

Zebra MQTT

API Guide



ZEBRA

ZEBRA and the stylized Zebra head are trademarks of Zebra Technologies Corporation, registered in many jurisdictions worldwide. All other trademarks are the property of their respective owners.
© 2024 Zebra Technologies Corporation and/or its affiliates. All rights reserved.

Information in this document is subject to change without notice. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of those agreements.

For further information regarding legal and proprietary statements, please go to:

SOFTWARE: zebra.com/linkoslegal

COPYRIGHTS: zebra.com/copyright

PATENTS: ip.zebra.com

WARRANTY: zebra.com/warranty

END USER LICENSE AGREEMENT: zebra.com/eula

Terms of Use

Proprietary Statement

This manual contains proprietary information of Zebra Technologies Corporation and its subsidiaries (“Zebra Technologies”). It is intended solely for the information and use of parties operating and maintaining the equipment described herein. Such proprietary information may not be used, reproduced, or disclosed to any other parties for any other purpose without the express, written permission of Zebra Technologies.

Product Improvements

Continuous improvement of products is a policy of Zebra Technologies. All specifications and designs are subject to change without notice.

Liability Disclaimer

Zebra Technologies takes steps to ensure that its published Engineering specifications and manuals are correct; however, errors do occur. Zebra Technologies reserves the right to correct any such errors and disclaims liability resulting therefrom.

Limitation of Liability

In no event shall Zebra Technologies or anyone else involved in the creation, production, or delivery of the accompanying product (including hardware and software) be liable for any damages whatsoever (including, without limitation, consequential damages including loss of business profits, business interruption, or loss of business information) arising out of the use of, the results of use of, or inability to use such product, even if Zebra Technologies has been advised of the possibility of such damages. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Publication Date

December 23, 2024

Contents

About This Document	6
Introduction	6
Chapter Descriptions.....	6
Notational Conventions	7
Icon Conventions	7
Related Documents and Software	7
Description	8
General Information	8
Key Value Pair Names	9
MQTT Support	9
SGD Printer Settings	10
mqtt-enable	10
mqtt-connX-server_address	10
mqtt-connX-mqtt_version	11
mqtt-connX-tenant_id	11
mqtt-connX-username.....	11
mqtt-connX-password	12
mqtt-connX-qos.....	12
mqtt-connX-clean_session_flag	12
mqtt-connX-retry_interval_random_max	12
mqtt-connX-ping_interval	13
mqtt-restore_defaults	13
mqtt-connX-reset_now	13
mqtt-connX-reset_required.....	13
mqtt-logging-entries	14
mqtt-logging-max_entries.....	14
mqtt-logging-clear	14
device-zuid	14

Contents

ip-firewall-proxy	15
ip-firewall-authentication-add	15
ip-firewall-authentication-remove	15
ip-firewall-authentication-entries	16
alerts-send_current_status_alerts	16
Printer Security Files.....	17
E:MQTTx_CA.NRD	17
E:MQTTx_CERT.NRD	17
E:MQTTx_KEY.NRD	17
Example.....	18
Topics	19
Subscribe	19
Command	19
Publish	19
Connected	20
Response	21
Notification	23
Methods.....	24
Command Methods.....	24
Provisioning	25
File.....	29
Settings.....	33
Notification Methods.....	34
Alert	35
File.....	38
CSR_ready	40
Error.....	41
Logging	42
Printer (Syslog)	42
MQTT Specific	42
Connecting to SOTI Connect.....	43
Before You Get Started.....	43
Printer Configuration	43

Instructions 43

Connecting to HiveMQ Cloud TLS 8883 45

- Before You Get Started 45
- Setup the HiveMQ Cloud Cluster 45
- Connect a Client to the Broker 47
 - Instructions 47
- Printer Configuration 50
- Sending your first command via MQTT 50

Connecting to Mosquitto MQTT TLS 8883 52

- Before You Get Started 52
- Connect a Client to the Broker 52
 - Instructions 52
- Printer Configuration 54
- Sending your First Command via MQTT 54

Connecting to Mosquitto MQTT TLS 8884 56

- Connect a Client to the Broker 56
 - Instructions 56
- Printer Configuration 59

About This Document

Introduction

This guide provides information about how to communicate with Zebra printers using MQTT. It is intended for ISVs and developers who want to write applications that will use MQTT to interact with Link-OS based printers.

Chapter Descriptions

Topics covered in this guide are as follows:

- [Description](#) provides information on key value pair names and supported printers.
- [SGD Printer Settings](#) provides information on configurable settings.
- [Printer Security Files](#) provides information on certificates.
- [Topics](#) provides information on the generic layout of topics.
- [Methods](#) provides information on command and notification methods.
- [Logging](#) provides information on system logs.
- [Connecting to SOTI Connect](#) provides information on connecting to SOTI Connect.
- [Connecting to HiveMQ Cloud TLS 8883](#) provides information on connecting to HiveMQ.
- [Connecting to Mosquitto MQTT TLS 8883](#) provides information on connecting to Mosquitto MQTT TLS 8883.
- [Connecting to Mosquitto MQTT TLS 8884](#) provides information on connecting to Mosquitto MQTT TLS 8884.

Notational Conventions

The following conventions are used in this document:

- **Bold** text is used to highlight the following:
 - Dialog box, window, and screen names
 - Drop-down list and list box names
 - Check box and radio button names
 - Icons on a screen
 - Key names on a keypad
 - Button names on a screen.
- Bullets (•) indicate:
 - Action items
 - Lists of alternatives
 - Lists of required steps that are not necessarily sequential.
- Sequential lists (such as those that describe step-by-step procedures) appear as numbered lists.

Icon Conventions

The documentation set is designed to give the reader more visual clues. The following graphic icon is used throughout the documentation set. This icon and their associated meaning is described below.



NOTE: The text here indicates information that is supplemental for the user to know and that is not required to complete a task.

Related Documents and Software

The following documents provide more information about Zebra printers, commands, and utilities.

- ZPL Programming Guide
zebra.com/content/dam/support-dam/en/documentation/unrestricted/guide/software/zpl-zbi2-pg-en.pdf
- Printer Administration Guide
zebra.com/content/dam/zebra_new_ia/en-us/software-printer/utilities/en/admin-files/printsecure-administration-guide-en.pdf
- Zebra Setup Utilities
zebra.com/setup

For the latest version of this guide and all guides, go to zebra.com/support.

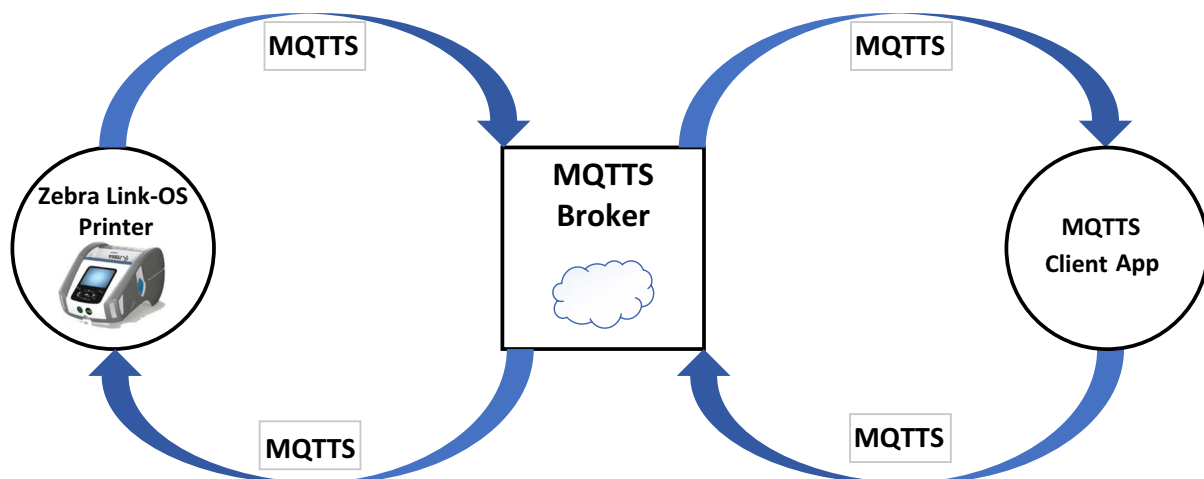
Description

General Information

The intent of this document is to provide customers of Zebra printers information on how to communicate using MQTT. The content will include settings for the purpose of configuring the printer, files needed, as well as the MQTT topic layout. In this document, the term broker and server are used interchangeably with the printer being referred to as the client.



Zebra MQTT only works over a TLS connection to the broker. This is shown in the diagram below as MQTTS.



Any code or examples should strive to be syntactically correct, parse-able, and use the correct quotation marks, for example:

```
{"lorem":"ipsum"}
```

This: "

Not: " or "

Key Value Pair Names

Zebra historically has supported settings and names with a (.) (period) in the name. For maximum compatibility across multiple MQTT brokers we also support (-) (minus). This means that any setting can be specified with either a (.) or a (-) in the setting and the printer will treat it the same. All setting names returns from the printer containing a (-).

MQTT Support

MQTT is supported in Link-OS starting with v6.7. The currently supported printers include:

Industrial	Desktop	Mobile
V92 ZT4x1	V83 ZD510	V91 ZQ5x1
V94 ZE5x1	V93 ZD4x1/ZD6x1	V100 ZQ6x0 Plus
V96 ZT510/ZT6x0		V101 ZQ3x0 Plus/ZR300 Plus
V97 ZT231		

Some model names may sometimes have letters appended at the end of the name based on various printer functionality including but not necessarily limited to C, D, R, T, HC, RH, and LH.

SGD Printer Settings

This section of the manual details the available printer commands that can be used with MQTT. The command structure uses a JSON RPC-like syntax, which can be seen in the examples at the bottom of each command. For experimentation, they can also be used as SGD commands by replacing the dashes (-) in the commands a period (.).

This section assumes familiarity with the list of configurable printer settings. Please see the [ZPL Programming Guide](#) for more information, if needed.

For settings names listed below, the uppercase (X) designates the connection number (1 or 2). The printer allows for two simultaneous MQTT connections, if desired. Any changes to these settings require an MQTT connection reset in order to take effect (see [mqtt-connX-reset_now on page 13](#) for more details). Some settings are considered “protected” in that if protected mode is enabled on the printer, the settings are “locked” at the current value unless the admin user is authenticated.

All settings can be configured using the settings method for the command topic.

mqtt-enable

Description	Control MQTT functionality. When disabled, open connections will be closed. When enabled, a connection will be established if all configuration requirements are met.
Set/get	set/get
Values	on, off
Default value	off
Protected	yes

mqtt-connX-server_address

SGD Printer Settings

Description	Specifies the URL (format <scheme>://<domain>[:port]) to connect the printer client to the MQTT broker. Port number can be included. If no port is specified, the default port for that scheme will be used (mqtts:// is 8883). The printer will not attempt to connect to the same server address on multiple connections or if the value is invalid. This address must be mqtts (a TLS MQTT broker), as non-TLS broker connections are not supported.
Set/get	set/get
Values	Max 2048 characters, for example: mqtts://broker.zebra.com:65412
Default value	<blank>
Protected	yes

mqtt-connX-mqtt_version

Description	Specifies the MQTT version of the broker to connect to, currently only 3.1.1 is supported.
Set/get	get
Values	3.1.1
Default value	3.1.1
Protected	yes

mqtt-connX-tenant_id

Description	Specifies the top level of the Topic to which the printer will subscribe and publish. Example format of the command channel Topic : <mqtt-connX-tenant_id>/<device-zuid>/command
Set/get	set/get
Values	Max. 64 printable non-whitespace ASCII characters except it cannot contain the characters: + # / \$ The minimum size must be 1 character and cannot be blank.
Default value	zebra
Protected	yes

mqtt-connX-username

Description	Specifies the username to use for the MQTT connection number.
Set/get	set/get

Values	Max. 64 characters
Default value	<blank>
Protected	yes

mqtt-connX-password

Description	Specifies the password to use for the MQTT connection number. The get shall NOT return the password.
Set/get	set
Values	Max. 64 characters
Default value	<blank>
Protected	yes

mqtt-connX-qos

Description	Specifies the QoS (quality of service) level to support for that MQTT connection
Set/get	set/get
Values	1, 2
Default value	2
Protected	yes

mqtt-connX-clean_session_flag

Description	Specifies whether to clear all previous values on connection or not for that particular MQTT connection.
Set/get	set/get
Values	on, off
Default value	off
Protected	yes

mqtt-connX-retry_interval_random_max

Description	Specifies the maximum random retry interval to attempt connection to an MQTT broker in case the connection is lost. Used to prevent a fleet of printers from all attempting to connect to a single broker after a network interruption that might cause denial of service (DOS) concerns. This value is in seconds.
Set/get	set/get

Values	1-600
Default value	120
Protected	yes

mqtt-connX-ping_interval

Description	Specifies the mqtt ping interval to keep the connection open. This value is in seconds.
Set/get	set/get
Values	1-300
Default value	30
Protected	yes

mqtt-restore_defaults

Description	This command tells the printer to default only the MQTT settings and no other. Any value is accepted and will restore all MQTT settings.
Set/get	set
Values	<n/a>
Default value	<blank>
Protected	no

mqtt-connX-reset_now

Description	This command tells the printer to reset the current connection to apply the recently changed settings.
Set/get	set
Values	<n/a>
Default value	<blank>
Protected	no

mqtt-connX-reset_required

Description	Any time a setting for MQTT is changed, a reset is required in order for that to take effect. This setting shows if a reset is required in order for any MQTT setting to take effect.
Set/get	get
Values	yes, no
Default value	no
Protected	no

mqtt-logging-entries

Description	An MQTT list of log entries for both mqtt1 (conn1) and mqtt2 (conn2) that are created on printer power up and cleared after power cycle or explicit clearing.
Set/get	get
Values	ASCII string log entries
Default value	no
Protected	no

mqtt-logging-max_entries

Description	Specifies how many entries the MQTT log will have before rolling over. If set to 0, no logging will be present in mqtt-logging-entries.
Set/get	set/get
Values	0-10000
Default value	500
Protected	no

mqtt-logging-clear

Description	Explicitly clears the MQTT log before a power cycle. This is useful if trying to capture specific behavior after a certain time.
Set/get	set
Values	<n/a>
Default value	<blank>
Protected	no

device-zuid

Description	A truly unique ID for the printer that is used for MQTT topic and MQTT client ID. Example format of the command channel topic: <mqtt-connX-tenant_id>/<device-zuid>/command
Set/get	get
Values	<n/a>
Default value	<blank>
Protected	no

ip-firewall-proxy

Description	This setting is applied before making an outgoing HTTP/HTTPs connection in case the printer must go through a proxy server beforehand. If no proxy server, connection specific value is set (such as for Weblink or Alerts), this will act as a general value to apply instead.
Set/get	set/get
Values	This setting has the following string format up to 2048 characters: [http https]://[user:pass@]domain[:port]/[path]
Default value	<blank>
Protected	yes

ip-firewall-authentication-add

Description	Allows the user to add a single server/username/password triplet into the list of authentication entries. This authentication entry is applied before making an outgoing HTTP/HTTPs connection in case the printer must go through an authentication server beforehand. This setting is separate from the proxy setting. The server, username, and password are separated by a single space (not a tab or another whitespace character). The server name is the only required field. If no username is supplied, but a password is, there must be two spaces between the server and the password fields. If there is a username but no password, or simply just the server name, no space is required at the end of the entry. Both DNS names and IP addresses are acceptable.
Set/get	set
Values	The setting has the following space-delimited string format up to 1024 characters: servername[username][password]
Default value	<blank>
Protected	yes

ip-firewall-authentication-remove

Description	Allows the user to remove a single server/username/password triplet from the list of authentication entries. To remove an entry, only the server name is supplied. The entire entry will be removed. If an invalid entry is supplied, no action is taken.
Set/get	set
Values	String server name
Default value	<blank>
Protected	yes

ip-firewall-authentication-entries

Description	Lists the server names added to the authentication entries list delimited by carriage return line feed. Only the server names will be shown. The username and passwords will not be shown.
Set/get	get
Values	String list of servers
Default value	<blank>
Protected	no

alerts-send_current_status_alerts

Description	For the specified destination generate all configured alerts for current status conditions in the printer.
Set/get	set
Values	[destination],[destination_address],[port] destination can be any of the value returned from alerts.destinations destination_address - applies to TCP,UDP,EMAIL,SNMP, SDK, MQTT, and HTTP POST destination types. • Port - Applies to TCP and UDP types The destination must be equal alerts to be generated and sent to that destination.
Default value	<blank>
Protected	no
Example	Example of a match: "MQTT,1" and "MQTT,1" "SNMP,255.255.255.255,162" and "SNMP,255.255.255.255,162"

Printer Security Files

This section describes the files needed on the printer to enable secure MQTT communication. These can be placed on the printer using the printers file transfer commands such as Multi-Part forms. See the example after the three certificate description..

All certificates must be in the PEM format. The signing digest must be SHA-256 or greater. Either RSA or ECDSA ciphers can be used. For RSA, a minimum key size of 2048 is expected. For ECDSA, it is recommended to use a Prime 256 (P-256) curve and the minimum of 224 bits is required. These files must be loaded onto the E: FLASH drive. See the PrintSecure Printer Administration Guide for more details. For those file names listed below, the lowercase 'x' designates the connection number (1 or 2). The printer allows for 2 simultaneous MQTT connections, if desired.

The files can be loaded through any file download mechanism. The CERT & KEY files can also be generated through the printer's CSR generation process. The advantage of the printer's CSR generation process is that the private key never leaves the printer and is not accessible except by the printer. If no CERT or _KEY file is provided, the printer will use a generic default client certificate when attempting to connect to a broker.

More information on how to create a CSR can be found in the provisioning methods command topic [provisioning-place_cert on page 28](#).

E:MQTTx_CA.NRD

The certificate authority (CA) the printer will use to verify trust in (for TLS) the MQTT server to which it is connecting. It can contain a single Root CA, optionally followed by multiple intermediate or subordinate CAs as well, all concatenated together in this one file.

E:MQTTx_CERT.NRD

This optional file allows for the configuration of a printer device certificate to present to the MQTT server for purposes of mutual authentication. If the desire is for the printer to send any intermediate or subordinate CAs used in signing this device certificate, they must come after the device certificate. Put another way, the device certificate must come first in a list of concatenated PEM contents in this file.

E:MQTTx_KEY.NRD

This optional file contains the corresponding private key of the device certificate used for mutual authentication with the MQTT server.

Example

A typical certificate file, including the multi-part form download wrapper, looks like the following. The certificate itself is in between the BEGIN CERTIFICATE/END CERTIFICATE markers and includes the markers. The material above and below the markers is the multi-part form wrapper.

```
{ }--P_q0t8QvwCIp-9Ny1G4F6xn9TpbIxj2i2E05D
Content-Disposition: form-data; filename="E:MQTT1_CA.NRD"; action="store"
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

-----BEGIN CERTIFICATE-----
MIIDBTCAe2gAwIBAgIUcIUQirABv6uIetEWqQN2DaUNY2kwDQYJKoZIhvcNAQEL
BQAwEjEQMA4GA1UEAwHU09USS1jYTAeFw0yMjAyMDExODQ1NDJaFw0yMjAyMDEx
ODQ1NDJaMBIxEDA0BgNVBAMMB1NPVEktY2EwggEiMA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKA... .HuFH/ouP4JzjgccBfZrQsw3cN1F+rxJ6EPGMOs1h8dNTCUMekdc8Z7ycXvEwClYq
xWa5NqGF1TFIbp/Tk/1of1o7a1kskmaYvteplGctCExDZMqAoT83K1QRPSsUIqAX
qj42uUGFdKPk
-----END CERTIFICATE-----
--P_q0t8QvwCIp-9Ny1G4F6xn9TpbIxj2i2E05D--
```

Topics

This section describes the generic layout of topics the printer will automatically publish and subscribe to on connection to an MQTT server.

Subscribe



NOTE: The maximum size of a single command request for incoming data must be less than 32k bytes.

Command

Description	This topic is what the printer uses to receive all commands on for processing. If multiple parties are interested in sending a command at the same time, it is recommended to use a unique <code>id</code> in the payload to ensure any response that is published is for the appropriate command.
Topic name	<code><mqtt-connX-tenant_id>/<device-zuid>/command</code>
QoS	<code><mqtt-connX-qos></code>
Payload	
<code>id</code>	Unique string or number for the command request
<code>method</code>	The name of the command to process, see Command Methods on page 24 .
<code>params</code>	Method specific parameters to use for command processing
Example	<pre>{"method":"settings","id":55,"params":{"appl-name":null}}</pre>

Publish



NOTE: The maximum size of a single publish response for outgoing data must be no more than 500,000 bytes.

Connected

Description	This topic is what the printer uses to show whether it is currently connected to an MQTT server.
Topic name	<mqtt-connX-tenant_id>/<device-zuid>/connected
QoS	<mqtt-connX-qos>
Retain flag	true
Dup flag	false
Payload	
connected	Displays true when connected and false when not connected via LWT.
mqtt_api_version	The MQTT API version number that the printer is using to specify the MQTT functionality and capability provided. This number may change if MQTT functionality is modified.
appl-name	The name of the firmware version running on the printer
appl-link_os_version	The Link-OS version of the firmware. This number will increase as new features are added and old features are deprecated. For a complete list of features, please refer to firmware release notes.
device-product_name	The model name of the printer
mqtt-connX-tenant_id	The tenant ID used for the connection number specified
Example	<pre>{ "method": "connected", "params": { "connected": true, "mqtt_api_version": 1, "appl-name": "V92.20.1Z", "appl-link_os_version": "6.4", "device-product_name": "ZT411", "mqtt-conn1-tenant_id": "zebra" } }</pre>

Last Will and Testament (LWT)

The printer will automatically setup its LWT to publish to the connected topic to clear the **connected** portion of the payload so that it shows the printer is no longer connected to the MQTT server (with it being set to false).

Printer Sleep Behavior

Some printers support the ability to sleep for power saving. If this occurs the printer will disconnect from the broker and attempt reconnection on wake. There is no way to wake the printer up from sleep via MQTT for reasons of maximizing battery life.

If the desire is for the printer to sleep with MQTT and a WiFi radio it is recommended to turn off the `power.wake.radio` setting. Other sleep settings include `power.sleep.enable`, `power.sleep.timeout`, `power.sleep.cradle`, and `power.sleep.unassociated`. Typically, the printer will be charging or in a cradle when remote management occurs.

Response

Description	This topic is what the printer uses to publish all responses from the command request. As this topic is directly related to the result of a subscribed command, the response payload will match the <code>id</code> from the original command request.
Topic name	<code><mqtt-connX-tenant_id>/<device-zuid>/response</code>
QoS	<code><mqtt-connX-qos></code>
Retain flag	<code>false</code>
Dup flag	<code>false</code>
Payload	
<code>id</code>	Unique string or number matching the original command request
<code>result</code>	Method specific results from the processed command
Example success	<code>{"id":55,"result":null}</code>
Example failure	<code>{"id":55,"error":{"code":10,"message":"generate_csr error"}}</code>

Response Status

In hopes to maintain a consistent interface between methods, this section will describe various guidelines to consider when working with methods regarding return values. These return values will be automatically published in the `response` topic for whatever command topic method was received

Where possible the payload of the MQTT topics strive to be JSON RPC compliant:

jsonrpc.org/specification

Success

Unless the method is specifically returning data, the method should return null as the result. It is assumed that most methods will return this.

```
{"id":12,"result":null}
```

Success with data

If the method was successful and returns data it can be returned as an int, Boolean, quoted string, array, or object depending on what is needed.

```

{"id":13,"result":1243}
{"id":13,"result":false}
{"id":13,"result":"string"}
{"id":13,"result":{"...}}
{"id":13,"result":[...,...,...]}
    
```

Failure

If the method encounters an error, the Response Object **MUST** contain the error member with a value that is an Object with the following members:

Code

An integer that indicates the error type that occurred.

Message

A string providing a short description of the error.

The message **SHOULD** be limited to a concise single sentence.

The negative error codes from and including -32768 to -32000 are reserved for pre-defined errors. Any code within this range, but not defined explicitly below is reserved for future use. The error codes are nearly the same as those suggested for XML-RPC at the following url:

xmlrpc-epi.sourceforge.net/specs/rfc.fault_codes.php

Code	Message	Meaning
-32700	"Parse error"	Invalid JSON was received by the server. An error occurred on the server while parsing the JSON text.
-32600	"Invalid Request"	The JSON sent is not a valid Request object
-32601	"Method not found"	The method does not exist / is not available
-32602	"Invalid params"	Invalid method parameter(s)
-32603	"Internal error"	Internal JSON error
-32000 to -32099	"Server error"	Reserved for implementation-defined server-errors
-32001	"Method not supported"	The method called is not supported over this communication channel.

For example:

```

{"id":14,"error":{"code":-32602,"message":"invalid params"}}
    
```

If the method needs to return something different than a pre-defined error, it will return a method specific positive error code. This suggests that two different methods could return the same code but mean something completely different and use a different message.

For example:

```

{"id":15,"error":{"code":402,"message":"ATM open"}}
{"id":16,"error":{"code":402,"message":"destination unreachable"}}
    
```

Notification

Description	This topic is what the printer uses to publish all notifications that occur. In general notifications occur asynchronously from any commands subscribed to. Because of this, they will not include an id parameter in the JSON payload.
Topic name	<mqtt-connX-tenant_id>/<device-zuid>/notification
QoS	<mqtt-connX-qos>
Retain flag	false
Dup flag	false
Payload	
method	The name of the notification that occurred, see notification method table elsewhere in the document
params	Method specific information from the notification
Example	<pre>{ "method": "file-download_complete", "params": { "digest": "96c5f3a78dcdd3f35e49b47996039ed31d59acf192e66554cda6afe65074fa" } }</pre>

Methods

Command Methods

This is a description of command methods that are used to implement the available functionality that can be used with MQTT.

The general command's format structure uses a JSON-RPC like communication as can be seen in the examples for each commands in this section.

The example response for each method is listed with the method documentation for enhanced readability. In actuality, the response will not be published via the command topic, and instead will be published in the response topic. Below is a summary of all the supported methods for the subscribed command topic.

Description	Method	SGD Name	Value
Update certs	file-download provisioning-generate_csr provisioning-place_cert	N/A	N/A
Update firmware	file-download file-apply_firmware	N/A	N/A
Configure settings	settings	See ZPL Programming Guide.	See ZPL Programming Guide.
Print test page	settings	device-print_2key	<anything>
Log file access	settings	device-syslog-entries	<anything>
Installed certs info	settings	file-cert_expiration	<anything>
Reset printer	settings	device-reset	<anything>
Factory reset	settings	device-restore_defaults	"all"
Factory reset MQTT	settings	mqtt-restore_defaults	<anything>
Delete files	settings	file-delete	See ZPL Programming Guide.
Alerts configuration	settings	alerts-add or alerts-configured	See ZPL Programming Guide using the MQTT alerts destination.

Provisioning

Provisioning Error Table

Error Name	Error Code
No error	0
System error	1
Bad service name string, pick one of the supported services	10
A problem occurred with the CRC when placing a cert over MPF	11
A problem occurred with the file size when placing a cert over MPF	12
Too many CSR requests are being processed	50
Bad JSON formatting	51
Bad common name (CN)	53
Bad key size for the algorithm, pick a supported size	54
Bad curve name, pick a supported name	55
Bad key algorithm type, pick a supported type	56
Bad location	57
Bad state	58
Bad country	59
Bad organization	60
Bad organization unit	61
Bad email	62
Bad subject alternate name	63
Bad digest type, pick a supported digest type	64
The challenge password specified was too large	65
The challenge specified was invalid	67
The customer organization unit was too large	68
The hardware common name was invalid	69
Certificate being placed is formatted incorrectly	150
Certificate being placed using a weak cipher	151
Certificate being placed does not match private key generated	152
Certificate being placed is not valid for time on printer	153

Provisioning-generate_csr

Method	provisioning-generate_csr(service, message_digest, cn, challenge, o, ou, email, l, st, c, san, key)	
Description	Instruct the printer to generate a CSR for a given network service. A response will return based on command success or failure. At some point later, a csr_ready notification will return if the command was successful and a CSR is available for signing.	
Parameters	Type	Notes
service	string	Can be MQTT1, MQTT2, WEBLINK1, WEBLINK2, HTTPS, TLSRAW, WIRED, WLAN
message_digest	string	Typically "sha256"
CN	string	Common name
names	array	Array containing an Object that contains all the optional names to be applied to the CSR request
names.challengePassword (optional)	string	A challengePassword token to add to the CSR. Can be used as a one-time token with a Certificate Authority to allow signing.
names.O (optional)	string	Organization
names.OU (optional)	string	Organization unit
names.emailAddress (optional)	string	An email address
names.L (optional)	string	Location
names.ST (optional)	string	State
names.C (optional)	string	Country
names.subjectAltName (optional)	string	Subject alternate name. The printer will always append an extra SAN of the <device-zuid>.
key	object	Object must have algo and either size or curve depending on which algorithm used
key.algo	string	"ecdsa" Or "rsa"
key.size	unsigned integer	Required for rsa, must be at least 2048 for that algo.
key.curve	string	Required for ecdsa, see the curves setting for full list of supported curves.

Methods

Example request	<pre>{ "method": "provisioning-generate_csr", "params": { "service": "MQTT1", "message_digest": "sha256", "CN": "MQTT Printer", "names": [{ "C": "US", "L": "Lincolnshire", "O": "Zebra Technologies", "ST": "Illinois", "challengePassword": "12345678911234567892123456789312" }], "key": { "algo": "ecdsa", "size": 256, "curve": "prime256v1" } }, "id": 61 }</pre>	
Response	Indicates success or failure of the printer CSR generation request.	
Field	Type	Notes
result	integer	Return status, see Provisioning Error Table on page 25 for list of codes.
Example response success	<pre>{ "id": 61, "result": null }</pre>	

provisioning-place_cert

Method	provisioning-place_cert(service, cert)	
Description	Used to place the signed CSR onto the printer for use in a network service. It will be successful if the current private key matches the public key in the installing certificate.	
Parameters	Type	Notes
service	string	Can be MQTT1, MQTT2, WEBLINK1, WEBLINK2, HTTPS, TLSRAW, WIRED, WLAN.
cert	string (PEM)	Signed CSR certificate
Example request	<pre>{ "method": "provisioning-place_cert", "id": 62, "params": { "service": "MQTT1", "cert": "-----BEGIN CERTIFICATE-----\r\n MIICNzCCAd6gAwIBAgIBAJAKBggqhkJOPQQDAjB4MQswCQYDVQQGEwJVUzEVMBMG\r\n nhFRgDwX6TAFBgNVHSMEGDAwBgQ/6yZbW0ypnHqHF+8s6w0zAk1fuzAMBgNVHRME\r\n BTADAQH/MAoGCCqGSM49BAMCA0cAMEQCIEOfDez2EhJw9AQFkwrUSqADOPvK/qBO\r\n Fli14pa51y4yAiBLoeQcOcc0ow8v4R86R9ZD9QpumJKyKnTVqmyWEWicw==\r\n -----END CERTIFICATE-----"} }</pre>	
Response	Indicates success or failure of the place cert request.	
Field	Type	Notes
result	integer	Return status, see Provisioning Error Table on page 25 for list of codes.
Example response success	{"id":62,"result":null}	
Example response failure	<pre>{ "id": 62, "error": { "code": "...", "message": "..." } }</pre>	

File

File Method Error Table

Error Name	Error Code
No error	0
System error	1
Bad firmware download type, pick one of the supported types	50
Bad web address to download from	51
Bad file download digest specified	52
Bad file size specified	53
Bad HTTP header specified	54
Bad certificate authority (CA) specified	55
Bad filename specified	56
Bad proxy specified	57
Bad authorization specified	58
The download of the file ran out of time (typically 2 minutes)	70
The digest specified does not match the calculated digest of the file received	71
The size specified does not match the calculated size of the file received	72
The file already exists	73
The download received an HTTP error	74
The download received a connection error	75
A file is already being downloaded, only one file can be downloaded at a time	76
Not enough space for file	77
No file is present to apply firmware for	150
The firmware to apply does not match the firmware file downloaded	151
Bad apply firmware digest specified	152
The ability to upgrade firmware has been disabled	153
The firmware version to apply is not correct for the printer	154
A generic firmware download failure	155
A request to apply firmware is already in progress. You cannot apply more than one firmware upgrade at a time.	156
Downloading this older firmware is not allowed.	157

File-download

Method	file-download(type, filename, web_address, digest, size, headers, ca_file, proxy, authorization)	
Description	Starts the process of downloading a file onto the printer to use as firmware, or store on the file system from an HTTPs URL GET	
Parameters	Type	Notes
type	integer	Printer Firmware 1 Store 2
filename (required for file store)	string	If storing a file, this will be the drive and file name destination. For example: E:MQTT1_CA.NRD
web_address	string	HTTPs address with required tokens to allow download of a file. Minimum length 9 bytes, Maximum length 2048 bytes.
digest	string	SHA-256 digest of the file. Used to ensure we get the file correctly. Length always 64 bytes. Hash must be lowercase as it is case sensitive.
size	integer	Size of the file to download, in bytes.
headers (optional)	string	Additional HTTP headers that should be used during the request. Each header must contain a (:) or (;) and be terminated with a \r\n. Maximum length 2048 bytes. Format: "<header>:<value>\r\n". To have no value use "<header>;\r\n", note (;) use. To suppress a header use "<header>:\r\n", note (:) immediately followed by \r\n..
ca_file (optional)	string	PEM Certificates to be used as the CA file. This is the complete CA chain. Maximum length 32 KB less the remaining size of the publish request not including the actual ca. If no file is specified, the MQTTx_CA.NRD file shall be used
Example request	<pre>{ "method": "file-download", "id": 71, "params": { "type": 1, "web_address": "https://dev.zebra.com/Q?X-Zsb-Algorithm=AWS4-HMAC-SHA256&X-Zsb-Credential=zebrasoft%2Fus-east-1%2Fs3%2Faws4_request&X-Zsb-Date=20190522T165855Z&X-Zsb-Expires=604800&X-Zsb-SignedHeaders=host&X-Amz-Signature=2fd9cc22883d9cb", "digest": "27defb7548c8c700bf10995e59debc35f31a51b67f74fa0e367ba337484080d3", "size": 2112000, "ca_file": "-----BEGIN CERTIFICATE-----\r\nMIIBrjCCAvoGAWIBAgIJAKUCiNNpCSRnMAoGCCqGSM49BAMCFIxCzAJBgNVBAYT\r\n-----END CERTIFICATE-----\r\n"} }</pre>	
Response	Indicates success or failure of queuing the download request. Once complete, a file-download_complete shall be published on the notification topic.	

Methods

Field	Type	Notes
error	integer	See File Method Error Table on page 29 .
Example response success	<pre>{"id":71,"result":null}</pre>	
Example response failure	<pre>{"id":71,"error":{ "code": "...", "message": "..." }}</pre>	

File-apply_firmware

Method	file-apply_firmware(digest)	
Description	Start the printer firmware upgrade process with the firmware that has already been downloaded. This required that the file-download command was already sent, and the firmware has completed the download of the file onto the printer.	
Parameters	Type	Notes
digest	string	SHA-256 hash of the firmware file. If the hashes do not match, the firmware will not be installed. Hash must be lowercase as it is case sensitive.
Example request	<pre>{"method":"file-apply_firmware", "id":72,"params":{"digest":"27defb7548c8c700bf10995e59debc35f31a51b67f74fa0e367ba337484080d3"}}</pre>	
Response	Indicates success or failure of queuing the upgrade request. If there is an error during the firmware installation process you will get a file-apply_firmware_failure notification from the printer. The printer may or may not reset depending upon the severity of the issue encountered.	
Field	Type	Notes
error	integer	See File Method Error Table on page 29 .
Example response success	<pre>{"id":72,"result":null}</pre>	
Example response failure	<pre>{"id":72,"error":{"code":..., "message":"..."}}</pre>	

Settings

Method	settings(setting,...)	
Description	Retrieve or set the value of one or more named settings on the printer. Setting names must not be duplicated in a single request. If doing a set, the values are checked for validity before applying. If any of the values are invalid, none of the values will be applied.	
Parameters	Type	Notes
setting	object	An array of setting name value pairs to set or get from the printer. Valid setting names can be found from the SGD section of the ZPL Programming Guide.
setting.name	string	The name of the setting to set or get
setting.value	string	The value to set, or null if doing a get. If doing a set, all setting values must be set as strings, no other native types are supported.
Example request	<pre>{ "method": "settings", "id": 85, "params": { "mqtt-conn2-server_address": "mqtts://zebra.com/mqtt", "appl-name": null, "rtc": null, "bad_setting_name": null } }</pre>	
Response	If an unsupported setting name is requested, the invalid setting name shall be returned with a null. If invalid values are specified for a setting, an invalid params error shall be returned	
Field	Type	Notes
setting	object	An object containing the requested key value pairs. Each member of the object will have its field key equal to the requested setting name, and the field value is the current value.
setting.name	string	The setting name
setting.value	string or object	The current setting value or null if the setting does not exist. If doing a get, an individual setting or group may be requested. If an individual value is requested, it is represented in the response as a string. If a group is requested, it is represented as an object consisting of individual values within the group. If a group contains one or more child groups, they are flattened and only the individual values within each child group are returned.

Example response	<pre>{ "id": 85, "result": { "mqtt-conn2-server_address": "mqtts://zebra.com/mqtt", "appl-name": "V91.20.07Z", "rtc": { "rtc-time": "07:31:02", "rtc-date": "05-05-2021", "rtc-timezone": "UTC00", "rtc-exists": "yes", "rtc-unix_timestamp": "1620199862" }, "bad_setting_name": null } }</pre>
Example response invalid setting	<pre>{"error": {"code": -32602, "message": "Invalid params"}, "id": 85}</pre>

Notification Methods

Below is a summary of all the supported methods for the published notification topic.

Notification Method Name
alert
file-download_complete
file-apply_firmware_failure
csr_ready
error

Alert

Method	alert(condition_id, condition_state, type_id, time_stamp, setting_name, setting_value, filename, condition_code)	
Description	<p>The printer will send asynchronous alerts whenever an important printer state change occurs. This is used for things that are serviceable by the user.</p> <p>Alert Notifications are sent for changes in state while the connection is active. They are not queued when the printer is offline. Regardless of what language is specified on the printer, no translation of alert information will be done for MQTT.</p>	
Parameters	Type	Notes
condition_id	string	Name of the alert. See the ZPL Programming Guide for a complete list of alerts the printer supports and how to configure the printer for sending them. By default ,the printer will not send any alerts.
condition_state	string	SET or CLEAR
type_id	string	ERROR, ALERT, or WARNING
time_stamp	string	The time on the printer when the alert occurred in the following format: YYYY-MM-DD HH:MM:SS
setting_info (SGD alert only)	object	The key value pair of the setting that was changed via SGD,
filename (CSR alert only)	string	The name of the certificate file name that is the source of the alert,
condition_code (CSR alert only)	unsigned integer	The numeric code from the result of the CSR operation, see Provisioning Error Table on page 25 for list of codes.
Example notification	<pre>{ "method": "alert", "params": { "condition_id": "SGD SET", "condition_state": "SET", "type_id": "ALERT", "time_stamp": "2021-01-03 19:27:56", "setting_info": {"mqtt-enable": "on"} } }</pre>	

Alert List

Name	Description
PAPER OUT	No labels are present
RIBBON OUT	No ribbon is present
HEAD TOO HOT	The printhead is too hot to function
HEAD COLD	The printhead is too cold to function
HEAD OPEN	The printhead or latch is open
SUPPLY TOO HOT	The power supply is too hot
RIBBON IN	Ribbon was detected when it shouldn't be used
REWIND	There was an error when attempting to rewind label liner
CUTTER JAMMED	The cutter is jammed
PRINTER PAUSED	The printer is paused
PQ JOB COMPLETED	The print quantity job has finished
LABEL READY	A label is ready and presented
HEAD ELEMENT BAD	There is a bad element on the printhead
BASIC RUNTIME	There is a ZBI runtime error
BASIC FORCED	There is a ZBI forced error
POWER ON*	The printer has turned on
CLEAN PRINthead	The printhead needs to be cleaned
MEDIA LOW	There is a small number of labels left
RIBBON LOW	There is a small amount of ribbon left
REPLACE HEAD	The printhead needs to be replaced
BATTERY LOW	The printer battery is low
RFID ERROR	There is an RFID error
COLD START*	The printer has received an IP address
SGD SET	An SGD setting set has occurred
CSR AVAILABLE	A CSR is ready to be signed and placed on the printer as a certificate
MOTOR OVERTEMP	The stepper motor is too hot to function
PRINthead SHUTDOWN	The printhead has shutdown
SHUTTING DOWN*	The printer is being powered off
RESTARTING*	The printer is being reset
NO READER PRESENT	No RFID reader is present when one was expected
THERMISTOR FAULT	The printhead has a bad thermistor
INVALID HEAD	The printhead has failed authentication
COUNTRY CODE ERROR	The RFID reader does not have a country code set
MCR RESULT READY	The magnetic card reader result is ready
*These alerts are not sent as their functionalities are covered by the connected topic.	

Methods


Name	Description
PMCU_DOWNLOAD	The PMCU download has failed
RIBBON_AUTH_ERROR	The ribbon cartridge has failed authentication
COUNTRY_CODE	The WLAN radio does not have a country code set
BATTERY_MISSING	The battery is not present
MEDIA_CARTRIDGE	The media cartridge has been unloaded
MEDIA_CARTRIDGE_LOAD_FAILURE	The media cartridge has an error on loading
MEDIA_CARTRIDGE_EJECT_FAILURE	The media cartridge has an eject failure
MEDIA_CARTRIDGE_FORCED_EJECT	The media cartridge was forced to eject
CLEANING_MODE	The media cartridge is cleaning
ODOMETER_TRIGGERED	An alert based on an odometer has occurred
ALL_MESSAGES	All the above messages
*These alerts are not sent as their functionalities are covered by the connected topic.	

File


File-download_complete

Notification	file-download_complete(digest, status_code, syslog_codes[])	
Description	This notification is sent by the printer when a download has completed or failed.	
Field	Type	Notes
digest	string	Is the file digest given in the file-download request.
status_code (present if there is an error or abnormal status)	unsigned integer	See File Method Error Table on page 29 .
syslog_codes (present if there is an error or abnormal status)	array[unsigned integer]	Syslog codes sent back in decimal form to track all the issues that occurred as part of an error or abnormal status.
Example notification success	<pre>{ "method": "file-download_complete", "params": { "digest": "96c5f3a78dcdd3f35e49b47996039ed31d59acf192e66554cda6c8afe65074fa" } }</pre>	
Example notification error	<pre>{ "method": "file-download_complete", "params": { "digest": "96c5f3a78dcdd3f35e49b47996039ed31d59acf192e66554cda6c8afe65074fa", "status_code": 6, "syslog_codes": [1, 40101, 221] } }</pre>	

File-apply_firmware_failure

Notification	file-apply_firmware_failure(status_code, downloader_log, installer_log)	
Description	<p>This notification is sent by the printer when a file-apply_firmware method for the command topic has failed. The printer may or may not reset after sending this notification. It depends upon the severity of the failure and the printer type.</p> <p> NOTE: Not all printer platforms are capable of returning this information.</p>	
Field	Type	Notes
status_code	unsigned integer	See File Method Error Table on page 29 .
digest	string	Is the file digest given in the file-apply_firmware request.
downloader_log	string	Downloader log file Some printers are not capable of returning any log information; in which case, this field shall be JSON null.
installer_log	string	Installer log file (may be from a previous run) Some printers are not capable of returning any log information; in which case, this field shall be JSON null.
Example notification error	<pre>{ "method": "file-apply_firmware_failure", "params": { "status_code": 155, "digest": "27defb7548c8c700bf10995e59debc35f31a51b67f74fa0e367ba337484080d3", "downloader_log": "08-21-2020 15:25:10.436: Starting download of firmware\r\n08-21-2020 15:25:10.442: Looking for magic numbers\r\n ...\r\n08-21-2020 15:25:10.508: Manifest verified and parsed:\r\n08-21-2020 15:25:10.510: Application name: V95.1.0.5\r\n08-21-2020 15:25:10.512: Link-OS version: 6.3\r\n08-21-2020 15:25:10.513: Build ID: 53628\r\n08-21-2020 15:25:11.516: Platform requires 4 sections\r\n08-21-2020 15:25:11.518: Installer required; ID=500\r\n08-21-2020 15:25:11.520: Begin section ID=1000; size=135776 bytes\r\n08-21-2020 15:25:11.522: Begin downloading installer.\r\n08-21-2020 15:25:11.653: Section ID=1000: parsing complete (size=135776) bytes\r\n08-21-2020 15:25:11.655: Processed installer (section 1000)\r\n08-21-2020 15:25:11.657: Begin section ID=502; size=0 bytes\r\n08-21-2020 15:25:11.659: Installer checkpoint reached.\r\n08-21-2020 15:25:11.661: Invoking the installer...\r\n08-21-2020 15:25:11.668: Installer spawned successfully.\r\n", "installer_log": "08-21-2020 15:25:11.723: Manifest metadata:\r\n08-21-2020 15:25:11.729: Application name: V95.1.0.5\r\n...\r\n08-21-2020 15:25:28.819: Firmware install process done. Rebooting...\r\n08-21-2020 15:25:28.821: Rebooting the system...\r\n" } }</pre>	

CSR_ready

Notification	csr_ready(service, csr, status_code)	
Description	<p>This notification is sent by the printer when a CSR file is ready for signing. This can be at the connection of MQTT or once the CSR file is generated.</p> <p> Note: The challenge password of the CSR will be returned as is specified by the generate_csr request and embedded within the CSR itself.</p>	
Field	Type	Notes
service	string	Can be MQTT1, MQTT2, WEBLINK1, WEBLINK2, HTTPS, TLSRAW, WIRED, WLAN.
csr (on success)	string (PEM)	CSR generated by the printer for signing.
status_code (on error)	unsigned integer	Defined error codes for CSR generation process.
Example notification success	<pre>{ "method": "csr_ready", "params": { "service": "MQTT1", "csr": "-----BEGIN CERTIFICATE REQUEST-----\r\n MIIBaDCCAQ0CAQAwgaoxJTAjBgNVBAMHDk5SjE5MDIwMDE0NTUyMjA3QzZDQTQw\r\n MTAwMDAxFTATBgNVBACMDExpbnNvbG5zaGlyZTERMA8GA1UECAwISWxsaw5vaXMx\r\n dAI8egVrQCDlk8DttBFDFew9jVufg3Ymx7CY7evYwDH3TnGgADAKBggqhkJOPQQD\r\n AgNJADBGAiEAg1qSBIYgWzwGKW1XxB7UApq07g9lGKC6vSt+3joNB+MCIQCYMlaa\r\n gBXdi+KwU0sXTtdzfikjSnKxLDBzvbCOCg3GhA==\r\n -----END CERTIFICATE REQUEST-----", }} </pre>	
Example notification error	<pre>{ "method": "csr_ready", "params": { "service": "MQTT1", "status_code": 67 }} </pre>	

Error

Notification	error(code, message)	
Description	This notification is sent by the printer when it receives data over MQTT that causes an error such that it cannot be tied to any particular topic, method, or id. In general, these are sent based on actions under user control.	
Field	Type	Notes
code	integer	See Error Notification Error Table below.
message	string	See Error Notification Error Table below.
Example notification	<pre>{ "method": "error", "params": { "code": -32700, "message": "Parse error" }}</pre>	

Error Notification Error Table

code	message	Description
-32600	"Invalid Request"	The payload was valid JSON, but not formatted according to JSON RPC Spec requirements. Make sure the payload has a valid "id" specified and is contained in a JSON object as shown in examples.
-32603	"Internal error"	There was an internal Zebra error that occurred, please contact Zebra Tech Support.
-32700	"Parse error"	Invalid JSON was sent in the payload. Please consider using a JSON formatter to validate the payload and try again.
-32000	"Invalid Request: exceeded max length"	The command topic received too much data, please reduce the size of data being sent in a single request to less than 32k bytes.

Logging

Printer (Syslog)

Normal syslog messages from the printer are returned in this format (when getting device-syslog-entries or inspecting the E:SYSLOG.TXT file):

```
Jun 12 12:52:06 localhost : [Power][Informational][0X14] Power On
Jun 12 12:52:06 localhost : [Weblink][Informational][0X1002] Starting Weblink for Conn 1
Jun 12 12:52:06 localhost : [conn1.1][Informational][0X1006] Waiting for an IP
Jun 12 12:52:06 localhost : [CSR][Informational][0X80800000] CSR WEBLINK1 is ready
Jun 12 12:52:07 localhost : [Bluetooth LE][Informational][0X322] Advertising enabled
Jun 12 12:52:07 localhost : [Bluetooth Classic][Informational][0X304] Discoverability off
```

Once in JSON format, it should look something like this:

```
{"method":"settings","id":649,"params":{"device-syslog-entries":null}}
}}
```

And this:

```
{"id":649,"result":{"device-syslog-entries":"Apr 28 14:21:00 localhost :
[conn99.3][Informational][0X8002000C] SSLv3 TLS\nApr 28 14:21:00 localhost :
[conn42.0][Error][0X8002000C]Sic mundus creatus est\n"}}
```

Refer to the [ZPL Programming Guide](#) for more information on how to configure the printer for Syslog.

MQTT Specific

To view logs only pertaining to MQTT printer connections, please make use of the [mqtt-logging-entries](#) on [page 14](#).

Connecting to SOTI Connect

Before You Get Started

Connecting Zebra Printers with the latest Link-OS firmware to SOTI Connect can be achieved following these few simple steps. Before you get started with configuring the printer, however, you will need the following information:

1. MQTT Broker Address that is connected to SOTI Connect.
2. MQTT Broker authentication information. This could be username/password or a certificate to use for the client.
3. A network connection for your printer (wired or wireless)
4. A Link-OS printer with the latest firmware
5. How to send configuration commands and files to the printer based on information from the ZPL Programming Guide and Printer Administration Guide

The information in this section is reliable at the time it is created.

Printer Configuration

This assumes:

- You are using the first connection available (conn1) for MQTT configuration.
- The default tenant for the topic path of
- The default topic path (Zebra) for the tenant is acceptable.

Instructions

1. Send the following configuration:

```
{ }{"mqtt-enable":"on",  
  "mqtt-conn1-server_address":"mqtts://your-mqtt-broker-name-here.com:8883"}
```

2. Load the Certificate Authority (CA) for the TLS broker so the printer can trust the connection. This can be done by securely loading the certificate file as MQTT1_CA.NRD onto the printer.

3. Setup the printer to authenticate with the broker. This can be done with username and password configuration, or by utilizing certificates.

```
{ } { "mqtt-conn1-username": "your-username-here",  
      "mqtt-conn1-password": "your-password-here" }
```

Alternatively, printer certificates can be used. This can either be done using CSR's to guarantee each printer has a unique certificate where the private key never leaves the printer, or with a pre-signed certificate and private key pairs. The service (and file name) to use for these certificates is "MQTT1". More details about certificate loading can be found in the Printer Admin Guide.

4. Reset the MQTT connection (without having to reset the printer). This can be done by sending:

```
{ } { "mqtt-conn1-reset_now": "yes" }
```
5. Verify the printer has connected to the broker and that it shows up in SOTI Connect with the matching client id `device-zuid`, or friendly name `device-friendly_name`.

Connecting to HiveMQ Cloud TLS 8883

Before You Get Started

Connecting Zebra Printers with the latest Link-OS firmware to a test instance of an MQTT broker like HiveMQ Cloud can be achieved following these few simple steps. If you want to get a feel for how MQTT works with Zebra printers, this is a great place to start, although it is certainly not recommended for production use.

Go to the HiveMQ website to find publicly available MQTT brokers to evaluate and test the HiveMQ prior to creating your own private broker.

Before you get started with configuring the printer, however, you will need the following information:

1. A network connection for your printer (wired or wireless)
2. A Link-OS printer with the latest firmware
3. How to send configuration commands and files to the printer based on information from the ZPL Programming Guide and Printer Administration Guide

The information in this section was reliable at the time it was created.

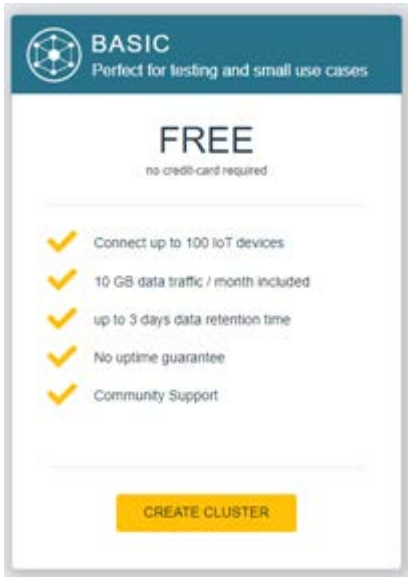
Setup the HiveMQ Cloud Cluster

This process was accurate as of June 2022 when published.

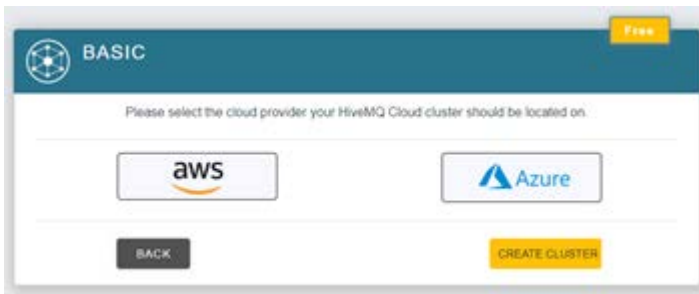
1. Create an account setup with HiveMQ Cloud at: console.hivemq.cloud/
2. Once logged in, click **Create New Cluster**.



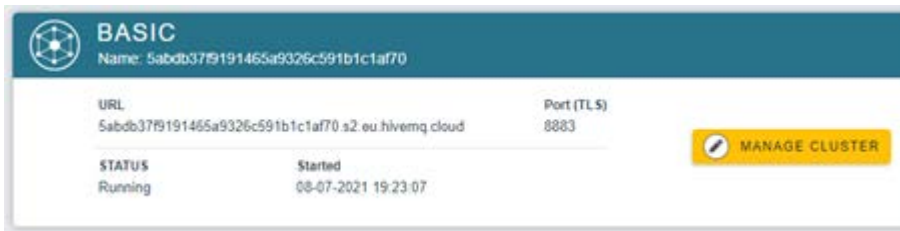
3. For now, select the free version for testing (non-production) purposes only and click **Create Cluster**.



4. Select either cloud provider you prefer as the printer will support AWS or Azure. and select **Create Cluster**. (For now, let's select AWS, but a second broker can always be created later with Azure.)



5. You should now see a cluster listed with a unique URL. Click **Manage Cluster**.



6. The URL and port number cannot change, but the username and password authentication can. Select **Access Management**, enter a username, and password twice, and then click **Add**.

Be sure to remember the password for use later on when configuring the client and printer.

Cluster Details [Back to clusters](#)

Overview **Access Management** Getting started

MQTT Credentials

Define the credentials used by your MQTT clients to connect to your HiveMQ Cloud cluster.
See [connect an MQTT client](#) for examples how to use the credentials to connect an MQTT client to your cluster.

Username: Password: Confirm password:

Active MQTT Credentials

These credentials give access to publish and subscribe to your HiveMQ Cloud cluster.

Username	Password	Actions
admin	*****	<input type="button" value="x"/>

Connect a Client to the Broker

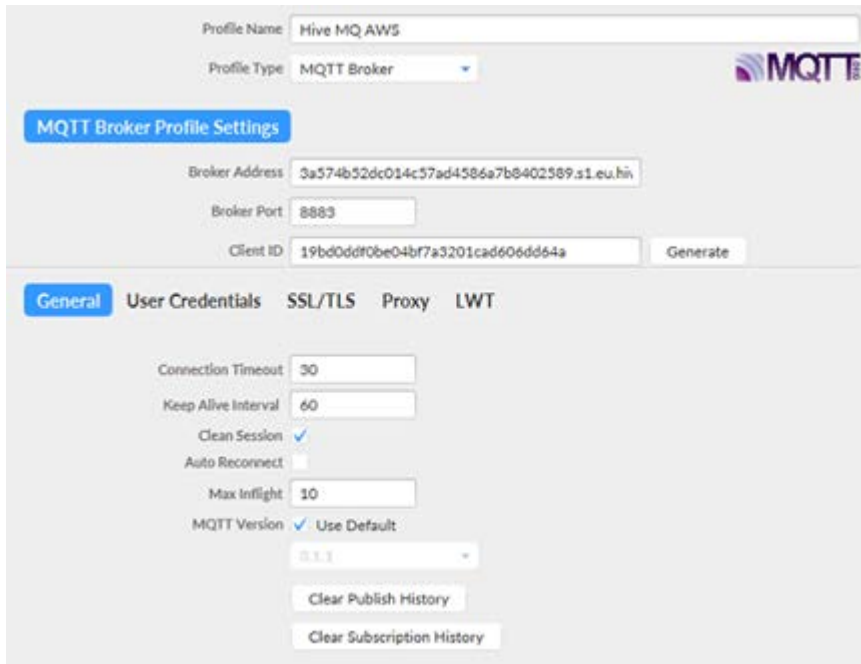
This assumes:

- You are using the second connection available (`conn1`) for MQTT configuration.
- The default port on the broker is 8883.
- The default topic path of Zebra for your tenant is acceptable.
- The network your printer is on can reach the Internet.
- You have set up a HiveMQ Cloud account, and have the URL and password ready.
- You have access to MQTT.fx version 1.7 or newer.

Instructions

1. In MQTT.fx, create a new connection profile for the HiveMQ broker.
2. Fill out the Profile Name to HiveMQ AWS.
3. Set the Broker Address to `<your-url-here>`.
4. Set the Broker Port to 8883.
5. Click **Generate** to create your Client ID.
6. Check the **Auto Reconnect** box in the **General** tab.

7. Click **Apply**.



The image shows the 'MQTT Broker Profile Settings' form in the HiveMQ console. The 'Profile Name' is 'Hive MQ AWS' and the 'Profile Type' is 'MQTT Broker'. The 'MQTT Broker Profile Settings' section includes: 'Broker Address' (3a574b52dc014c57ad4586a7b8402589.s1.eu.hivemq.cloud), 'Broker Port' (8883), and 'Client ID' (19bd0dd0be04bf7a3201cad606dd64a) with a 'Generate' button. Below this is the 'General' tab with settings for 'Connection Timeout' (30), 'Keep Alive Interval' (60), 'Clean Session' (checked), 'Auto Reconnect' (unchecked), 'Max Inflight' (10), and 'MQTT Version' (3.1.1, Use Default checked). There are also buttons for 'Clear Publish History' and 'Clear Subscription History'.

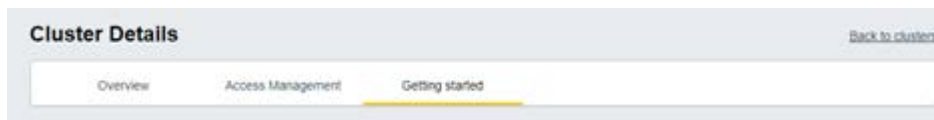
8. In the **SSL/TLS** tab, check the **Enable SSL/TLS** box.

9. From a browser, go to console.hivemq.cloud/ and click **Manage Cluster**.



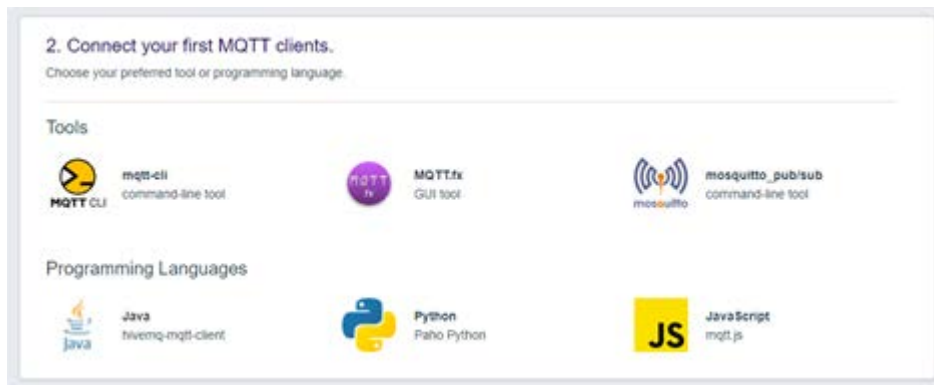
The image shows a 'BASIC' cluster overview card. It includes the cluster name '5abdb379191465a9326c591b1c1af70', the URL '5abdb379191465a9326c591b1c1af70.s2.eu.hivemq.cloud', and the Port (TLS) '8883'. The status is 'Running' and it started on '08-07-2021 19:23:07'. A yellow 'MANAGE CLUSTER' button is visible on the right.

10. Then, click **Getting Started**.



The image shows the 'Cluster Details' navigation bar with three tabs: 'Overview', 'Access Management', and 'Getting started'. The 'Getting started' tab is currently selected and highlighted in yellow. A 'Back to clusters' link is visible on the right.

11. Now, click **mosquitto_pub/sub**.



The image shows the 'Getting Started' page with the heading '2. Connect your first MQTT clients.' and the instruction 'Choose your preferred tool or programming language.' Below this are two sections: 'Tools' and 'Programming Languages'. The 'Tools' section includes: 'MQTT CLI' (mqtt-cli command-line tool), 'MQTT.fx' (MQTT.fx GUI tool), and 'mosquitto' (mosquitto_pub/sub command-line tool). The 'Programming Languages' section includes: 'Java' (hivemq-mqtt-client), 'Python' (Paho Python), and 'JavaScript' (mqtt.js).

12. Scroll down to the **FAQ** section, and click on the link to download the certificate authority from the cert link: letsencrypt.org/certs/trustid-x3-root.pem.
13. Back in MQTT.fx, select the **CA certificate file** radio button and browse to the recently downloaded pem file.
14. Click **Apply**.

The screenshot shows the 'MQTT Broker Profile Settings' window. The 'Profile Name' is 'Hive MQ AWS' and the 'Profile Type' is 'MQTT Broker'. Under 'MQTT Broker Profile Settings', the 'Broker Address' is '3a574b52dc014c57ad4586a7b8402589.s1.eu.hi', 'Broker Port' is '8883', and 'Client ID' is '19bd0ddf0be04bf7a3201cad606dd64a'. The 'SSL/TLS' tab is selected, with 'Enable SSL/TLS' checked and 'Protocol' set to 'TLSv1.2'. The 'CA certificate file' radio button is selected, and the file path 'C:\Users\Apekarske\Downloads\trustid-x3-root.pem' is entered in the 'CA Certificate File' field.

15. Cancel out of the box, select your new profile, and click **Connect**.



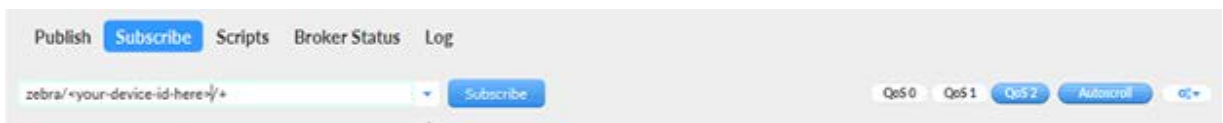
16. Once connected, you should see a green status and lock icon on the right.



17. Query your printer to find out what its client ID will be using for the device.zuid setting.
18. In the Subscribe tab, enter in the topic: **zebra/<device.zuid>/+**, and select the **QoS 2** option, then click **Subscribe**.



NOTE: Make sure there are no leading or trailing forward slash '/' characters in the topic.

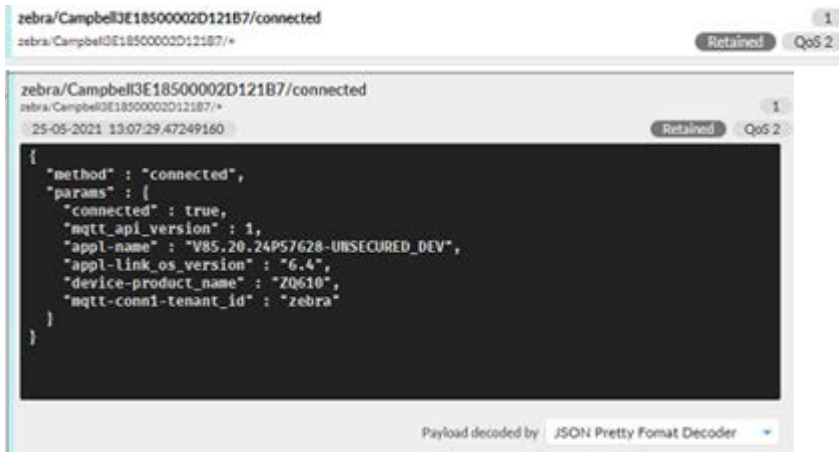


19. Once your printer is connected, you should see the connected topic showing a message from your printer.

Printer Configuration

1. Set the printer to connect to the HiveMQ Cloud TLS MQTT broker using the following settings:


```
{ }{"mqtt.enable": "on",
  "mqtt.conn1.server_address": "mqtt://<your-url-here>:8883",
  "mqtt.conn1.tenant_id": "zebra"}
```
2. Take the CA Certificate file used previously in connecting MQTT.fx and save it on the printer as MQTT1_CA.NRD to the E: FLASH drive.
3. Restart your printer and verify the connected message shows up as it connects to the broker, see the connected topic for more details. Once you see a message on the connected topic with a payload that shows "connected":true, you are connected!



This screenshot contains developer only information and should be retaken.

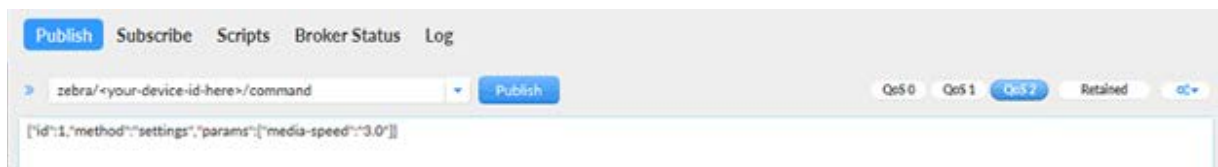
4. You can now configure the printer to send alerts to the broker using the following setting:


```
{ }{"alerts.add": "HEAD OPEN, MQTT, Y, Y, 1, , N, "}
```

Sending your first command via MQTT

Now that you are connected, you can send a simple command to the printer and get a response all over MQTT!

1. First, you will need to create a message for the command topic using the settings method. Let's start by setting the QoS to 2 and making sure Retain is not pushed in the **Publish** tab of the client.
2. Fill out the topic path using your tenant, and the device ID to which you previously subscribed.
3. Next, enter the payload for that topic. For this example, we will use the settings method to set and get the media speed. Click **Publish**.



4. In the **Subscribe** tab, you should see the command to set the media speed, on the command topic, and a response of the printer showing if it was able to set that value or not in the response topic.

The screenshot displays two MQTT messages in a sequence. The first message is a command sent to the topic `zebra/Campbell3E18500002D121B7/command` with a payload of `{ "id" : 1, "method" : "settings", "params" : { "media-speed" : "3.0" } }`. The second message is a response sent to the topic `zebra/Campbell3E18500002D121B7/response` with a payload of `{ "id" : 1, "result" : { "media-speed" : "3.0" } }`. Both messages are timestamped at 25-05-2021 13:27:45.48465024 and 25-05-2021 13:27:45.48465717 respectively.

5. Notice that if you attempt to set a setting with an invalid value (like not using a string), you will get an error.

The screenshot displays two MQTT messages in a sequence. The first message is a command sent to the topic `zebra/Campbell3E18500002D121B7/command` with a payload of `{ "id" : 1, "method" : "settings", "params" : { "media-speed" : 3.0 } }`. The second message is a response sent to the topic `zebra/Campbell3E18500002D121B7/response` with a payload of `{ "id" : 1, "error" : { "code" : -32602, "message" : "Invalid params" } }`. Both messages are timestamped at 25-05-2021 13:30:40.48640493 and 25-05-2021 13:30:41.48641191 respectively.

Connecting to Mosquitto

MQTT TLS 8883

The information in this section is reliable when it is created.

Before You Get Started

Connecting Zebra Printers with the latest Link-OS firmware to a public, test instance of an MQTT broker like Mosquitto can be achieved following these few simple steps. Port 8883 only requires that the printer contain the certificate authority information for the server to which it is connecting. If you want to get a feel for how MQTT works with Zebra printers, this is a great place to start, although it is certainly not recommended for production use. For purposes of demonstration, we have used MQTT.fx as a client to interact with our printer while connected to the broker.

Before you get started with configuring the printer, however, you will need the following information:

1. A network connection for your printer (wired or wireless)
2. A Link-OS printer with the latest firmware
3. How to send configuration commands and files to the printer based on information from the ZPL Programming Guide and Printer Administration Guide

The information in this section was reliable at the time it was created.

Connect a Client to the Broker

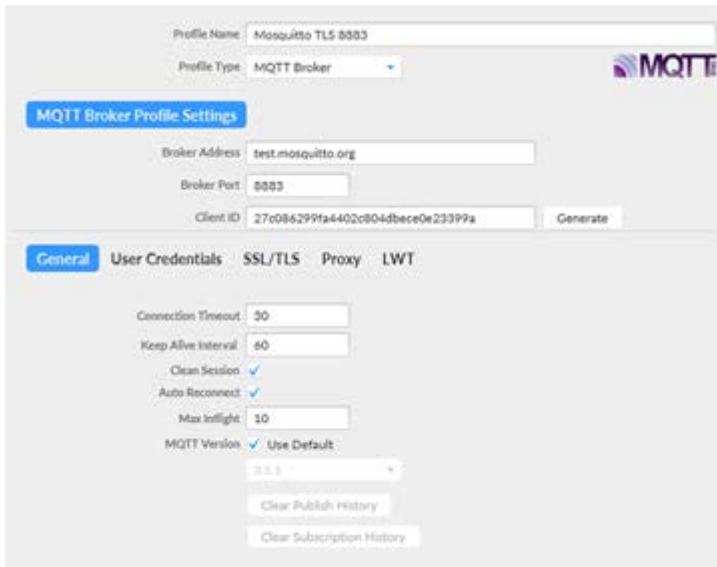
This assumes:

- You are using the first connection available (conn1) for MQTT configuration.
- The default port on the broker is 8883.
- The default topic path (Zebra) for the tenant is acceptable.
- The network your printer is on and can reach the internet.
- You have access to MQTT.fx version 1.7 or newer.

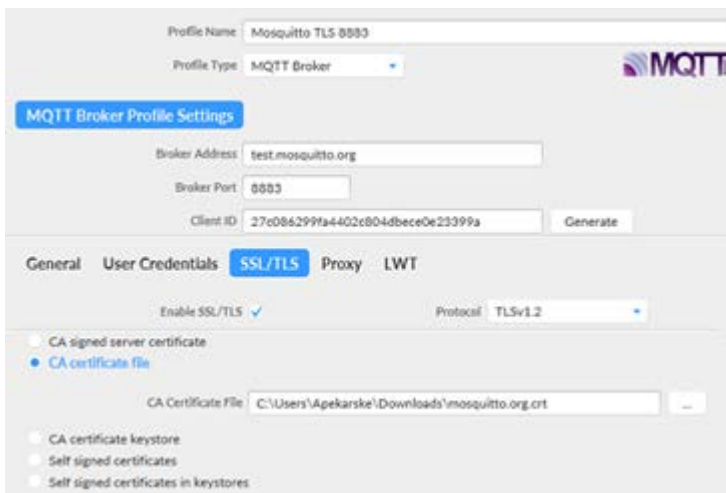
Instructions

1. In MQTT.fx, create a new connection profile for the test Mosquitto broker.
2. Fill out the **Profile Name** to Mosquitto TLS 8883.
3. Set the **Broker Address** to test.mosquitto.org.
4. Set the **Broker Port** to 8883.
5. Click **Generate** to create your "Client ID".

6. Check the **Auto Reconnect** box in the **General** tab.
7. Click **Apply**.



8. In the **SSL/TLS** tab, check the **Enable SSL/TLS** box.
9. From a browser, go to test.mosquitto.org/ and download the mosquitto.org.crt in PEM format to a location of your choice.
10. Back in MQTT.fx, select the **CA certificate file** radio button and browse to the recently downloaded crt file.
11. Click **Apply**.



12. Cancel out of the box, select your new profile, and click **Connect**.



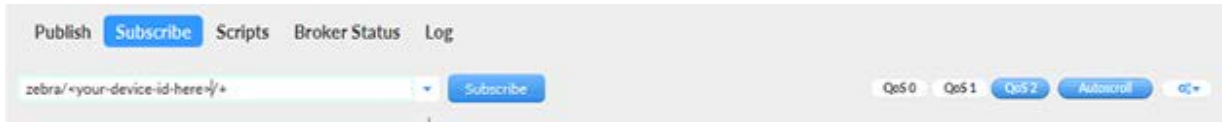
13. Once connected, you should see a green status and lock icon on the right.



14. Query your printer to find out what its client ID will be using for the device.zuid setting.
15. In the Subscribe tab, enter the topic: **zebra/<device.zuid>/+** and select the **QoS 2** option, then click **Subscribe**.



NOTE: Make sure there are no leading or trailing forward slash '/' characters in the topic

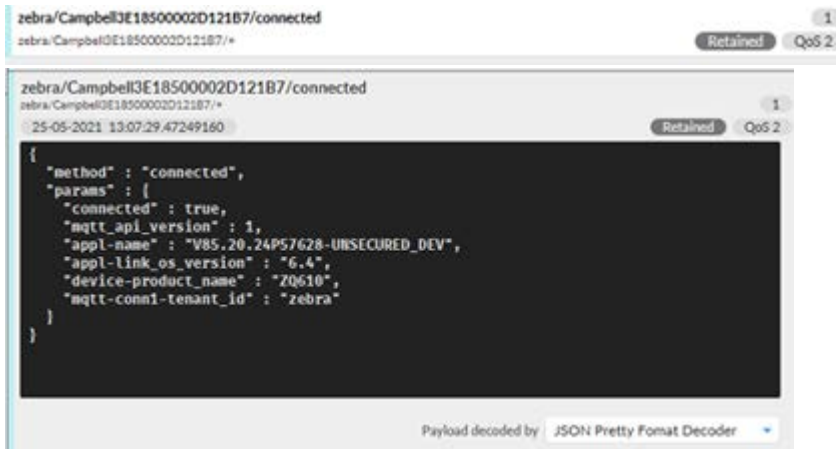


16. Once your printer is connected, you should start seeing the connected topic showing a message from your printer.

Printer Configuration

1. Set the printer to connect to the public Mosquitto TLS MQTT broker using the following settings:


```
{
  "mqtt.enable": "on",
  "mqtt.conn1.server_address": "mqtt://test.mosquitto.org:8883",
  "mqtt.conn1.tenant_id": "zebra"
}
```
2. Take the CA Certificate file used previously in connecting MQTT.fx and save it on the printer as MQTT1_CA.NRD to the E: FLASH drive.
3. Restart your printer and verify the connected message shows up as it connects to the broker. See the connected topic below for more details. Once you see a message on the connected topic with a payload that shows `connected:true`, you are connected!



This screenshot contains developer only information and should be retaken.

4. You can now configure the printer to send alerts to the broker using the following setting:


```
{
  "alerts.add": "HEAD OPEN,MQTT,Y,Y,1,,N,"
}
```

Sending your First Command via MQTT

Now that you are connected, you can send a simple command to the printer and get a response all over MQTT!

1. First, you will need to create a message for the command topic using the settings method. Let's start by setting the QoS to 2 and making sure Retain is not pushed in the **Publish** tab of the client.

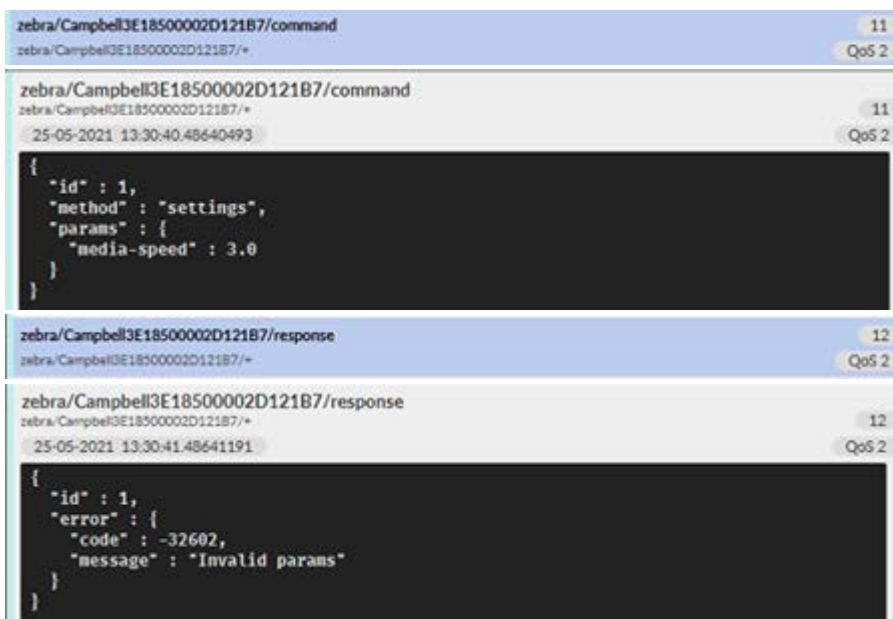
2. Fill out the topic path using your tenant, and device ID to which you previously subscribed.
3. Next, enter the payload for that topic. For this example, we will use the settings method to set and get the media speed. Click **Publish**.



4. In the **Subscribe** tab, you should see the command to set the media speed, on the command topic, and a response of the printer showing if it was able to set that value or not in the response topic.



5. Notice that if you attempt to set a setting with an invalid value (like not using a string), you will get an error.



Connecting to Mosquitto

MQTT TLS 8884

The information in this section is reliable when it is created.

This section is similar to the other Mosquitto MQTT TLS 8883 section but requires the use of client certificate on the printer. For this broker, the client certificate must be signed by the Mosquitto web page, and cannot use the default generic printer client certificates. This also assumes that the printer is already connected on MQTT conn1 to the 8883 broker. Please follow those steps ([Connect a Client to the Broker on page 52](#)) before attempting this section.

Connect a Client to the Broker

This assumes:

- You are using the second connection available (conn2) for MQTT configuration.
- The default port on the broker is 8884.
- The default topic path (Zebra) for the tenant is acceptable.
- The network your printer is on can reach the Internet.
- You have access to MQTT.fx version 1.7 or newer.
- You have access to OpenSSL command line or some other mechanism of generating a private key and CSR (certificate signing request) to use with MQTT.fx.

Instructions

1. In MQTT.fx, create a new connection profile for the test Mosquitto broker.
2. Fill out the **Profile Name** to Mosquitto TLS 8884.
3. Set the **Broker Address** to test.mosquitto.org.
4. Set the **Broker Port** to 8884.
5. Click **Generate** to create your Client ID.
6. Check the **Auto Reconnect** box in the **General** tab.

- Click **Apply**.

- Generate the private key and CSR for the MQTT.fx client.
For reference, here is something to use with OpenSSL:

```
openssl ecparam -out ec_params.pem -name prime256v1
openssl req -newkey ec:ec_params.pem -days 30 -new -keyout "client_key.key" -out "client_cert.csr" -verify -subj "/C=US/ST=Illinois/L=Lincolnshire/O=Zebra Technologies/CN=MQTT Client" -config ./openssl.cnf -nodes
```
- From a browser, go to test.mosquitto.org/ssl and paste the contents of the client_cert.csr file into the text box and click **Submit**.

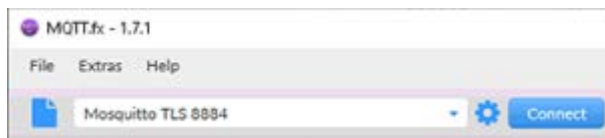
Paste your CSR here

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBJTCBzAIBADBqHQswCQYDVQQGEwJVUzERMABGA1UECAwISWxsah5vaXNxFtA
T
BgNVBACMDExpbnNvbG5zaGlyZTEbMBkGA1UECgwSbWVicmEgVGVjaG5vbG9naWV
z
MRQwEgYDVQQDDAtNUVRUIENsahVudDBZMBMGBYqGSM49AgEGCCqGSM49AwEHA0I
A
BEahq6Wc117yQ3BhWn2YYOn9oV7m4pHVP1hhcg5tLn8pxsJHf7mkHSG5161ATc
9
+ZJTXcbeKQYyuElVuJ08hRCgADAKBggqhkJOPQQDAgNIADBFA1EAq+T0toWlyuc
q
qICSWdiqpgCQ2iGDPTRFbiL7E/c7lUMCIFSAAr+ubeOUvWVCUF/PFO0jFfzcSwr
v
xudjCo92GZLD
-----END CERTIFICATE REQUEST-----
```

- In the **SSL/TLS** tab, check the **Enable SSL/TLS** box.
- Back in MQTT.fx, select the **Self signed certificates** radio button.
- Select the same CA file that was used for the 8883 Mosquitto broker.
- Select the private key that was generated for the key file.
- Select the downloaded client.crt file that was signed from the web page.

15. Click **Apply**.

16. Cancel out of the box, select your new profile, and click **Connect**.



17. Once connected, you should see a green status and lock icon on the right.

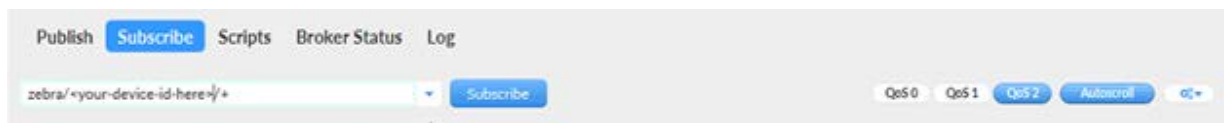


18. Query your printer to find out what its client ID will be using for the `device.zuid` setting.

19. In the **Subscribe** tab, enter in the topic: `zebra/<device.zuid>/+` and select the **QoS 2** option, then click **Subscribe**.



NOTE: Make sure there are no leading or trailing forward slash '/' characters in the topic.



20. Once your printer is connected, you should start seeing the connected topic showing a message from your printer.

Printer Configuration

1. Set the printer to connect to the public Mosquitto TLS MQTT broker using the following settings:
2. Take the CA Certificate file used previously in connecting MQTT.fx client, and save it on the printer as MQTT2_CA.NRD to the E: FLASH drive.
3. Have the printer generate its own CSR by using the provisioning-generate_csr command method (using MQTT1 connection from the previous section):

```
{
  "method": "provisioning-generate_csr",
  "params": {
    "service": "MQTT2",
    "message_digest": "sha256",
    "CN": "MQTT Printer",
    "names": [
      { "C": "US", "L": "Lincolnshire", "O": "Zebra Technologies", "ST": "Illinois" }
    ],
    "key": {
      "algo": "ecdsa",
      "size": 256,
      "curve": "prime256v1"
    }
  },
  "id": 61
}
```

Once complete, take the CSR file that was generated and use it below.

4. From a browser, go to test.mosquitto.org/ssl and paste the contents of the CSR file into the text box and click **Submit**.

Paste your CSR here

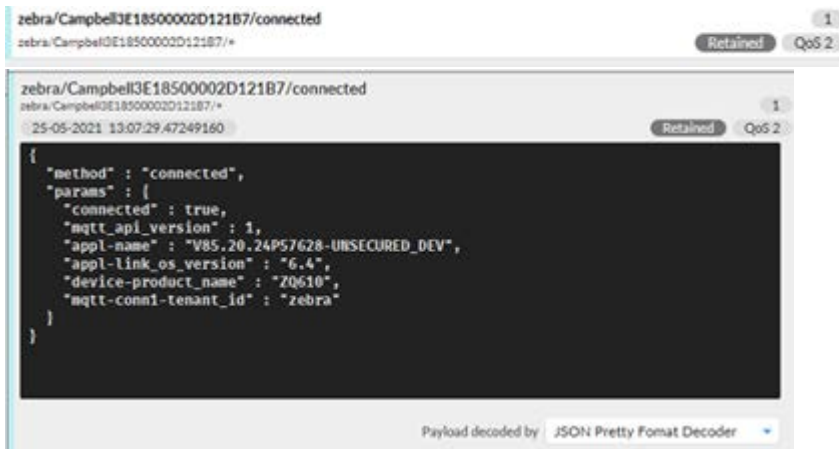
```
-----BEGIN CERTIFICATE REQUEST-----
MIIB7TCBzAIBADBqHQswCQYDVQQGEwJVUzERMABGAIUECAwISHhsaW5vaXNFTFA
T
BgNVBACmNDExpbnVvG5zaGlyZTEbMBkGA1UECgwShmVlcmEgVGVjaG5vbG9naWV
=
MRQwEgYDVQQDDAtNURVUIENsaiVudDBZHBMBGByqGSM49AgEGCCqGSM49AwEHA0I
A
BEahq6Wc117yQ3BhN2YYOn9oV7m4pHVP1hhcg5tLnBpxsJHf7mkHSG5161ATc
9
+ZJTxcbeKQYyuElVuJ08hRCgADAKBggqhkJOPQDAgNIADBFA1EAqp+T0toWyuC
q
qICSWdiqpgCQ2iGDpTRFb1L7E/c7lUMCIFsAAr+ubeOUvWVCUF/PFO0jFfzcSwr
V
xudjCo92GZLD
-----END CERTIFICATE REQUEST-----
```

Submit

5. Open the certificate file that was downloaded in a text editor. Replace all binary line feeds with string escaped line feeds (\r\n with \\r\\n) and copy the contents.
6. Paste the contents into the provisioning-place_cert command method (using MQTT1 connection from the previous section) by filling in the cert section:

```
{
  "method": "provisioning-place_cert",
  "id": 62,
  "params": {
    "service": "MQTT2",
    "cert": "
-----BEGIN CERTIFICATE-----\r\n
MIICNzCCAD6gAwIBAgIBAJAKBggqhkJOPQDAjB4MQswCQYDVQQGEwJVUzEVMBMG\r\n
nhFRgDwX6TAFBgNVHSMEGDAwGBo/6yZbW0ypnHqHF+8s6w0zAk1fuzAMBGNVHRME\r\n
BTADAQH/MAoGCCqGSM49BAMCA0cAMEQCIEOfDez2Ehjw9AQFkwrUSqADOPvK/qB0\r\n
Fli14pa51y4yAiBLoeQc0cc0ow8v4R86R9ZD9QpumJKyKnTVqmyWEWiciw==\r\n
-----END CERTIFICATE-----"}
}
```

- Restart your printer and verify the connected message shows up as it connects to the broker. See the connected topic below for more details. Once you see a message on the connected topic with a payload that shows `connected:true`, you are connected!



```
zebra/Campbell3E18500002D121B7/connected
zebra/Campbell3E18500002D121B7/+
Retained QoS 2

zebra/Campbell3E18500002D121B7/connected
zebra/Campbell3E18500002D121B7/+
25-05-2021 13:07:29.47249160
Retained QoS 2

{
  "method": "connected",
  "params": {
    "connected": true,
    "mqtt_api_version": 1,
    "appl-name": "V85.20.24P57628-UNSECURED_DEV",
    "appl-link_os_version": "6.4",
    "device-product_name": "Z0510",
    "mqtt-conn1-tenant_id": "zebra"
  }
}
```

Payload decoded by: JSON Pretty Format Decoder

This screenshot contains developer only information and should be retained.

- You can now configure the printer to send alerts to the broker by using the following setting:
`{{"alerts.add": "HEAD OPEN, MQTT, Y, Y, 1, , N, "}}`

