OpenC2 Producer / Consumer Needs

From Dave Lemire, G2 (originated December 2017)

I've tried to take the ecosystem ideas that Danny Martinez articulated in his OpenC2 Ecosystem Considerations paper and focus in on what producers and consumers need to operate within the ecosystem. I'm keeping in mind that we're working on a Version 1 specification, and not shooting for the moon, with a focus on the things beyond creating and understanding OpenC2 command / response messages. Also, saying a producer or consumer "needs" something here isn't the same as saying they soley implement it or that it's automatically part of the OpenC2 language; finding the balance between what's within OpenC2 and what's supplied by the ecosystem is the goal. As such, some of these things should perhaps be *explicitly* acknowledged as "outside the scope of the OpenC2 protocol" and assumed to be provided by the implementation / environment. Again, there's a balancing act there: having too much out-of-scope will hinder interoperability.

An OpenC2 Producer needs:

- A means to be directed to send commands (i.e., assuming the Producer and its Actuators implement *only* the IACD Acting function, something [the Decision Making function] has to provide direction)
    - Invocation API or message interface
    - Clearly outside the scope of OpenC2
- Unique identity (at least within its "community")
    - How this is defined / expressed / configured seems outside the scope of OpenC2
- Inventory (w/CRUD) of actuators / consumers available to be commanded
    - Identity for each actuator / consumer
    - Profile(s) supported by each available actuator / consumer
    - Implementation-specific, outside scope of OpenC2
- Communications channel(s) to available actuators / consumers
    - Means to
        - Address (individual / groups of / all available) actuators / consumers
        - Send commands to (individual / groups of / all available) actuators / consumers
        - Receive responses from actuators / consumers
    - Confidentiality of communications between producers and consumers
    - This has always been explicitly outside the scope of OpenC2, but can (should?) be captured in a transport specification or implementation profile.
    - Anticipate a variety of communications channels will be used by varying implementations
        - HTTPS API

        - Pub / Sub messaging services
        - Others?
- Means to authenticate
    - Itself to actuators / consumers it is commanding
    - Actuators / consumers when they respond
    - Ideally authentication should be a feature of the communication channel
- Integrity protection
    - Applied to commands sent
    - Checked on responses received
    - Ideally integrity protection should be a feature of the communication channel
- Version negotiation / agreement between Producer and Consumer
    - At minimum, clear indication of language (and profile?) version in command messages or the communications channel
- Mechanism for producer-unique (/ globally-unique?) identification of messages sent
    - Currently (Mar 2018) being proposed as mandatory top-level field in OpenC2 commands
    - Could be an attribute of the communications channel
- Ability to associate response messages received with commands sent
    - If the L-SC agrees on a mandatory command-id field, that will satisfy this requirement
- Ability to track state of command execution over time for commands sent
    - Accept multiple responses for single command as execution progresses
    - A mandatory command-id field will facilitate this

An OpenC2 Consumer needs:

- Unique identity (at least within its "community")
- Inventory of producer(s) authorized to command it via OpenC2
- Communications channel(s) with authorized producers
    - Means to receive commands from authorized producers
    - Means to return responses to authorized producers
    - Confidentiality of communications between producers and consumers
- Way to authenticate
    - Itself to producer
    - Producer(s) from which it receives commands
- Integrity protection
    - Checked on commands received
    - Applied to responses sent
- Version negotiation / agreement between Producer and Consumer
    - At minimum, clear indication of language (and profile?) version in response messages or the communications channel
- Self-knowledge of the actuator profile(s) it supports

- Ability to associate response it sends back to commands received from authorized producer(s)
- Ability to track state of command execution over time for commands received
    - Produce multiple responses for single command as execution progresses