

# faircom **edge** V3.0.1

H I G H L I G H T S



FairCom®

# Edge V3.0.1 Highlights Summary

FairCom is pleased to share the following enhancements on top of the FairCom EDGE V3.0 release made available in November 2020. These enhancements are available with FairCom EDGE with build dates of 210827 (YYMMDD format) or later.

## ThingWorx

FairCom's plug-in solution for ThingWorx has always supported mapping any data to any "Thing" within the ThingWorx ecosystem, thereby leveraging ThingWorx ability to create Digital Twins. This support has now been expanded to support mapping concurrently to Multiple Things.

### Multiple ThingWorx "things"

The ThingWorx plug-in can now map a FairCom EDGE instance to multiple "things" on the ThingWorx platform making it possible for a business to create a digital twin for each of its customers and map a subset of data in FairCom EDGE to each customer.

As shown in the following example, Mike's car can be associated with two Things, the location where service is provided (Car Dealer), and with the Digital Twin of Mike's car.



[Learn more...](#)



## ThingWorx Store and Forward

This release also has a new powerful persistent data store. FairCom EDGE's Store & Forward solution is critical for ensuring data integrity across an application. In a complex IIoT scenario (think factory floor, smart city, etc.), being able to rely 100% on communication links is not realistic. When some level of outage occurs, being able to safely secure the message is a must. The Store & Forward concept provides the peace of mind that the information needed by the next component in the IIoT ecosystem will be available once the communication link is reestablished. Store & Forward maintains a record of the last message received. Once communication is reestablished, any messages that haven't been delivered will automatically be sent. This brings an unsurpassed level of data integrity to the ThingWorx platform.

[Learn more...](#)

## Automatic Reconnection to ThingWorx

When the FairCom EDGE instance loses its connection to the ThingWorx platform (because of the network, device outage, or any other reason), the FairCom EDGE server instance automatically reconnects.

[Learn more...](#)

# MQTT

FairCom EDGE's MQTT V3 support has been expanded to include:

- **Store and Forward**
- **MQTT Explorer Improvements**

## MQTT Store and Forward

For specifically configured topics, the database can store MQTT messages and forward them to subscribers. These topics are called "Store & Forward Topics." This feature enhances MQTT to guarantee delivery for specific topics even when a subscriber is unavailable to receive them.

When an existing subscriber to a Store & Forward Topic becomes unavailable and later becomes available, the MQTT Broker forwards messages to the subscriber starting after the last successfully delivered message.

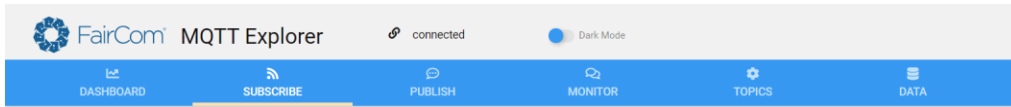
[Learn more...](#)



## MQTT Explorer Improvements

The FairCom EDGE MQTT Explorer has been enhanced in this release to make you even more productive. The built-in explorer has been expanded with the following enhancements:

The new toolbar matches your workflow.



The new dashboard shows subscribers, publishers, topics, and message throughput.

The dashboard displays the following data:

Topic	Message Count	Avg Per Minute	Avg Per Hour	Avg Per Day
mqtt/test10	219			
mqtt/test11	178			
mqtt/test12	43	1	11	111
mqtt/test13	107	2	27	60
mqtt/test14	240	5	61	160
mqtt/test15	188	4	48	228
mqtt/test16	196	4	50	146
mqtt/test17	208	4	53	927
mqtt/test18	183			
mqtt/test19	22			

The detailed view for **mqtt/test14** shows:

- Message Count: 240
- Avg Per Minute: 5
- Avg Per Hour: 61
- Avg Per Day: 160



Subscribe to messages coming into the MQTT Broker for monitoring messages, for troubleshooting, etc.

The screenshot shows the 'SUBSCRIBE' tab in the FairCom MQTT Explorer. The interface includes a navigation bar with 'DASHBOARD', 'SUBSCRIBE', 'PUBLISH', 'MONITOR', 'TOPICS', and 'DATA'. The main area has a 'Topic' dropdown set to 'display3', a 'QoS' dropdown set to 'QoS 2 (exactly once)', and a 'Topic Color' dropdown set to '009900'. Below these are four subscription cards, each with a 'QoS: 2' label and an 'X' icon. The cards are labeled 'ctreeAdministration', 'device2', 'display3', and 'sensor1'. A copyright notice 'Copyright 2021 FairCom Corporation. All rights reserved.' is visible at the bottom.

### Publish messages

The screenshot shows the 'PUBLISH' tab in the FairCom MQTT Explorer. The interface includes a navigation bar with 'DASHBOARD', 'SUBSCRIBE', 'PUBLISH', 'MONITOR', 'TOPICS', and 'DATA'. The main area has a 'Topic' dropdown set to 'test', a 'QoS' dropdown set to 'QoS 2 (exactly once)', and a 'Retain' dropdown set to 'No'. Below these is a large text area for the message content, which contains a JSON object: 

```
{
  "myProperty": "myValue",
  "myArray": [ 1,2,3,4],
  "myObject": {
    "myBooleanProperty": true,
    "myNumberProperty": 123.456
  },
  "myArrayOfObjects": [
    { "prop1": true, "prop2": "2021-03-21" },
    { "prop1": false, "prop2": "2021-04-17" }
  ]
}
```

 A copyright notice 'Copyright 2021 FairCom Corporation. All rights reserved.' is visible at the bottom.



### Monitor received messages

The screenshot shows the 'MONITOR' tab in MQTT Explorer. It displays a list of messages from subscribed topics. The first message is from 'test' at 'Apr 28 2021 09:13:22' with QoS: 2 and Retained: No. Its payload is a JSON object: {"myProperty": "myValue", "myArray": [1,2,3,4], "myObject": ...}. The second message is from 'test' at 'Apr 28 2021 09:12:58' with QoS: 2 and Retained: No. Its payload is a JSON object: {"method": "createIndex", "params": {"database": "nameOfDatabase..."}.

### Manage topics

The screenshot shows the 'TOPICS' tab in MQTT Explorer. It features a table titled 'Persistence Topics' with columns: Topic Name, Table Name, Database Name, and Database Connection. The table contains three entries: 'anotherTopic', 'someTopic', and 'someTopic'. The 'someTopic' entry is selected. To the right of the table is a JSON configuration editor with a preview of the configuration for the selected topic, including fields like 'persistenceTopic', 'databaseConnectionString', 'databaseUserName', 'tableName', 'tableAutoTimeStamp', 'tableAutoTimeStampIndex', 'tableReplicationReady', 'extractedLibraryName', and 'mapOfPropertiesToFields'.

### Manage data

The screenshot shows the 'Create Persistence Table' dialog box. It has a title bar with a close button. Below the title bar is a section titled 'DATA MAPPER' with a subtitle: 'To save the entire contents of an MQTT message directly into a table, do not map any JSON properties to table fields.' Below this is a table with the following columns: Property Path, Field Name, Field Type, Field Width, Field Scale, Property Format, and Field Format. The table contains four rows of data:

Property Path	Field Name	Field Type	Field Width	Field Scale	Property Format	Field Format
robot.temperature	temperature	DOUBLE	0	0		
robot.humidity	humidity	NUMBER	0	2		
robot.action.count	count	INTEGER	0	0		
error	alert	VARCHAR	65500	0	stringify	

At the bottom of the dialog, there is an 'ADD +' button and navigation buttons: '<- PREV' and 'NEXT ->'.



## Plug-in Interface Enhancements

The FairCom EDGE plug-in support has been enhanced by providing multiple interfaces for starting, stopping, and making calls to FairCom plug-ins on-the-fly.

### ctadmn Command-Line

You can call **ctadmn** as a configurable command-line utility to start and stop the plug-in using the following syntax:

```
ctadmn -s <server name> -u <user name> -p <password> -c <command>
```

### ctadmn Interactive (Option 10)

Dynamically load a plug-in on demand after c-tree Server has started up:

1. Execute the **ctadmn** utility.
2. Select option 10, Change Server Settings.
3. Again select option 10, Change the specified configuration option.
4. Enter the configuration option and its value:  

```
>> PLUGIN cthttpd;./web/cthttpd.dll  
Successfully changed the configuration option.
```

### ctPlugin Function Call

A new function, named **ctPlugin**, can be used on the client-side to programmatically start and stop plug-ins. It has the following signature:

```
NINT ctPlugin(ctPLUGIN_COMMAND command, pTEXT inputBuffer, pTEXT outputBuffer, pVLEN pOutputBufferSize);
```

where:

- *command*: currently, the options are *ctPLUGIN\_START* or *ctPLUGIN\_STOP* (defined by the *ctPLUGIN\_COMMAND* enum type)
- *inputBuffer* is flexible; currently, it expects only the plug-in name (loaded in the server by the *PLUGIN* keyword in *ctsvr.cfg*)

### ctSETCFG Function Call

A function call is available for programmatically starting and stopping a plug-in. Use the same *PLUGIN* configuration option syntax that you would use in *ctsvr.cfg* in a call to **ctSETCFG()**.

```
ctSETCFG(setcfgCONFIG_OPTION, "PLUGIN cthttpd;./web/cthttpd.dll");
```

## Android Installation

The FairCom documentation now provides a tutorial with detailed steps to install and test FairCom EDGE on an Android IoT platform. This tutorial focuses on emulators; the changes to install on a real device should be minimal.

[Learn more...](#)