FAQ: Sparkplug

- Q: What is Sparkplug in a nutshell?
  - A: Sparkplug is a set of definitions on top of MQTT to serve the following purposes:
    - Define a set of topics to ensure state/quality of data be ensured in a backend MQTT client application
    - Define a standard payload format that allows an edge node/device client to communicated with a backend application
    - Define a flow of messages to ensure the state/quality of data.
- Q: What is the difference between Sparkplug A and Sparkplug B?
  - A: The 'A' and 'B' qualifiers differentiate payload encoding formats. The following points show the similarities and differences between the two:
    - Important note! Sparkplug A has been deprecated and any new projects being created should use Sparkplug B. If you need a copy of the Sparkplug A specification you can contact support@cirrus-link.com to get a copy.
    - Both A and B use the same topic format with a qualifier as the first token in the topic namespace to denote the payload encoding format
    - The message flow of both A and B is the same
    - The only difference is the payload encoding format.
    - Both formats are open source
    - Both are based on Google Protobuf definitions
    - Sparkplug A uses Eclipse Kura's payload definition found here: https://raw.githubusercontent.com/eclipse/kura/develop/kura/org.eclipse.kura.core.cloud/src/main/protobuf/kurapayload.proto
    - Sparkplug B uses an expanded definition allowing for more metadata found here: https://raw.githubusercontent.com/Cirrus-Link/Sparkplug/master/sparkplug_b/sparkplug_b.proto
- Q: What is the difference between a Template definition and instance?
  - A: There are two parts to a Template
    - Template Definition
      - This metric will will contain all the member Metrics with their default values/properties, as well as any Parameters with their default values. This metric is always only published in the NBIRTH message
    - Template Instance
      - This metric is one or more "instances" of the above definition. It will contain the any member metrics with values /properties that are overridden/different from the defaults in the Template Definition. They are published as you would normal tags (BIRTH/DATA/CMD). When published in a NDATA/DDATA/NCMD/DCMD message they may only contain a subset of the member Metrics (if those are the only values that have changed)
    - Other template fields
      - is_definition
        - true for a Template Definition metric, false for a Template Instance Metric
      - template_ref
        - This is used by a Template Instance to specify the "name" of the Template Definition
      - is_historical
        - Used to mark a Template Instance (or any Metric) as a historical (rather than a live) value. MQTT Engine may handle historical Metrics differently depending on how it is configured (see MQTT Engine documentation)
      - is_transient
        - Not currently used, will be ignored by MQTT Engine, can be omitted or set to false
      - version
        - Not currently used, will be ignored by MQTT Engine, can be omitted or set to null
    - This template_samples.txt contains some sample template structures (in JSON) as well as some Java code for creating a Template. These should help show the basic Template structure and contain the important fields of a Template Definition and Instance. However, they are just references expressed in JSON, not actual protobuf representations (which may be a bit more verbose)