Authentication (HTTP Digest Authentication)

If the eGauge meter is set with site-wide password authentication, all resources will require HTTP Digest Authentication. Many programming languages and libraries have support for HTTP Digest Authentication, so well maintained and trusted libraries for HTTP Digest Authentication are preferable to use over custom implementations. In the event there is no existing support for HTTP Digest Authentication, it can be manually implemented. eGauge Systems cannot provide assistance on this implementation.

Wikipedia has information and examples of how to perform Digest Authentication and what header values must be returned. Note, any protected resources under the XML API will use a "quality of protection" (qop) of "auth", which will affect how the header values are calculated using HTTP Digest Authentication.

The basic steps to perform HTTP Digest Authentication are:

1. Make an unauthenticated request to the resource you wish to access.

2. Look at the WWW-Authentication header returned. Store the following values:
   `Digest realm` (realm space of protected resource. Generally this is "eGauge Administration" unless the interface is custom-branded)
   `nonce` (server nonce, changes for each unauthenticated request)
   `qop` (always will be "auth" for the eGauge XML API)

3. Perform the following calculations:
   `HA1 = MD5(username:realm:password)`
   `HA2 = MD5(method:digestURI)` (note the *digestURI* is the path of the resource, not including host or FQDN. E.g., */cgi-bin/egauge-show*)
   `response = MD5(HA1:nonce:nonceCount:cnonce:qop:HA2)` (*nonceCount* may be "1" for the initial request. *cnonce*
   is a nonce generated by the client, such as a 64 bit hexadecimal value)

4. Send a request to the same URI in step 1, and this time include an "Authorization" header with the following parameters and values:
   1. `Digest username` = username for authentication
   2. `realm` = realm (from step 2)
   3. `nonce` = server nonce (from step 2)
   4. `uri` = request uri (used in HA2 in step 3)

5. `response` = response (generated in step 3)
6. `qop` = "auth"
7. `nc` = nonce count (can simply be "1" for testing, should be incremented for subsequent requests that reuse the same *nonce*)
8. `cnonce` = client nonce generated in step 3

> It is not always necessary to perform step 2 in this process if making repeated requests to the same resource (URI or "digestURI"). A server nonce expire after 10 minutes and attempting to re-use will result in a 401 with WWW-Authenticate header like found in step 1, with a "stale=true" parameter in the header string.
