Timestamps and the MQTT modules

In general, the Edge is the source of truth for tag timestamps although there are several events that will cause the timestamps to differ across the system:

In a BIRTH message

- The BIRTH message for each Edge Node and Device will set the timestamp associated with each tag to the current time of the publishing client server's OS.
- This means that the timestamp at the Edge will show the last known timestamp and the receiving application will display the timestamp included in the BIRTH message.

OPC Tag Disabled

- When an OPC tag is disabled, the timestamp at Edge will reflect the time the tag was disabled.
- The associated DDATA message will reflect the timestamp of the disabled tag change event.

OPC Tag Enabled

- When an OPC tag is enabled, the timestamp at Edge will reflect the last known change timestamp from the PLC.
- The associated DDATA message will reflect the timestamp of the enabled tag change event.

OPC Restart Tag

- When an OPC tag is restarted, the timestamp at Edge will reflect the last known change timestamp from the PLC.
- The associated DDATA message will reflect the timestamp of the tag restart event.

The timestamp for an Edge tag read from a PLC will be the timestamp assigned by the PLC and/or associated Ignition driver.

The timestamp for any other Edge tag will be the Ignition server OS time for change in Qualified Value (Value, Quality or Timestamp).

All timestamps published on the wire are in UTC and displayed converted to local time at the receiving application.

If times are not synchronized across the system components, for example by using an NTP server, its possible that the time at the PLC is older than other system components. If using MQTT Engine, this can result in tag change events being ignored as the new event time is before current time for the tag at MQTT Engine.

Common scenarios where timestamps can be different across the MQTT system

Let's assume an Edge Node has a single tag named Tag1

MQTT Transmission Store and Forward Not Enabled

1. Transmission connects and publishes its BIRTH
    a. The timestamp for Tag1 set at the Edge to 'now' per the Edge's system clock.

2. Tag change events occur naturally at the Edge
    a. During this time the timestamp is always set using the tag change event time at the Edge. This results in Tag1's timestamp at Engine matching that of the Edge.

3. Transmission loses connection.
    a. At this point, Engine 'stales' the tag by setting the quality to BAD_STALE and sets the timestamp of Tag1 to 'now' per the Central Gateway's system clock.
    b. The tag may or may not be changing at the Edge - Engine won't know in real time this is happening so it sets the quality of Tag1 to BAD_STALE.

4. Transmission reconnects and publishes its BIRTH
    a. The timestamp for Tag1 set at the Edge to 'now' per the Edge's system clock.
    b. Assuming the quality of Tag1 at the Edge is GOOD - it will be set to GOOD at Engine.
    c. If the tag value had never changed at the Edge, then you would see three events with the same value, three different timestamps, and the quality go from GOOD -> BAD_STALE -> GOOD.

MQTT Transmission Store and Forward Enabled with Flush history in-order (synchronously)

1. Transmission connects and publishes its BIRTH
    a. The timestamp for Tag1 set at the Edge to 'now' per the Edge's system clock.

2. Tag change events occur naturally at the Edge
    a. During this time the timestamp is always set using the tag change event time at the Edge. This results in Tag1's timestamp at Engine matching that of the Edge.

3. Transmission loses connection
    a. Transmission caches a BIRTH to the history store with the timestamp for Tag1 set to 'now' per the Edge's system clock.
    b. All changes to Tag1 at the Edge are stored to the configured History Store.
    c. At this point, Engine 'stales' the tag by setting the quality to BAD_STALE and sets the timestamp of Tag1 to 'now' per the Central Gateway's system clock.
    d. The tag may or may not be changing at the Edge - Engine won't know in real time this is happening so it sets the quality of Tag1 to BAD_STALE.

4. Transmission reconnects and publishes the cached history store BIRTH
    a. The timestamp for Tag1 set at the Edge to 'now' per the Edge's system clock.
    b. The history store is flushed before any live data is published. This means that any changes to Tag1 will be stored to the history store until the flush is complete.
    c. Assuming the quality of Tag1 at the Edge is GOOD - it will be set to GOOD at Engine.
    d. If the tag value had never changed at the Edge, then you would see three events with the same value, three different timestamps, and the quality go from GOOD -> BAD_STALE -> GOOD.

MQTT Transmission Store and Forward Enabled with Flush history (asynchronously)

1. Transmission connects and publishes its BIRTH
    a. The timestamp for Tag1 set at the Edge to 'now' per the Edge's system clock.

2. Tag change events occur naturally at the Edge
    a. During this time the timestamp is always set using the tag change event time at the Edge. This results in Tag1's timestamp at Engine matching that of the Edge.

3. Transmission loses connection
    a. Transmission caches a BIRTH to the history store with the timestamp for Tag1 set to 'now' per the Edge's system clock.
    b. All changes to Tag1 at the Edge are stored to the configured History Store.
    c. At this point, Engine 'stales' the tag by setting the quality to BAD_STALE and sets the timestamp of Tag1 to 'now' per the Central Gateway's system clock.
    d. The tag may or may not be changing at the Edge - Engine won't know in real time this is happening so it sets the quality of Tag1 to BAD_STALE.

4. Transmission reconnects and publishes the cached history store BIRTH
    a. The timestamp for Tag1 set at the Edge to 'now' per the Edge's system clock.
    b. The history store will be flushed whilst live data is published. This means that any changes to Tag1 will be interleaved with the historic data publishes.
    c. Assuming the quality of Tag1 at the Edge is GOOD - it will be set to GOOD at Engine.
    d. If the tag value had never changed at the Edge, then you would see three events with the same value, three different timestamps, and the quality go from GOOD -> BAD_STALE -> GOOD.