occasion, the persistence quorum might not converge immediately after a successful sync.
FTL-13022	If a previously serialized message is modified, sent on a process transport (without release-to-send), and then sent on a network transport (inline send), the message is not re-serialized for the network send.
FTL-13017	If receiving from a network transport and sending on a proc transport (with release-to-send), the subscriber on the network endpoint might eventually stop receiving messages.
FTL-13015	`tibMap_SetMultiple` fails if called repeatedly with more than 1,000 keys.
FTL-13005	After sending a message to the persistence service, a late-arriving send ack could be leaked by the client.

Key	Summary
FTL-12994	A crash could result if a message is received from a proc transport and sent on a network transport (no release to send).
FTL-12993	A leadership change at the Disaster Recovery (DR) site might lead to a synchronization failure in the DR persistence service.
FTL-12990	Fixed a defect in the FTL client library that could cause memory leaks under certain error conditions.
FTL-12989	Reduced the memory footprint of the persistence service when catching up on messages from a route, following a disconnect on that route.
FTL-12958	Using request-reply in a routed store could cause an unexpected exit of the persistence service.
FTL-12955	Fixed a defect where the DR server could re-assign an already assigned client ID.
FTL-12935	After disk persistence was changed from enabled to disabled, the persistence service printed an exception stack the first time it was restarted.
FTL-12928	When using async disk mode, a retransmit over a route could result in an abrupt exit of the persistence service.
FTL-12923	In rare cases when using async disk mode, a message could be leaked, provided the remove of a durable was in flight.
FTL-12921	If a large message (about 4K or more) is retransmitted across a route, and disk persistence is enabled, the message might be leaked on the disk.
FTL-12920	Suppressed a benign warning related to route retransmissions: "received ack... now expecting...".
FTL-12907	The client API did not raise an exception if a lock was broken between map API calls using the same lock (for example, due to disconnect). Change in API behavior:

Key	Summary
	• Previous versions of FTL did not report an exception if a lock was broken between map API calls using the same lock. That defect has been fixed. • As always, correct client applications must always check for an exception. In addition, if an exception occurs, the application must now return or destroy the lock before making another attempt.
FTL-12886	If Disaster Recovery (DR) persistence services are started while DR is disabled for the cluster, and then DR is later enabled, the DR followers might fail to synchronize.
FTL-12876	In rare cases, the realm backup command could lead to a crash.
FTL-12874	Fixed excessive CPU usage by timers when the tibEventQueue is inline.
FTL-12870	Fixed a defect in the multicast transport that, during multicast transport initialization, caused a race condition that prevented SPMs from being sent periodically.
FTL-12866	Clients can not connect when there is clock skew between the client and the secure server.
FTL-12863	An issue with the ftlstart script does not successfully connect a satellite server to a primary server when both are in secure mode.
FTL-12830	Workspace lock with timeout less than 5 seconds were not being honored correctly.
FTL-12829	Map iterators are not functional if the map durable uses a prefetch of 1.
FTL-12827	If disk persistence is enabled, the size of the persistence files is unnecessarily large at high send rates or when using a slow disk.
FTL-	If using async disk mode, and a message is expired due to TTL while being

Key	Summary
12819	acknowledged by a subscriber, the persistence service could log spurious warnings, leak memory, or exit abruptly.
FTL-12818	If disk persistence is enabled and a persistence service exits immediately after syncing, it might resurrect a message deleted during sync when it reloads the disk.
FTL-12808	If disk persistence is enabled and a persistence service crashes immediately after syncing, it might fail to re-load the disk with the exception "key already deleted".
FTL-12807	In the UI, a new field added in the format appears at the beginning and should be at the end. The option to sort fields results in confusion.
FTL-12785	Fixed a defect in tibmongateway where in a connection to uninitialized InfluxDB server caused issues in tibmongateway.
FTL-12777	In certain situations (for example, the network is severed non-gracefully), FTL server briefly stops servicing its socket, possibly causing timeouts (including quorum disruptions).
FTL-12720	If a subscriber on a shared durable and a subscriber with no store are added to the same event queue, the subscriber with no store could miss some messages if the client reconnects to the store.
FTL-12687	Deleting a channel now delete all internal objects associated with the channel.
FTL-12683	When disk persistence is enabled, recovery times for messages larger than 4K were higher than necessary.
FTL-12649	Disaster recovery activation could fail after multiple simultaneous failures at the active site.
FTL-12639	**Summary:** The route `prefetch` default value is 32, with a higher risk for accumulation of messages. **Resolution:** The default `prefetch` value is now changed to 1024.

Key	Summary
FTL-12638	Fixed a defect in the FTLServer GUI that prevented an admin from setting the spin limit on the cluster transport.
FTL-12637	There are unnecessary redeliveries over route for async disk mode.
FTL-12526	UI should open separate cluster UIs in the same browser and other UI updates.
FTL-12350	The FTL UI shows ftl.routing.inbox.store, ftl.routing.nonpersistent.store, and ftl.routing.persistent.store stores even when the FTL server cluster is not at full strength.
FTL-12344	Fixed a UI defect where durables on ftl.system.log.store and ftl.system.mon.store were not being displayed.
FTL-12295	Fixed a FTL server UI defect where the "Reachable Clusters" in the client statistics page was not readable.
FTL-12238	Fixed a FTL Server UI defect where the client information page wasn't available for durables on a store associated with a zone.
FTL-12041	Can not login a satellite server by clicking the satellite server within the primary server UI when auth is enabled for the primary server but the auth is not enabled for the satellite server.
FTL-11767	The FTL server would sometimes log the message: "request error: context canceled".
FTL-11714	In creating an application using peer-to-peer transport, the cluster should not be able to be set to 'none'.

Release 6.7.3

Key	Summary
FTL-12638	In the administrative GUI, there is no option for setting the spin limit for persistence cluster transport.

Key	Summary
FTL-12295	In the administrative GUI, the Client Statistics display has some overlapping text.
FTL-12238	**Summary:** In the administrative GUI, store forwarding durables, client ID, client information is not displayed when clicked. **Resolution:** Client information for this internal client is not available, so the client ID link is now non-clickable.

Release 6.7.2

Key	Summary
FTL-13017	If receiving from a network transport and sending on a Process transport (with release-to-send property), the subscriber on the network endpoint might eventually stop receiving messages.
FTL-13015	The tibMap_SetMultiple call fails if called repeatedly with more than 1,000 keys.
FTL-13009	If calling tibMap_SetMultiple repeatedly, using varying values for numKeys, the call may return before all values are committed by the persistence service. A failure of the leader persistence service may therefore result in data loss.
FTL-13005	After sending a message to the persistence service, a late-arriving send acknowledgment could be leaked by the client.
FTL-13001	A call to tibMessage_SetOpaqueDirect could cause the application to abruptly exit, especially if the opaque data was larger than 16k and the message was re-used for send.
FTL-12995	If a message is received from a Process transport (with release-to-callback property) and then sent on a network transport (with release-to-send), the client application might exit unexpectedly.
FTL-12994	If a message is received from a Process transport (without release-to-callback property) and then sent on a network transport (without release-to-send), the client application might exit unexpectedly.

Key	Summary
FTL-12990	Certain error conditions could cause internal memory leaks.
FTL-12989	When catching up on messages from a route following a disconnect on that route, a persistence service's memory footprint is larger than necessary.
FTL-12958	Using request-reply in a routed store could cause an unexpected exit of the persistence service.
FTL-12955	The DR server might re-assign a client ID that is already assigned.
FTL-12935	After disk persistence is changed from enabled to disabled, the persistence service prints an exception stack the first time it is restarted.
FTL-12928	With disk mode set to async, a retransmit over a route could result in an abrupt exit of the persistence service.
FTL-12923	With disk mode set to async, in rare cases a message could be leaked while a remove-durable request was in flight.
FTL-12921	With disk persistence enabled, if a large message (approximately 4kB or larger) is retransmitted across a route, the message might be leaked on the disk.
FTL-12915	For durables in a routed store, a REST request returns an incorrect message limit value.
FTL-12907	The client API does not report an exception if a lock was broken between map API calls using the same lock. **Note:** Client applications must always check for a broken lock exception. If an exception occurs, the application must now return or destroy the lock before making another attempt.
FTL-12870	During multicast transport initialization, the transport causes a race condition that prevents SPMs from being sent periodically.

Key	Summary
FTL-12877	The FTL Server startup time is now significantly improved.
FTL-12829	For map durable with a prefetch value of 1, map iterators are not functional.
FTL-12828	In scenarios where a client is continuously sending messages, the connection is interrupted, and a new DNS resolve is made available, the reconnection delay is too long.
FTL-12827	If disk persistence is enabled, and if send rates are high or the disk is slow, the size of the persistence files is unnecessarily large.
FTL-12819	With disk mode async, if, while being acknowledged by a subscriber, a message is expired due to a TTL expiration, the persistence service could log spurious warnings, leak memory, or exit abruptly.
FTL-12818	If disk persistence is enabled, and a persistence service exits immediately after syncing, when it reloads the disk, it might resurrect a message that was deleted during the synchronization.
FTL-12808	If disk persistence is enabled, and a persistence service crashes immediately after syncing, it might fail to re-load the disk with the exception key already deleted.
FTL-12804	eFTL clients slow to connect sometimes experience a connection failure.
FTL-12786	If disk persistence is enabled, in rare cases the persistence service could deadlock when a quorum reformation occurs.
FTL-12777	In certain situations (e.g., if the network is severed non-gracefully), the FTL server briefly stops servicing its socket, possibly causing timeouts (including quorum disruptions).
FTL-12683	When disk persistence is enabled, recovery times for messages larger than 4kB are higher than necessary.
FTL-12637	If the data rate through a routed store is very high or bursty, redeliveries are more common than necessary.

Release 6.7.0

Key	Summary
FTL-12617	In the administrative GUI, durables that were created on the Primary Core servers do not appear upon a start of the Satellite Servers.
FTL-12614	If the realm property Manage All Formats is set to true, then the tibPublisher_SendRequest API call fails to send the request.
FTL-12611	If an API call tibMap_SetMultiple or tibMap_SetMultipleWithLock has more than 1000 keys, the client could abruptly exit.
FTL-12598	On Windows, attempting to set a transport backlog buffer size of 2GB or larger for a persistence transport causes an error in the persistence service.
FTL-12587	On Linux systems with Transparent Huge Pages enabled, the persistence service might use more memory than expected.
FTL-12546	If an application has an endpoint with a routed store, realm connect could fail following a disruption at the core servers.
FTL-12544	If the inter-cluster connection is interrupted for 20 seconds or more, messages queued for delivery across a route while the connection is down might be lost.
FTL-12543	A subscriber that experiences a lengthy disconnect and uses asynchronous acknowlegements might, after it reconnects, incorrectly acknowledge messages no longer owned.
FTL-12536	Unacked messages on a route might be duplicated if the inter-cluster link goes down.
FTL-12525	A persistence service might not recover from a synchronization failure if last-value durables with message swapping enabled were in use.
FTL-12511	In rare cases, attempts to create publishers in parallel on endpoints configured with a store could deadlock.
FTL-12499	A realm deployment could cause the persistence service **Server Status** field

Key	Summary
	in the administrative GUI to erroneously display the service as **Offline**.
FTL-12498	A sender discard during syncronization of the persistence services might permanently disrupt the quorum.
FTL-12494	An FTL client might not log anything if it attempts to connect to an unresponsive or unreachable FTL server.
FTL-12492	A persistence service might crash if a store is deleted from, and added again to, the realm definition.
FTL-12464	In some scenarios where FTL servers in a cluster fail, client group members do not receive GROUP_MEMBER_DISCONNECTED status advisories, and connected members receive erroneous GROUP_MEMBER_JOINED status advisories.
FTL-12463	If a group service becomes inactive due to an FTL server leader restart, the process during which the group service returns to an active state can cause erroneous status messages being sent to group clients.
FTL-12457	In the administrative GUI, some error messages for incorrect numeric field are vague.
FTL-12447	For messages received directly from a publisher, the delivery count returns a value of zero, when it should be -1.
FTL-12445	If a subscriber's endpoint is configured with a store and asynchronous auto acknowledgements, the acknowledgements could erroneously be sent to the persistence service prior to the callback handling of the messages.
FTL-12436	When restoring the default cluster from a dump file, the quorum might not form automatically unless all members are present.
FTL-12419	On Windows installations, the swap file used by persistence services is unnecessarily large.
FTL-12416	During cluster quorum formation, FTL might issue a warning that follower

Key	Summary
	replied to a message not yet sent.
FTL-12410	A persistence service that experiences a network interruption while synchronizing a last-value durable could ultimately fail to synchronize at all.
FTL-12379	See FTL-12464.
FTL-12376	The realm service uses more memory than necessary.
FTL-12349	A client application might abruptly exit when calling tibRealm_Close.
FTL-12303	In rare cases data loss in a persistence cluster can occur after a sync failure.
FTL-12296	**Summary:** The tibftladmin tool with option --available returns an exit code of 0 if the FTL server is unavailable. **Resolution:** The tibftladmin tool with option --available returns a non-zero exit code if the FTL server is unavailable.
FTL-12281	In rare cases a message in a routed store could be reflected back to the originating cluster.
FTL-12272	The administrative GUI contains an inaccurate Clients Connect Count display.
FTL-12239	If a REST request to the ftl server contains a malformed JSON body, the error response is vague.
FTL-12229	The persistence services ignore the configured weights after the first quorum formation.
FTL-12228	Go-language client applications using API calls from both the FTL and ActiveSpaces libraries might throw an exception.
FTL-12226	If multiple persistence services at the DR site are syncing, they could experience excessive CPU usage.

Key	Summary
FTL-12222	In Windows installations, the ftlstart.bat script fails to change directory to the data directory if the data directory of the FTL server is set to a different drive.
FTL-12196	Retry durations specified in the subscriber, publisher, map, and lock create calls were not interpreted correctly for very large positive values.
FTL-12182	When starting an FTL server with invalid password type stdin, the appropriate error message does not always display.
FTL-12144	In rare cases, after syncing, durable TTL might not be calculated correctly.
FTL-12130	A core realm service might fail to sync with the other core realms services after a restart..
FTL-12124	The API call tibMap_Get fails to honor the configured retry duration.
FTL-12121	Enabling centralized logging for a process at the INFO level, from the administrative GUI, might result in log messages not being sent to the realm service.
FTL-12114	See FTL-12228.
FTL-12104	Subscriber close may block for an unnecessarily long time for subscribers in non-replicated stores.
FTL-12053	A subscriber on a durable in a non-replicated store might experience an unnecessary reconnect if the quorum is temporarily disrupted.
FTL-11873	For a realm deployment with formats already part of realm configuration, the client library erroneously reports a needs restart message
FTL-11713	In the administrative GUI, the monitoring status of zone stores erroneously displays the status as Forwarding instead of Running or Off Line.
FTL-11306	Large concurrent publishes can cause discards on the persistence service cluster transport.

Key	Summary
FTL-10407	If interrupted while forming a quorum for the first time, the persistence service might log a warning about sync failure.

Release 6.6.1

Key	Summary
FTL-12506	A defect in the multicast transport could cause a subscriber to abruptly exit while processing a data packet. This can happen as a result of a race condition that prematurely destroys the multicast receiver object even while data is being processed.
FTL-12493	The multicast transport sometimes causes excessive CPU utilization when Packet Send Limit is set to unlimited (zero).
FTL-12480	When delivered by a persistence service, built-in opaque messages with a very small opaque field (less than 30 bytes) contains extra data at the end of the opaque field.
FTL-12477	A defect in the client library prevents a subscriber from FTL version 5.4.x to receive published messages from clients of version 6.5.x or greater.
FTL-12403	In the administrative GUI, the Channel Details page is missing the Save button and durables templates.
FTL-12393	The samples setup script GOPATH setting is incorrect.
FTL-12380	A client repeatedly calling tib_open/tib_close can sometimes result in thread local exhaustion.
FTL-12297	If a subscriber matches against a keyed opaque message, it does not receive messages.
FTL-12273	On occasion, if receiving messages just before a reconnect, a subscriber can stop receiving messages after the reconnect.
FTL-12254	In rare cases, performing a realm deployment while sending messages on a route can cause the persistence service to stop acknowledging messages on

Key	Summary
	the route.
FTL-12237	Persistence store forwarding does not function for built-in formats.

Release 6.6.0

Key	Summary
FTL-12178	Fixed a defect in the FTL golang library where a 'nil' value for the GroupPropertyMessageMemberDescriptor could cause the client to abruptly exit.
FTL-12151	In the administrative GUI, made corrections to some tables column headings for client and server statistics.
FTL-12148	Fixed an issue where a long-running persistence client could stop accepting deployments after an extended disconnect or a large number of previous deployments.
FTL-12147	Fixed a GUI issue where removing a transport from an eFTL channel on the eFTL table grid page was unsuccessful.
FTL-12125	Fixed an issue where the realm database could become inconsistent after a particular pattern of partitions and restarts.
FTL-12119	Fixed an issue where a REST API call to collect the latest persistence monitor counters would return unknown counters.
FTL-12078	Fixed an issue where a persistence service could occasionally fail to sync if leadership changed while it was waiting for sync to begin.
FTL-12052	Fixed an issue where, if a client subscribed to a durable using a matcher with duplicate fields, the persistence quorum was permanently disrupted.
FTL-12033	Fixed an issue where, if the FTL server was configured with a hostname (other than localhost) that resolved to local interfaces only, the FTL server would open ports on external interfaces.
FTL-12032	Fixed an FTLServer defect where an invalid core server address in the YAML

Key	Summary
	configuration file caused the FTLserver to abruptly exit.
FTL-12029	Improved the GUI displays of some FTL Realm pages for client statistics.
FTL-11990	Fixed an issue where some FTL REST API requests would return an HTTP status of 503 for a POST or PUT request with a body conting no content. These requests now return an HTTP status of 400.
FTL-11989	Fixed an issue where, in some instances, an FTL REST API call could return an HTTP status of 200 when the resource was not found (/api/v1/server/, /api/v1/realm/map/, /api/v1/realm/deployments/, /api/v1/applications/). These now return a status of 404.
FTL-11985	Fixed an issue where, in environments with high network or disk latency, a persistence service would occasionally be left out of a quorum, and would not attempt to rejoin.
FTL-11983	Fixed an issue where, if a secure ftlserver was started with a default realm configuration, the group server, group client, and eFTL cluster transports were not secure by default.
FTL-11975	Fixed an issue where, if started without satelliteof /drfor, an auxiliary ftlserver process at a satellite/dr site could initiate a deployment.
FTL-11974	Fixed an issue where, if using DR transports of type auto, disabling DR via realm deployment would cause all persistence services to respond with a Needs Restart display.
FTL-11973	Fixed an issue where the group server transport would open a port on an external interface even when the FTL server was listening on a local interface.
FTL-11966	Fixed an issue where status codes for REST commands to a persistence service would indicate success even if the persistence service didn't actually exist.
FTL-11961	Fixed an issue where, in some disaster recovery (DR) scenarios, FTL servers

Key	Summary
	would sometimes fail to start if FTL server names were reused across the DR event (e.g., if the original primary servers and the new DR servers shared the same names).
FTL-11957	An FTL persistence service now logs the name of the store if the store byte limit is exceeded.
FTL-11956	Fixed a REST API issue where, when calling /api/v1/reflector and a timeout occured, the realm would incorrectly log a warning-level error.
FTL-11952	Fixed a client library defect that prevented static and dynamic format messages from being forwarded to a subscriber on an endpoint that has a process transport configured.
FTL-11947	Fixed an issue where monitoring of an FTL server cluster sometimes caused high CPU usage.
FTL-11942	Fixed an issue where, for a single client process with many publishers, a reconnect to the persistence service might fail.
FTL-11940	Fixed an issue where, in rare cases (such as a very specific pattern of network delays), a client would be unable to store messages after reconnecting to a persistence service.
FTL-11935	Fixed an issue where a client calling tibLock_Return might cause a reconnect under high-load conditions.
FTL-11932	Fixed a GUI issue where The FTL Server monitoring page showed all zeros for **Connections at this server.**
FTL-11913	Fixed a secure-connection issue with Go language clients. Because FTL 6.6.0 uses Go version 1.15, custom certificates that are specified to the eFTL service must specify the hostname in the Subject Alternates Names section of the certificate. For more information see the Go 1.15 Release Notes at https://golang.org/doc/go1.15.

Key	Summary
FTL-11900	Fixed an issue where, if a persistence service lost leadership and then became leader again, it might log an exception such as violated radix heap invariant, though it would continue to function normally.
FTL-11890	In the administrative GUI, improved the way that statistics are displayed.
FTL-11885	The FTL server's configuration database no longer has a byte limit.
FTL-11883	Fixed an issue where the realm service was writing to disk even when no new clients were connecting and no deployments were in progress.
FTL-11880	Fixed a UI persistence store issue where the Used/Max Bytes and Consumed Store Limit value displays were sometimes erratic.
FTL-11875	Fixed a monitoring defect where an incorrect query prevented the FTL Persistence dashboard from displaying persistence service data.
FTL-11874	Fixed an FTL Go API library defect where a nil value specified for the group.GroupPropertyMessageMemberDescriptor property could cause the client to abruptly exit.
FTL-11871	Fixed an issue where the persistence servers' monitoring overview page was not displaying history, consistency, or disk status.
FTL-11831	Fixed an issue where the client would leak some memory on tibRealm_Close.
FTL-11793	Fixed a GUI issue where eFTL statistics data overlapped other data.
FTL-11789	Fixed an issue where, if tibSubscriber_Close was called while the client was disconnected from the persistence service, the call could block for an unnecessarily long time.
FTL-11758	Fixed panic in tibftladmin when the FTL server responded with a compressed HTTP response.
FTL-11705	Fixed erroneous HTTP headers returned by the realm service.

Key	Summary
FTL-11255	Fixed an issue where the message TTL for existing messages would reset after a rolling restart of persistence services.
FTL-11044	Fixed an issue where, at steady-state (no messages stored or acknowledged), a DR persistence service would sometimes report its status as syncing despite already being in sync.

Release 6.5.0

Key	Summary
FTL-11818	Fixed a client library defect that could cause the client to abruptly exit. This symptom could occur if the application had connected and then closed the connection to the FTL server numerous times.
FTL-11776	Fixed a Windows issue where tibPublisher_SendRequest() may return a timeout if multiple reply messages are received.
FTL-11771	Fixed an issue where, in the administrative GUI, Cluster Grid, Channel, for FTL Cluster and FTL Store, the selection None was not available.
FTL-11742	Fixed a defect in the FTL server that could cause the FTL server to abruptly exit if the tls.trust.file yaml parameter was invalid.
FTL-11738	Fixed an issue where tibMap_RemoveWithLock could hang if the lock was concurrently lost.
FTL-11734	Fixed a defect in the realm service that was causing it to report an incorrect satellite FTL server count.
FTL-11702	Fixed an issue where, for shared durables and standard durables with prefetch, if a message limit was configured and the discard policy was old, the persistence service could occasionally discard a recent message.
FTL-11700	Fixed an issue where a format created in the administrative GUI would not display after deployment.
FTL-11679	Fixed a defect in the client library that was preventing the administrator

Key	Summary
	from setting the multicast transport's UDP receive buffer size and UDP send buffer size to a value greater than 16MB.
FTL-11676	Fixed an issue where if the tibMap_Get call failed, the error description might be missing or incorrect.
FTL-11675	Fixed a memory leak caused by a failed realm connect operation.
FTL-11674	Fixed an issue where an application created in the administrative GUI would not allow the removal of an automatically configured a persistence cluster or store.
FTL-11671, FTL-6844	Fixed an issue where, during high throughput, a multicast sender could silently discard data. When the discarded data are fragments of a large message, the receiver could receive incomplete messages.
FTL-11662	Fixed an issue where, during reconnect, the FTL URLs would not randomize for load balancing.
FTL-11654	Fixed an issue where sending a reused message with NULL format on a non-inline peer to peer transport could result in message loss. This problem occurred only if a new field was added to the reused message between sends.
FTL-11653	Fixed an issue where a client connected to a persistence service would not fail over if a network partition was introduced between the original leader and the replicas and a blocking API call was in progress (e.g., send message).
FTL-11649	Fixed a defect in the FTL client library that was preventing Actives Spaces clients from invoking tibdg_Open()/tibdg_Close() in a loop.
FTL-11643	Fixed a defect that caused the administrative GUI to sometimes display an incorrect number of stores.
FTL-11642	Fixed a defect in the multicast transport where when an FTL 5.4.x receiving client logged a warning message ParseOpts ignoring unknown option type when

Key	Summary
	receiving data from a FTL 6.x sender. This message is now only logged once per FTL 6.x sender.
FTL-11636	Fixed an issue where traces for aggregated lvo durable discard advisories reported an incorrect event count.
FTL-11611	Fixed an issue of high memory usage when running the FTL server administrative GUI.
FTL-11610	Fixed an administrative GUI issue where the transport grid Address column was not visible, and where the browser console was displaying errors.
FTL-11602	In the administrative GUI, the eFTL channel's FTL persistence_duration configuration ios now displayed in the Channel information page.
FTL-11600	Fixed an issue where deployments could occasionally fail because a change to the default zone triggered a Needs Restart response.
FTL-11598	Fixed an issue where any time a message expired based on a TTL timeout, the persistence service could silently expire other messages too early or discard newly published messages.
FTL-11594	Fixed a Linux issue where running FTL servers or clients under a user whose group did not exist or was not mapped to a name could cause a client or the FTL server to abruptly exit.
FTL-11577	Fixed an issue where whenever a message was expired based on TTL, the leader persistence service would log an inexact count. The count is now exact and logged less frequently.
FTL-11563	In the administrative GUI, the Logging button for processes listed under Other Services now functions correctly.
FTL-11561	Fixed an issue where a realm service could abruptly exit when requesting monitoring data via the administrative GUI.
FTL-11560	Fixed an issue of occasional excess message iterator contention.

Key	Summary
FTL-11538	Fixed an issue where the administrative GUI could display the incorrect transport type (Dynamic TCP instead of Auto) after REST API configuration and deployment.
FTL-11535	Fixed an issue where if a persistence service encountered an exception while storing a message, the message could be lost even if the client received a send ack.
FTL-11532	Fixed an administrative GUI issue where when a new eFTL channel was created the channel's Cluster, Store, and Template were not blank.
FTL-11507	Fixed an issue with the ftlstart script where supplying both -secure and --auth would fail to start the FTL server.
FTL-11496, FTL-11387	Fixed an issue where creating a store and selecting the Scope as Zone would fail.
FTL-11494	Fixed an issue where, in the administrative GUI, Application Grid, for Cluster and Store, the selection None was not available.
FTL-11484	Fixed a defect where many clients connected to a persistence service could cause an internal timer in the persistence service to fire very rapidly.
FTL-11476	Fixed an issue where a persistence quorum could fail to store all messages from a call to tibPublisher_SendMessages if the middle of the batch goes missing on the network.
FTL-11466	Fixed a tibftlserver yaml file issue that would not allow valid tls.trust.everyone values.
FTL-11459	Fixed a defect in the client library where tibRealm_RemoveMap was not honoring the map retry duration property.
FTL-11456	Fixed an issue where if a tibstore of a version older than 6.4 has a non-empty datalog on disk, and a migration to 6.4 or later is attempted, then the new tibstore failed to upload the data log from the disk, and would throw a _tibMsgProps_Deserialize exception.

Key	Summary
FTL-11434	Fixed an issue where a restarted satellite FTL server would fail to reconnect to a cluster's primary server.
FTL-11427	Fixed a defect in the group client library where in if the observer flag is set to false during group join, the group member is not treated as ordinal member.
FTL-11406	Fixed an issue where a persistence cluster quorum could reform if publishers were sending at the same time that messages were expiring.
FTL-11390	Fixed an issue where the FTL client could abruptly exit when creating event queue timers with an interval of zero.
FTL-11333	Fixed an issue where the FTL server could abruptly exit on a call to `api/v1/support`.
FTL-11315	Corrected an administrative GUI label to indicate that Dynamic TCP port fields must always be a range.
FTL-11314	Fixed an issue where running the FTLserver docker image (`ftl-tibftlserver:6.4.0` and `ftl-tibeftlserver:6.4.0`) resulted in a benign warning (`Cannot assign requested address`) and an exception stack trace being printed on startup.
FTL-11305	A timeout can now be provided when creating a realm configuration workspace. Once the timeout expires, if the workspace has not yet been deployed the workspace is automatically deleted.
FTL-11298	Fixed an issue where, in the Go API, a deadlock could occur under a combination of the following conditions: • An unbuffered message channel is created. • A subscriber is created using the unbuffered message channel on a given event queue. • One or more messages are delivered to the application via the unbuffered message channel, but the Go code does not receive the messages (perhaps the Go routine servicing the unbuffered message channel has exited).

Key	Summary
	• The event queue's CloseSubscription() or Destroy() method is called.
FTL-11282	Timed-out clients of all types can now be purged.
FTL-11268	Fixed an FTL 6.4 issue where, if a persistence service discarded on its transport to the client, the message could be lost.
FTL-11247	Fixed an administrative GUI issue where, in eFTL Server Details, the last row is shown as Undefined.
FTL-11243	Fixed a client library defect where a create subscriber call on an advisory endpoint failed if the properties passed in had durable properties set.
FTL-11226	Fixed an issue where partially connected zones could experience a buildup of acknowledgments for messages exchanged between clusters.
FTL-11129	Fixed an issue where uploading a JSON file with tibftladmin could flag an error in the administrative GUI, but without any details available.
FTL-11127	Fixed a defect in the FTL client library where in a call to tibMessage_ToString with a small buffer could cause the client to abruptly exit.
FTL-11115	Fixed an issue where if tibRealm_Unsubscribe was called while the client had a corresponding subscriber object still open, or if tibRealm_RemoveMap was called while the client had a corresponding map object still open, the call was likely to time out.
FTL-11113	Fixed an issue where eFTL client connections could fail if an HTTP load balancer was placed between the client and the FTL server.
FTL-11024	Fixed an error where, when a durable monitoring call (GET api/v1/persistence/clusters/<cluster>/durables/<durable>) was issued, the FTL server REST API could mistakenly return 404 Not Found if the persistence service was unavailable.
FTL-10939	Fixed an issue where, when an FTL server exited, sometimes not all services exited.

Key	Summary
FTL-10801	Fixed an issue where store limit overruns or cluster reconnects might not be logged.
FTL-10546	The FTL server now rejects the yaml configuration if the supplied password in the yaml file uses the 'stdin' form.

Release 6.4.0

Key	Summary
FTL-11283	Fixed a defect in the client library that was failing to throw an exception if the 'send' capabilities were not set on the transport.
FTL-11183	Fixed an issue where a REST web request formats output incorrectly included an internal field entry.
FTL-11177	Fixed a realm service defect in which the internal _inboxEndpoint was erroneously removed from applications upon migration from 5.4.x.
FTL-11163	Fixed a monitoring issue where a store's disk_state is always incorrectly 0 in the response for REST web call api/v1/persistence/<cluster_name>/servers/<server_name>. Also, note that a successfully completed dump-to-disk operation for a persistence service is indicated when disk_state becomes 2.
FTL-11155	Fixed an issue that would occasionally, but rarely, cause a persistence service to unexpectedly exit.
FTL-11149	Fixed a client library defect that could cause the client to deadlock when publishing to and receiving from persistence service.
FTL-11144	Fixed an persistence service defect, where under certain conditions, the following warning was continually repeated: `realm <timestamp> warn ftl: Subscriber(1075) purge. Not found!`
FTL-11134	Fixed a defect in the persistence service where the monitoring counters

Key	Summary
	that return the number of messages out of the persistence service were incorrect.
FTL-11126	Fixed an issue where a persistence service would occasionally restart with disk_mode set to swap.
FTL-11121	Fixed an issue where a persistence service might fail to join its quorum if given an invalid state file to load.
FTL-11082	Fixed an issue where, in rare cases, a persistence service could under-report store message and byte counts after catching up from another persistence service.
FTL-10979	Fixed an issue where, if a store was migrated from 5.4.x, after migration the inbound/outbound message monitoring counters would not function for that store.
FTL-11075	Fixed a persistence service defect that could cause the persistence service to abruptly exit. This symptom could occur if message swapping was enabled.
FTL-11062	Fixed a realm service defect that was preventing the log level to be set in the presence of multiple eFTL services.
FTL-11032	Fixed an issue where a call to tibRealm_Close could occasionally stall for clients that received inbox messages via a persistence store.
FTL-11031	Fixed an issue where a subscriber configured for asynchronous acknowledgments would sometimes reconnect if a follower persistence service rejoined the quorum.
FTL-11027	Fixed a client library defect that could cause the client to abruptly exit. This symptom could occur during a reconnect to the persistence service.
FTL-11023	Fixed an issue where map operations using a lock could timeout if a persistence service concurrently rejoined a quorum.

Key	Summary
FTL-11022	If a deployment determines that a client or satellite is not going to reply, this is now reflected in realm deployment status queries.
FTL-11020	Fixed an issue in the FTL Server in which the FTL Server failed to process boolean arguments.
FTL-11006	Fixed an issue where when starting a set of FTL Servers for the first time, the initial deployment could occasionally time out.
FTL-11005	Fixed an issue where On occasion, a durable subscriber configured for asynchronous acknowledgments could drop an ack when reconnecting to a shared durable.
FTL-10997	Via the web API call for server status (api/v1/server), you can now see whether disaster recovery realm services are in sync with the latest deployment ("dr_in_sync": true).
FTL-10991	Fixed an issue where, in rare cases, a persistence service could crash if a client makes a call to tibSubscriber_Create immediately after calling tibSubscriber_Close for a subscriber on the same durable.
FTL-10987	Fixed a client library defect that could cause clients or FTL services to abruptly exit.
FTL-10979	Fixed an issue where, if a store was migrated from 5.4.x, after migration the inbound/outbound message monitoring counters would not function for that store.
FTL-10978	Fixed an issue where, on rare occasions, a realm service could stall during the initial deployment, preventing its FTL Server from fully initializing.
FTL-10977	Fixed an issue where, in rare cases, if a deployment changed the publisher mode of a store from store_confirm_send to store_send_noconfirm, a client with existing publishers might not return from a call to tibPublisher_ Send/tibPublisher_SendMessages.
FTL-10972	Fixed an issue where, in rare cases, a persistence service could stall when a

Key	Summary
	quorum reformed.
FTL-10969	Fixed a client library defect where tibMessage_ToString was returning a string containing internal field names.
FTL-10960	Fixed an issue where store names were required to be unique across clusters. Note that while store names no longer need to be unique across clusters, in the presence of multiple zones, stores names must be unique when associated with a zone.
FTL-10957	Fixed an issue where, in rare cases, a persistence service could crash if it started while a deployment was in progress.
FTL-10947	Fixed an issue where, in rare cases, a client publishing to a store configured with store_confirm_send could deadlock during a realm deployment.
FTL-10891	Fixed an issue where calls to tibPublisher_Send could timeout if a persistence service concurrently rejoined a quorum.
FTL-10783	Fixed an issue where durable message TTLs would reset if the leader of a persistence cluster changed.
FTL-10764	Fixed an issue where occasionally a sync ack durable subscriber would needlessly disconnect from the persistence service if a quorum follower rejoined the quorum.
FTL-10669	Fixed a client library defect that could cause a client application using a shared memory transport to deadlock when an administrator updated the realm in a way that affected the transport.
FTL-10635	Fixed an issue that allowed FTL clients to attempt to reconnect to the same port from which it just lost its connection, when alternate ports were available.
FTL-9550	Fixed a defect in the FTL Server that could cause it to abruptly exit if the certification file is empty.

Key	Summary
FTL-7627	In the administrative GUI, added a an icon/checkbox etc to show all the fields or reset to the default fields.

Release 6.3.1

Key	Summary
FTL-11075	Fixed a persistence service defect that, with message swapping enabled, could cause the persistence service to abruptly exit.
FTL-11051	Fixed a client library defect that prevented the client from accepting a new deployment after a reconnect to the FTL Server.
FTL-11048	Fixed a client library defect in which the application-specified identifier was erroneously being overwritten after a deployment.
FTL-10721	Fixed a defect in the FTL Server GUI in which the log level and centralized logging flags were not displayed correctly after their values were changed.

Release 6.3.0

Key	Summary
FTL-10928	Fixed an issue for when environment variable ALL_PROXY was set, the FTL server would not start.
FTL-10915	Corrected usage help text for ftlserver--init-auth-only and --init-security.
FTL-10859	Fixed an issue where stores defined in the default cluster did not have a byte limit by default.
FTL-10858	Fixed an issue where an unsubscribe operation would occasionally fail.
FTL-10853	Fixed an Administration GUI issue where the endpoint description was labeled Application Description.
FTL-10847	Fixed an issue where a persistence service storing messages whose formats have no fields defined would fail when syncing another persistence service.

Key	Summary
FTL-10830	Fixed an issue where if the leader of the default cluster was shut down, messages would start to accumulate in ftl.system.mon.store.
FTL-10826	Fixed an issue where auxiliary FTL servers would fail to initialize if provided a 5.4.x realmserver database file.
FTL-10824	Fixed an issue where automatic re-deployments for out-of-sync clients would take longer than expected.
FTL-10814	For the function tibRealm_CreateMap you can now supply NULL for endpointName. If NULL is specified for endpointName then the client library sets map as the endpointName.
FTL-10805	Fixed an Administration GUI issue where the Firebase Cloud API key text box (under eFTL channel settings) was too small to display the entire key.
FTL-10795	Fixed an issue where a persistence service configured for store forwarding would not close its connections to other persistence clusters when a zone-scoped store was deleted.
FTL-10789	A structure definition for the FTL advisory message was added to facilitate unmarshaling a received advisory message, for simpler processing.
FTL-10788	For consistency, the function group.Version() was added to the FTL golang Group API.
FTL-10782	Fixed a defect where the FTL golang API did not properly handle properties where the property value was ftl.Message.
FTL-10772	Fixed an issue where if an FTL client subscribed with an invalid content matcher, the persistence server could abruptly exit.
FTL-10751	Fixed an issue where if an FTL server was restarted with the same name but a different host/port, connections among FTL servers were sometimes not made properly.
FTL-10743	Fixed an issue where a call to tibRealm_Connect would not fail immediately,

Key	Summary
	as it should, if a password was specified without a username.
FTL-10725	Fixed an issue where if a persistence service encountered an invalid dump file, it would not exit.
FTL-10722	Fixed an issue where ftlserver would occasionally fail to restart a service correctly after that service became unresponsive.
FTL-10716	Fixed an issue where map iterators using matchers returned incomplete results if there was a large number of key/value pairs in the map.
FTL-10715	Fixed an issue where map operations with a lock could fail unnecessarily if the client was disconnected from the cluster during the request.
FTL-10699	Fixed an issue where, when deleting an eFTL channel, the eFTL server could report an exception following deployment, requiring a restart of. the eFTL server.
FTL-10690	Fixed a defect in which, on rare occasions, DTCP transport connections could be delayed by a minute or more.
FTL-10685	Fixed an issue in the client library to improve how fast a client can connect/reconnect to the persistence service.
FTL-10684	Fixed an issue where deployment status would be reported inconsistently across different FTL servers while a deployment was in progress.
FTL-10681	Fixed a defect in which an FTL server could fail to continue a deployment while a disaster recovery FTL server was connected.
FTL-10614	Fixed an issue where a GET api/v1/persistence/clusters/<cluster>/store/<stores>/durables/<durable> or GET api/v1/persistence/zones/<zone>/store/<stores>/durables/<durable> request could return, in addition to <durable>, other durables not named <durable>.
FTL-10609	Fixed a defect where if a deployment changed the name of the durable template assigned to an endpoint, associated clients did not automatically

Key	Summary
	prompt for a restart.
FTL-10594	Fixed a defect in which, if very high latency existed between a client and a persistence service, the client could fail to add a durable subscriber to an event queue.
FTL-10592	Fixed a defect in which, while sending a message to an inbox via a store, a client could hang.
FTL-10585	Fixed an issue where, on rare occasions, an FTL server that started a deployment could erroneously time out other FTL servers in the same cluster.
FTL-10573	Fixed a defect in which, if a client's application instance had been deleted from the realm configuration, that client could fail to get a deployment.
FTL-10572	Fixed a defect where when making a change to an application definition, the FTL server could fail to maintain the configured ordering of application instances.
FTL-10553	Fixed a defect with .NET clients, when if a call to TibMap.Key(key, lock) failed, the API erroneously returned a non-null message.
FTL-10552	Fixed a defect in which a client with many (i.e., several thousand) standard durable subscribers could experience a delay of several minutes before receiving messages for some subscribers.
FTL-10548	Fixed a defect in which, for a persistence cluster definition, increasing the pserver_timeout_pserver value to greater than the default value of 3 seconds would cause the quorum to take longer than expected to form.
FTL-10540	Fixed a defect for while a client was creating an inbox subscriber on an endpoint with a store, that client could experience timeouts.
FTL-10529	Fixed an issue where an eFTL channel's connection and subscription monitoring counters could be incorrect when using the eFTL REST API.

Key	Summary
FTL-10524	Fixed a defect where if client_pserver_heartbeat was set to less than 2 seconds in a persistence cluster configuration, clients of that cluster were unable to connect.
FTL-10476	Fixed an issue where client transport settings in a default persistence cluster would reset after an FTL server restart.
FTL-10475	Modified Administration GUI numeric field displays to show a consistent and limited number of digits after a decimal.
FTL-10462	Fixed an Administration GUI issue where monitoring data did not show client instant changes.
FTL-10460	Fixed a defect where if any FTL server in a disaster recovery cluster with persistence services using a DTCP cluster transport was restarted, the quorum might not reform.
FTL-10459	Improved exception stack prints in the FTL Golang API and the realm service.
FTL-10458	Fixed a defect where if a GET /channel/v1/subscribe/<durname> REST call was immediately followed by a DELETE /channel/v1/subscribe/<durname> REST call in a different connection, the DELETE call could fail.
FTL-10457	Fixed a defect where, on rare occasions, a persistence service in a store-forwarding zone was unable to maintain its connection to another persistence service in another cluster of the zone.
FTL-10452	Fixed a defect in the realm service that was preventing eFTL clients from invoking KVMap REST APIs when authentication is enabled on the eFTL server.
FTL-10448	Improved the administration GUI so that when viewing durables, content matcher strings are fully visible without the need to scroll.
FTL-10447	Fixed an issue where you could not always purge applications that were no longer running.

Key	Summary
FTL-10442	Fixed a defect where if persistence cluster configuration changes required the quorum to restart (e.g., timeout or heartbeat intervals), the quorum did not always reform.
FTL-10437	Samples have been updated to be more flexible and available in more languages.
FTL-10430	In the samples file dotnet/Makefile.Linux, removed a line at the top of the file with GroupClient.
FTL-10419	Fixed a defect where when under heavy load, if realm components became unavailable, the realm service or go clients might exit.
FTL-10418	Fixed a defect where if the satellite core FTL servers quorum reforms while the satellite servers are disconnected from the primary cluster, a satellite FTL server could abruptly exit.
FTL-10413	Fixed a defect in which redeploying a disaster recovery cluster could erroneously generate a read only error message.
FTL-10404	Fixed a defect in which an FTL server internal timeout could erroneously return an HTTP 404 for certain HTTP requests.
FTL-10392	Fixed an issue where the DR (Disaster Recovery) realm service was not updated with a deployment in a timely manner.
FTL-10366	Fixed a defect where core-server quorum delays (e.g., a slow disk) prevented some of the FTL servers from initializing properly.
FTL-10363	Fixed a defect in which processes that include the tib library, including tibftlserver and its modules, could on rare occasions crash when exiting.
FTL-10237	Fixed an issue where setting certain http proxy-related environment variables, such as http_proxy, could prevent basic FTL functioning.
FTL-10201	Fixed a defect where an auxiliary FTL server that was disconnected from the core FTL servers for an extended period of time could cause a REST API

Key	Summary
	shutdown command to the auxiliary FTL server to fail.
FTL-10185	Fixed an issue where ephemeral durables would not automatically expire.

Release 6.2.0

Key	Summary
FTL-10372	Fixed a defect in the eFTL Service that caused Apple push notifications to fail.
FTL-10347	Fixed a defect where the default cluster did not always reform a quorum following a restart of 2 out of 3 FTL servers.
FTL-10346	Fixed a defect where an FTL Server sometimes generates error message error processing cluster commit.
FTL-10333	Fixed a defect where, when authentication is enabled, the persistence service could abruptly exit.
FTL-10332	Fixed a defect where the persistence service would sometimes abruptly exit while printing a message.
FTL-10328	Fixed a defect where a server removed from a cluster sometimes caused the quorum protocol to abruptly exit on heartbeats.
FTL-10286	Fixed a defect where an FTL server in a cluster sometimes fails before a quorum is first formed.
FTL-10258	Fixed a client library defect that caused a client to abruptly exit when reconnecting to a persistence service.
FTL-10257	Fixed a defect that caused the realm service in the core servers to abruptly exit when it loses heartbeats from an affiliate.
FTL-10243	Fixed a defect where restarted servers in a cluster sometimes fail to form a quorum and abruptly exit.

Key	Summary
FTL-10180	The ftlstart script from the samples/scripts directory no longer deletes the auto-generated YAML configuration file.
FTL-10093	Fixed a defect in the realmservice that caused ActiveSpaces tibdg process from getting status of ActiveSpaces nodes and proxies.
FTL-10090	Fixed a defect that caused the persistence service to abruptly exit upon shutdown, instead of a clean shutdown.
FTL-10049	Fixed a defect where an FTL server sometimes did not run required services.
FTL-10047	Fixed a defect where the primary realm service would sometimes stop processing satellite server connect requests.
FTL-9957	Fixed a Windows Settings defect where the Uninstall buttons for FTL and eFTL were unresponsive.
FTL-9956	Fixed a memory leak in the client library where, as seen with SubscribeToInbox, excessive subscribers are created.
FTL-9950	Fixed an FTL Server defect that prevented from it parsing the YAML configuration.
FTL-9941	Fixed a defect where an eFTL server failed to start due to an incorrect max log file size parameter.
FTL-9935	Fixed a defect where sometimes after long quorum disruption, subscribers were timed out early.
FTL-9930	Fixed a defect where sometimes a client would abruptly exit while reconnecting to an FTL server.
FTL-9901	Fixed a defect where persistence servers sometimes reported their status as out-of-sync.
FTL-9814	Removed the occurrence of ??? in all server output logs.

Key	Summary
FTL-9770	Fixed a defect in which the UI and the Web API allowed a user to change the durable/template type after it was deployed.
FTL-9745	Fixed a defect where an endpoint with no associated store/template incorrectly used a server based transport, but no validation error was generated
FTL-9684	Improved UI performance when there are large number of clients.
FTL-9360	Fixed a defect where a client used a *persistent* HTTP connection and sent *multiple* REST requests that the FTL server routes to different target services (for example, realm service and eFTL service) and some requests then failed with HTTP error 404 page not found.
FTL-9138	auxiliary persistence servers do not get the realm server available/unavailable advisory anymore.
FTL-8716	Fixed a defect where durable subscribers could throw an exception while acknowledging messages after an extended network disconnect from the persistence service. This symptom sometimes affected shared durables and standard durables with prefetch.
FTL-8464	Fixed a persistence service defect that could cause it to abruptly exit during shutdown.
FTL-7959	Fixed a defect in the client library that could cause a client using a DTCP transport to abruptly exit.

Release 6.1.0

Key	Summary
FTL-9864	Fixed a memory leak associated with durable subscribers.
FTL-9687	Fixed a defect in which the realm service could erroneously omit clients from dynamic TCP transports.
FTL-9766	Modified a default value for new eFTL clusters to allow unlimited client

Key	Summary
	connections.
FTL-9751	Fixed a defect in which it was possible to circumvent realm validation when defining or modifying a static durable or a dynamic durable template.
FTL-9724	Improved memory performance of the realm service when managing dynamic TCP transport buses.
FTL-9723	Modified the default parameters for new persistence stores and durable templates in ftl.default.cluster for improved performance.
FTL-9651	Fixed a defect in which realm validation erroneously allowed hub-and-spoke zones without a hub cluster.
FTL-9627	Improved throughput performance of the client API.
FTL-9626	Fixed a defect in which simultaneous realm deployment operations could interfere with one another.
FTL-9611	Fixed a defect in which the FTL server incorrectly parsed IPv6 literal addresses in its configuration file.
FTL-9607	Improved performance of TCP and dynamic TCP transports.
FTL-9584	Fixed a client library defect in which creating an inbox subscriber on an endpoint that disables the receive inbox ability could cause the client to abruptly exit.
FTL-9582	Fixed a Go language client library defect in which clients could abruptly exit in the method Message.String().
FTL-9575	Improved error reporting during parsing the FTL server configuration file.
FTL-9568	Fixed a Go language API defect in which calling Realm.Unsubscribe() on an uninitialized realm object could cause programs to abruptly exit.
FTL-9562	Fixed a memory leak in tibmongateway.

Key	Summary
FTL-9557	Improved performance of FTL servers with many clients.
FTL-9544	Fixed a defect in which the FTL server erroneously reported the status of some internal clients.
FTL-9524	Fixed a defect in which the monitoring gateway could abruptly exit while communicating with the Prometheus pushgateway.
FTL-9515	Fixed a client library defect in which the realm server could abruptly exit while attempting to get a non-existent field from a message.
FTL-9512	Fixed a client library defect in which internal clean-up of dynamic formats could interfere with event queue timers.
FTL-9510	Fixed a client library defect in which client monitoring API and GUI could omit dynamic formats that begin with an underscore character.
FTL-9507	Fixed a memory leak in the client library affecting durable subscriptions.
FTL-9500	Fixed a client library defect that prevented 6.x clients from connecting to realm servers from Release 5.4 and earlier.
FTL-9497	Fixed a persistence service defect in which clients did not failover to a new persistence leader after network segmentation.
FTL-9495	Fixed a defect that affected durable discard behavior.
FTL-9493	Fixed a defect in which the modifications to the definition of the default persistence cluster, ftl.default.cluster, were not persisted to the realm database. After stopping all of the core FTL servers, then restarting them, those modifications were lost.
FTL-9492	Fixed a defect in which FTL server could exhaust file descriptors and so could not accept client connections nor service REST requests.
FTL-9491	Fixed a defect in which the realm service could abruptly exit while outputing log messages.

Key	Summary
FTL-9452	Fixed a defect in which a persistence service could abruptly exit after reconnecting to its cluster following a period of disconnection.
FTL-9418	Fixed a defect in which realm validation erroneously allowed an application endpoint associated with a wide-area store but not associated with a dynamic durable template.
FTL-9356	Fixed a defect in which client programs could erroneously publish to internal stores.
FTL-9338	Fixed Go language API defects in which constants were misspelled. For more detail, see . If you copied the incorrect names of these constants into your Go program code, correct them before migrating to Release 6.1 or later.
FTL-9336	Improved memory usage in the realm service.
FTL-8930	The realm connect call now emits warning logs when it cannot connect to an FTL server.
FTL-8724	Fixed a GUI defect in which the zones grid misreported the number of non-forwarding clusters.
FTL-8511	Fixed a defect in which multicast transports could erroneously not transmit backlog outbound message data packets.
FTL-8507	Fixed a defect in which multicast transports could incorrectly sequence outbound message data packets. This symptom could occur if the transport send rate parameter was unlimited.

Release 6.0.1

Key	Summary
FTL-9463 FTL-9430	Improved scalability of client connections.

Key	Summary
FTL-9444 FTL-9416	Fixed a defect in which the GUI and REST API did not include wide-area stores in the list of persistence stores.
FTL-9434	Fixed a GUI defect affecting the durable interest field.
FTL-9429 FTL-9428	Fixed a defect in which the FTL server could update heartbeat values but continue to timeout clients according to old values.
FTL-9417	Fixed a memory leak on Windows platforms triggered when clients connected to the FTL server.
FTL-9402	Fixed a defect in which a restarted FTL server could erroneously delete a dynamic durable template.
FTL-9395	Fixed a client library defect in which FTL servers or clients could abruptly exit after persistence services reformed a quorum.
FTL-9394	Fixed a defect in which realm services and eFTL services could not start because of port collisions.
FTL-9391	Fixed a defect in which the GUI could list the status of a wide-area persistence store as *unknown*.
FTL-9389	Fixed a GUI defect in which purging wide-area durables was ineffective.
FTL-9357	Fixed a defect in which the persistence service did not automatically create the data directory if it was different from the savedir.
FTL-9324	Fixed a defect in which the FTL administration utility did not print a label when it prompted for a password.
FTL-9322	Fixed a defect in which the FTL administration utility printed status fields that are no longer relevant.
FTL-9300	Fixed a defect in which FTL servers that could not reach their quorum leader could log many warning messages, such as the following:

Key	Summary
	warn ftl: Error publishing DTCP Transport Join notification message: unavailable
FTL-9273	Fixed an issue in which an FTL server that restarted could hold state information about clients that had stopped or disconnected. The monitoring GUI and web API could report incorrect state about those clients.
FTL-9271	Fixed a defect in printing the names of auxiliary servers.
FTL-9259	Fixed a defect in which FTL servers could output log messages such as the following: warn ftl: bad role ftl-primary Invalid message type received from peer 'NULL'
FTL-9224	Clarified logging output during migration from Release 5.x to Release 6.x.
FTL-9217	Fixed a defect on Windows platforms in which a cluster of FTL servers could become unstable if the data directory were a network drive.
FTL-9194	Fixed a defect in which an FTL server that restarted could hold state information about clients that had stopped or disconnected. The monitoring GUI and web API could report incorrect state about those clients.
FTL-9078	Fixed a GUI defect in the cluster details panel in which users could appear to remove an externally reachable address even when not in edit mode.
FTL-9060	Fixed a client library defect in which applications and services using dynamic TCP transports could abruptly exit.
FTL-8949	Fixed a defect in which the GUI appeared to permit purging of internal persistence stores.

Key	Summary
FTL-8918	Fixed a defect in which the GUI varied the order of services in the server status pane.
FTL-8905	Fixed a misleading warning message when starting an empty data store.
FTL-8541	Fixed a client library defect in which applications and services could abruptly exit after the Close cleanup call.
FTL-8893	Fixed a defect in which the FTL server GUI could display hidden internal endpoints even when the checkbox option "Show hidden FTL Objects" was cleared.
FTL-8308	Fixed an defect in which the web API command to compact the realm server's database was not effective on Windows platforms.

Release 6.0.0

Key	Summary
FTL-9275	Fixed a defect in which clients using dynamic TCP transports could not reconnect to the realm server after the realm server had been restarted. In some instances, clients became unresponsive. This symptom could also affect services that used dynamic TCP transports.
FTL-8551	Fixed a defect in which monitoring data from TIBCO ActiveSpaces was not available for display.
FTL-8521	Clarified the documented behavior of the FTL administration utility.
FTL-8505	Fixed a defect in which group members could miss status updates.
FTL-8491	Fixed a defect in which new DTCP clients could exhibit significant delays in joining the bus, while they attempted to connect with defunct DTCP clients.
FTL-8472	Fixed a defect in which clients for which centralize logging is enabled could abruptly exit.
FTL-8470	Fixed a defect in which a multicast subscriber could erroneously miss initial

Key	Summary
	messages from a sender.
FTL-8410	Fixed a defect in which clients that collect subscription statistics could abruptly exit.
FTL-8403	Fixed a misleading error string that was output when exceeding a message redelivery count.
FTL-8401	Fixed a Java client library defect in which FTL.setLogHandler could cause the client to abruptly exit.
FTL-8390	Fixed a defect in which exception strings could contain unreadable characters.
FTL-8374	Fixed a defect in which publisher, subscriber, and map calls rejected the persistence retry value -1.
FTL-8371	Fixed a API defect that affected the persistence retry duration properties in calls to publisher and subscriber methods.
FTL-8240	Relaxed a constraint: application clients no longer require restart after administrators modify or remove a mapping from a subscriber name to a durable name.
FTL-8171	Fixed a log service defect related to incorrect parameters.
FTL-8142	Fixed a log service defect related to unrecognized parameters.
FTL-8092	Fixed a log service defect related to realm server restart.

Known Issues

The following issues exist in this release of TIBCO FTL®:

Key	Summary
FTL-13960	**Summary:** Sending extremely large messages, for example 1 GB messages, could disrupt the FTL persistence quorum. **Workaround:** Increase the pserver_timeout_pserver for the affected persistence cluster.
FTL-13934	**Summary:** When an extremely large disk persistence file is compacted, the persistence service may experience a brief stall. **Workaround:** If necessary, increase client_timeout_pserver and/or pserver_timeout_pserver.
FTL-13924	**Summary:** The disk usage limit feature (max.disk.fraction) does not consider messages swapped onto disk from non-replicated stores. When disk persistence is enabled, messages in replicated stores are written to the persistence service's data directory. If non-replicated messages are swapped to the same disk, the persistence service could exceed max.disk.fraction when writing. **Workaround:** Configure appropriate storage limits (byte limit and/or message limit) for all stores. Or, use the swapdir parameter in the FTL server yaml file to swap non-replicated messages to a different disk.
FTL-13923	**Summary:** This issue is related to either of the following scenarios: • Swapping non-replicated stores when disk persistence and disk swap are enabled

Key	Summary
	• Swapping replicated and non-replicated stores when only disk swap is enabled Swapping may fail if the space required is not adequate. There is no control on the size of the swap file. Swapping does not look at the max.disk.fraction. Even with max.disk.fraction enabled, you still need to configure reasonable byte limits and message limits at the cluster level, store level, or both. **Workarounds:** Ensure you have adequate storage space. Assign the swap file to another disk using the swapdir parameter. For details, see Persistence Service Configuration Parameters, swapdir.
FTL-13790	**Summary**: After upgrading one FTL server to 6.9.0 or later, another FTL server at 6.8.0 or lower may log a message like the following: Error processing cluster message: invalid type. **Workaround**: None. There is no functional impact. The warning may be ignored.
FTL-13411	**Summary**: On macOS, when the installation package is downloaded through a web browser, it may get labeled as quarantined by the operating system. Installation may result in a system prompt stating that the package cannot be opened. **Workaround**: Remove the quarantine flag from the package before installing it. For example: 6.9.0 and later: xattr -d com.apple.quarantine TIB_ftl_6.10.1_macos_x86_64.pkg Before 6.9.0: xattr -d com.apple.quarantine TIB_ftl_6.8.0_macosx_x86_64.pkg
FTL-13171	**Summary**: The use_endpoint_store_for_inbox and enable_

Key	Summary
	permissions features require 6.8.0 or later clients. If an older client is sending or receiving inbox traffic on an endpoint with no direct path transports, and either feature is enabled, the older client should raise an exception during the realm deployment. The older client either accepts the deployment or replies with needs restart. In both cases, the older client is not functional and must be upgraded to 6.8.0 or later. **Workaround**: Upgrade clients to 6.8.0 or later when using the use_endpoint_store_for_inbox and enable_permissions features.
FTL-13170	**Summary**: The realm property use_endpoint_store_for_inbox requires 6.8.0 or later clients. If this property is set to true, and a client older than 6.8.0 attempts an inbox send on an endpoint with no direct-path transports, the old client crashes rather than raising an appropriate exception. **Workaround**: Upgrade clients to 6.8.0 or later when using the use_endpoint_store_for_inbox feature.
FTL-12794	**Summary**: In the Go API in FTL 6.8.0, a submessage can be set in with two different value types: 1. ftl.Message 2. MsgContent map[string]interface { } A MsgContent is returned by: func (m Message) Get (fields ...string) (c MsgContent, err error) If the message is a submessage field the c MsgContent has another MsgContent for the corresponding submessage. When the second method is used to set a submessage and the application is configured to manage all formats, then the APIthrows an exception like the following: **Exception**:

Key	Summary
	TIBEX_SET_ERROR(e, TIB_NOT_PERMITTED, "Format is not defined for " "this application and all formats must be managed " "- cannot create a dynamic format."); **Workaround**: *Assumptions*: 1. The configuration is set to manage_formats = true or manage_all_formats = true. 2. The Go client created a message with a statically configured format that has a submessage: msg, _ := realm.NewMessage("outermessage") subMsg, _:= realm.NewMessage("innermessage") *Procedure*: To create the correct formats for the message and submessage (example): 1. Set some fields: subContent, _ := subMsg.Get() subContent["someLongField"] = 123455678940000 content, _ := msg.Get() content["longname-field"] = 256 content["submessage-field"] = subMsg content[FieldNameString] = "internal msg" msg.Set(content) 2. Add one more field as follows: allcontent, _ := msg.Get() allcontent["newFieldString"] = "newly added field value" msg.Set(allcontent) We have created the correct formats for the message and submessage. The last msg.Set() operation fails even though the correct formats are created.
FTL-12619	**Summary**: FTL 6.4 clients that have endpoints configured

Key	Summary
	with server-based transports cannot use request/reply calls or request/reply inboxes with FTL 6.5 or later clients. **Workaround**: Upgrade the clients to 6.5 or later.
FTL-12580	**Summary**: Although the API to backup a realm database (api/v1/server command backup) returns 200 on success, the backup has not necessarily completed when the request returns. The response body should contain a status code of 202. The status code of the HTTP response itself is 202, which correctly indicates the semantics of the backup request, namely that the backup may not have been completed when the response is returned. **Workaround**: None.

FTL-12517	**Summary**: On Linux Platforms, if you have *both* FTL 6.7.0 and eFTL 6.7.0 installed and are using yum or zypper package managers to upgrade to FTL/eFTL 6.7.1, the upgrade procedure can fail.
	
	Workaround: When installing, follow these steps:
	
	1. Install FTL/eFTL 6.7.1 together.
	
	For yum:
	
	`yum install -y TIB_ftl_6.7.1/rpm/*.rpm TIB_eftl_6.7.1/rpm/*.rpm`
	
	For zypper:
	
	`zypper install TIB_ftl_6.7.1/rpm/*.rpm TIB_eftl_6.7.1/rpm/*.rpm`
	
	2. Manually move
	/opt/tibco/eftl/6.7/bin/modules/tibeftlserver
	to
	/opt/tibco/ftl/6.7/bin/modules/tibeftlserver

FTL-12181	**Summary**: For the Administrative GUI, resizing the window

Key	Summary
	sometimes hides the vertical scrollbar. **Workaround**: Resize to a larger window or avoid resizing.
FTL-11597	**Summary**: If an FTL client creates many subscribers on durables using async acks and a low ack batch time, the client experiences high CPU usage. **Workaround**: None.
FTL-11185	**Summary**: If the cluster message swapping setting `disk_mode` is set to swap and the disk is being accessed via NFS, client or quorum timeouts of up to a few seconds can occur. **Workaround**: Use a local filesystem for the location of swap files, via the new cluster message swapping setting `swapdir`.
FTL-10631	**Summary**: Inbox subscribers do not explicitly acknowledge message delivery if the subscriber's endpoint has no direct path transports associated. *Note:* This applies to FTL releases prior to 6.8.0 or if `use_endpoint_store_for_inbox` is set to false. **Workaround**: None.
FTL-10393	**Summary:** FTL 6.3.x or later versions are not compatible with EMS 8.5.0 or earlier releases. **Workaround**: Upgrade to EMS 8.5.1.
FTL-10335	**Summary**: When 6.2 or later clients connect to a 5.4 realm server, the realm server logs a panic. The 5.4 realm server does not crash, and is functional after it logs a panic. 6.2 clients are able to successfully connect. **Workaround**: Upgrade all FTL servers before upgrading clients.
FTL-9499	**Summary**: Importing realm definition data into Release 6.0.1 that had been output in JSON format from Release 6.0.0

Key	Summary
	could cause the FTL server to reject the deployment even though it did not report validation errors. **Workaround**: From _GroupServer application, remove the definitions of: _clientEndpoint _inboxEndpoint _loggingEndpoint _monitoringEndpoint
FTL-9293	**Summary**: During migration from Release 5.4 to Release 6.x, it is possible that the old realm server and the new FTL server could both assign ordinals to group members. This could result in thrashing behavior by the group members. **Workaround**: Immediately stop the old 5.4 realm server.
FTL-9281	**Summary**: The REST command to compact an FTL server database is not functional. **Workaround**: None.
FTL-9231	**Summary**: When FTL servers are not in a quorum, the GUI displays incorrect monitoring data. **Workaround**: None
FTL-8319	**Summary**: On Windows platforms, after silent installation of one installation type, subsequently installing with a different installation is ineffective. **Workaround**: Completely uninstall the previous installation type, and reinstall with a new installation type.
FTL-8280	**Summary**: Subscribers on the monitoring endpoint could miss the final monitoring metrics from a closing client. **Workaround**: None.

Key	Summary
FTL-7718	**Summary**: Debian Linux Uninstall After uninstalling Debian Linux packages, the command `dpkg -query` reports that the package `tibco-ftl-thirdparty` is still installed (even though it has, in fact, been successfully uninstalled). **Workaround**: This command resolves this issue by removing package information from the database. `sudo dpkg --purge tibco-ftl-thirdparty`
FTL-7161	**Summary**: The realm server GUI does not support Internet Explorer (IE) 11 and earlier. However, it does support Edge (Windows 10). **Workaround**: Use any supported browser as listed in the file: readme.txt.
FTL-5630	**Summary**: Changing a Persistence Cluster If you change the name of a persistence cluster, all running persistence servers in the cluster require restart. However, the realm server GUI does not detect this condition. **Workaround**: Ensure that you restart persistence servers after changing their cluster's name.
FTL-4386	**Summary**: Chrome and Safari browsers can no longer access TIBCO HTML documentation from a file system, that is, using the `file://protocol`. **Workaround**: Access using a different browser, or access the HTML documentation through the web, that is, using the `http://protocol`.
FTL-496	**Summary**: On Microsoft Windows platforms (only), `tcp` and `shm` transports support at most 60 simultaneous connections. For example, when 60 tcp transports are connected to one listening transport in an application program (for example, a server hub application), the 61st

Key	Summary
	cannot connect, and FTL logs an error. Similarly, when shm transports in 60 application processes on the same host computer are connected to the same shared memory segment, the 61st cannot connect, and FTL logs an error. When inline mode is disabled, this limitation of 60 connections applies separately to each transport. However, when inline mode is enabled - by using the property TIB_EVENTQUEUE_PROPERTY_BOOL_INLINE_MODE when creating an event queue - then this limitation applies cumulatively across all the transports of all the endpoints (that is, subscribers) associated with that queue. The sum of all the connections to those transports cannot exceed 60. **Workaround**: None.

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 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.

Product-Specific Documentation

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

TIBCO FTL® Documentation Set

The following documents can be found on TIBCO FTL Product Documentation as Web Help or PDFs.

- *Installation*: Read this guide before installing or uninstalling the product.
- *Concepts*: Review this guide for an introduction to FTL software fundamentals.
- *Quick Start*: Use this guide to quickly start FTL and send and receive a message.
- *Getting Started*: Use the guide to set up, start, and run various TIBCO FTL sample programs that demonstrate typical messaging functionality.
- *FTL Tutorials*: Use this guide for a step-by-step approach to build TIBCO FTL applications. You use options and properties, exception handling, programming models, and the user interface for configuration and monitoring.
- *Administration:* Administrators read this manual to learn how to use the FTL server, its interfaces, and its services, and how to define a realm. Developers can also benefit from understanding FTL software from an administrator's perspective.

- *Development*: Application developers and architects read this manual to understand concepts relevant in any supported programming language.
- *Monitoring*: Administrators read this manual to learn about monitoring and metrics. Developers read this manual to learn how an application can subscribe to the stream of monitoring data.
  Important: As of FTL 6.10.0, TIBCO® Messaging Monitor for TIBCO FTL® must be used. This replaces the FTL monitoring (monitoring directory) including Grafana and tibmongateway which are removed. Monitoring metric types are also removed as of FTL 6.10.0.
- *Shifting to FTL:* This manual contrasts TIBCO FTL with TIBCO Enterprise Message Service™, and offers suggestions to smooth your transition to TIBCO FTL. Application developers, architects, and administrators familiar with TIBCO Enterprise Message Service read this manual.
- *Security*: This manual contains security-related tasks for administrators and security tips for application developers.
- *API Documentation*: Application developers use this documentation to learn the details of the FTL API in specific programming languages. This is available as Web Help only.
- *TIBCO FTL Glossary:* The glossary contains brief definitions of key terms used in all other parts of the documentation set.
- *Release Notes:* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

Additional information resources can be found, after file extraction, in the samples directory. These include a Getting Started guide, Tutorials, readme.txt files, and sample applications.

Updated Resources on TIBCO Community

Supplemental resources are now distributed at the TIBCO FTL Community Wiki in the Reference Info tab. You can always find the latest versions of these resources in that location.

Those resources include *TIBCO FTL Getting Started Guide* and *TIBCO FTL Tutorials*. They also include sample FTL server configuration files and sample realm definition files.

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 TIBCO Support

Get an overview of TIBCO Support. You can contact TIBCO Support in the following ways:

- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support website.
- 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 TIBCO Support website. 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 TIBCO Community.

Legal and Third-Party Notices

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: https://scripts.sil.org/OFL

This 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 this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.