oneM2M's Value Proposition to Application Developers

Dale Seed

oneM2M ARC Chair

Convida Wireless

IoT deployments are getting more complex

- Many IoT deployments typically start off small (e.g. PoCs)
  - Typically focused on a single use case within a single vertical
  - Typically starts off in a controlled environment having limited complexity
- As IoT deployments evolve, so does their complexity
  - Complexity is often fueled by the need to connect up diverse types of IoT devices to satisfy deployment requirements
  - Rarely do you find all devices needed for a given deployment based on the same set of technologies

Example Use Case Deployment – Smart City

Smart City deployments commonly have a very diverse set of IoT devices based on many different types of technologies.

Source: www.itproportal.com

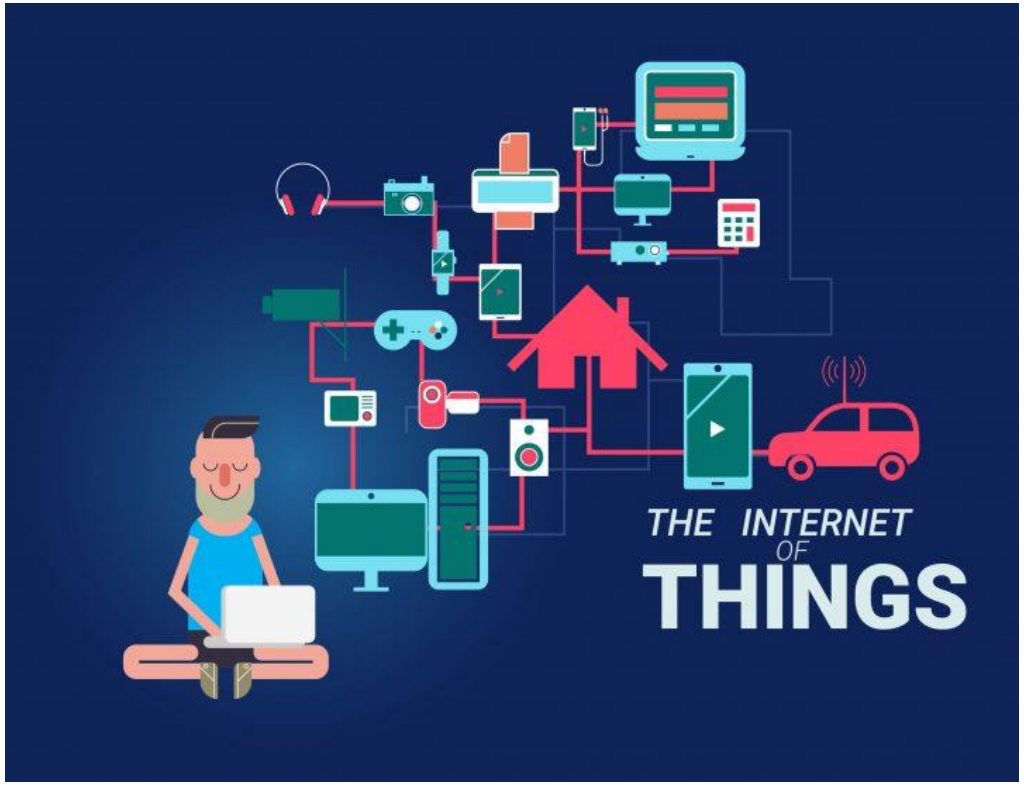
Some examples of varying technologies:

- **Underlying Networks**
- **Security Frameworks**
- **Transport Protocols**
- **Content Serializations**
- **Service/Management**
- **Semantic Ontologies**

The IoT App Developer

- IoT App Developers build apps that interact with IoT devices

- Typical types of interaction
  - Collect and analyze sensor readings
    - E.g. Monitor and predict traffic congestion
  - Issue commands to actuator
    - E.g. Turn on/off valves
  - Manage IoT devices
    - E.g. Upgrade a device's firmware



Source: www.ircrwireless.com

Typical IoT App Developer “Pain Points”

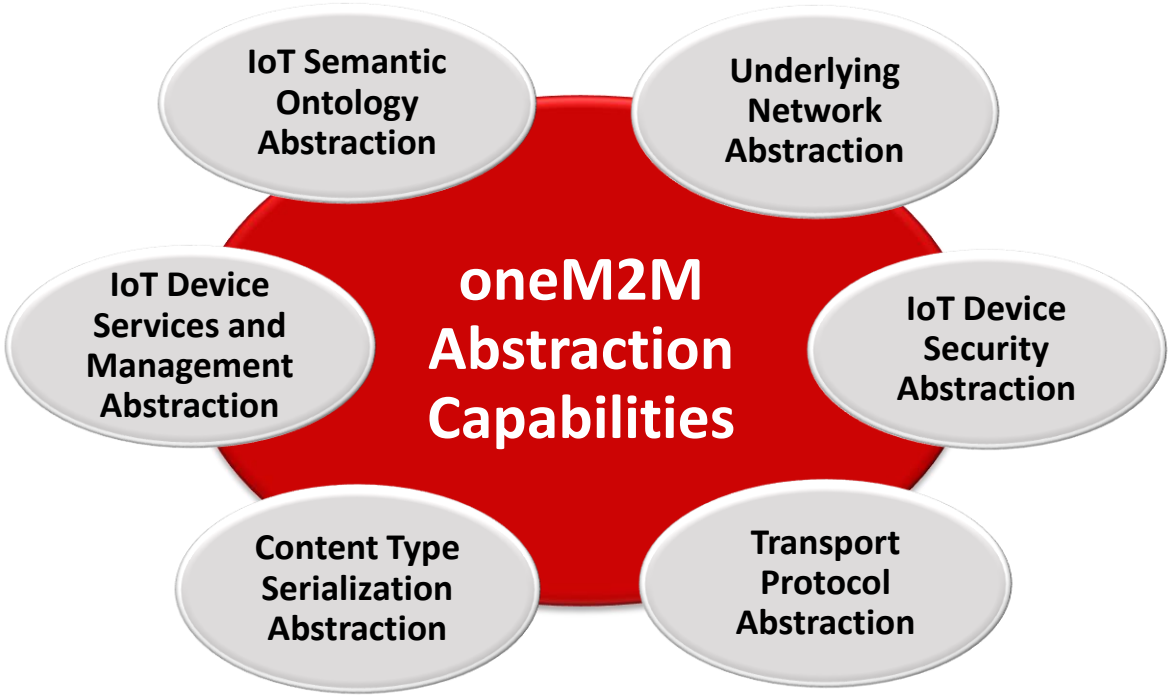
→ Lots of different IoT devices that use varying combinations of technologies

IoT Device Underlying Networks	• E.g. Cellular, Fixed, Wi-FI, …
IoT Device Security Models	• E.g. DTLS, TLS, PSK, PKI, …
IoT Device Transport Protocols	• E.g. HTTP(s), CoAP(s), MQTT(s), WebSockets, …
IoT Device Content Serializations	• E.g. XML, JSON, CBOR, Plain-Text, …
IoT Services and Management Technologies	• E.g. OCF, LWM2M, Thread, OMA DM, TR-069, …
IoT Device Semantic Ontologies	• E.g. SAREF, W3C TD, …

→Building an app that targets multiple device types can be challenging

oneM2M's Value Proposition to Developers

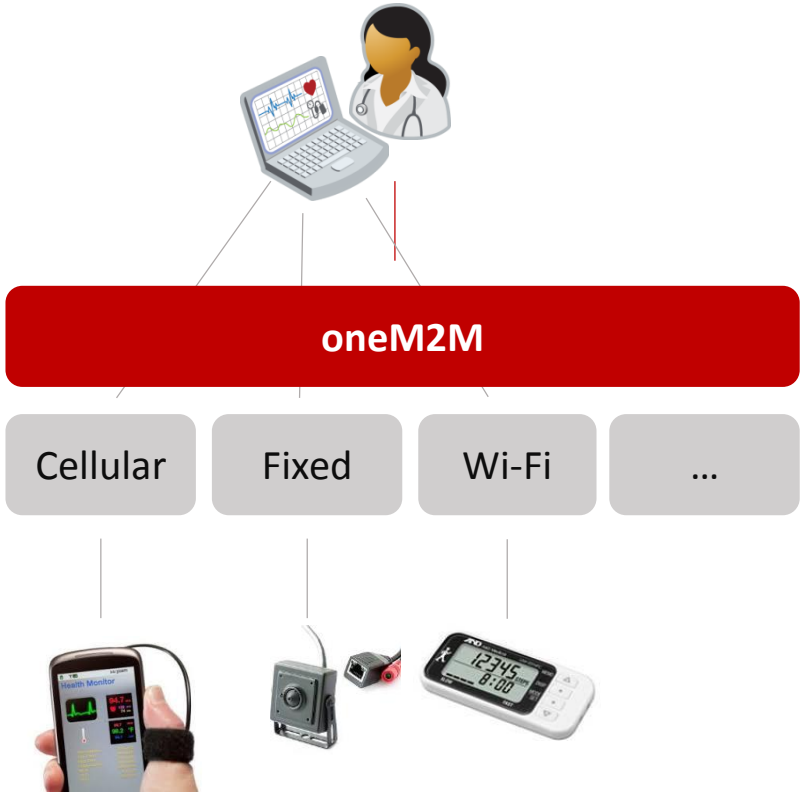
→ *Abstraction, Abstraction, Abstraction!*



→ *Via its capabilities to abstract, oneM2M hides from the App Developer the complexities involved with interacting with a diverse set of IoT devices*

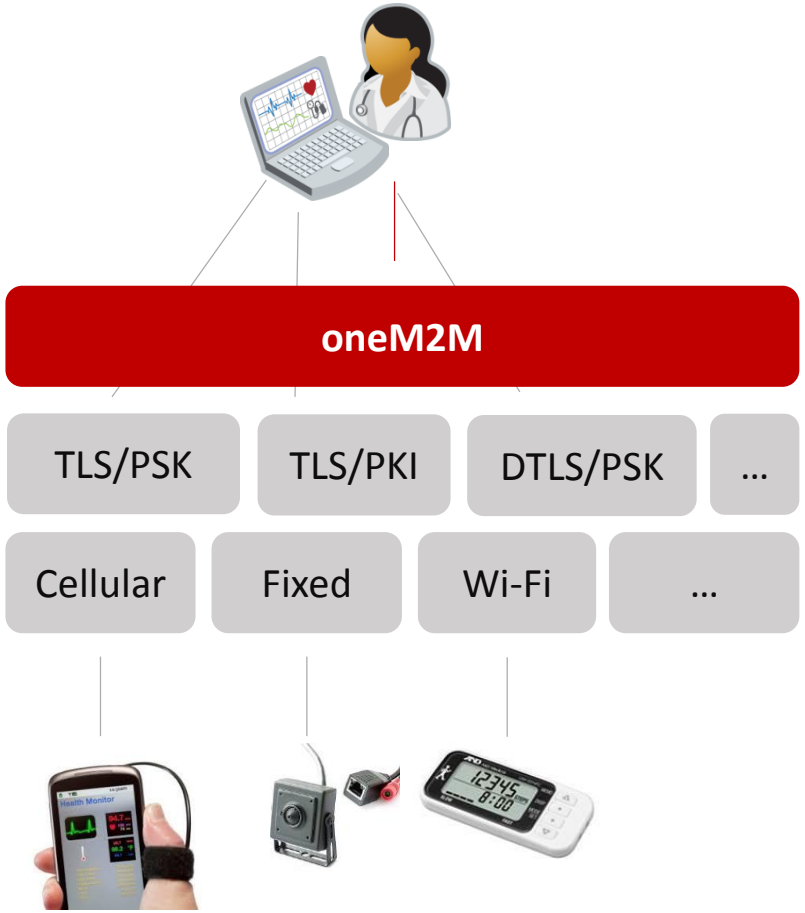
Underlying network abstraction

- oneM2M manages the connectivity and communication to IoT devices on behalf of the apps
  - Scheduling and buffering of messages based on device reachability
  - Selection of underlying network connectivity options for device communication
  - Triggering of devices to establish a network connection based on when apps need to communicate with devices



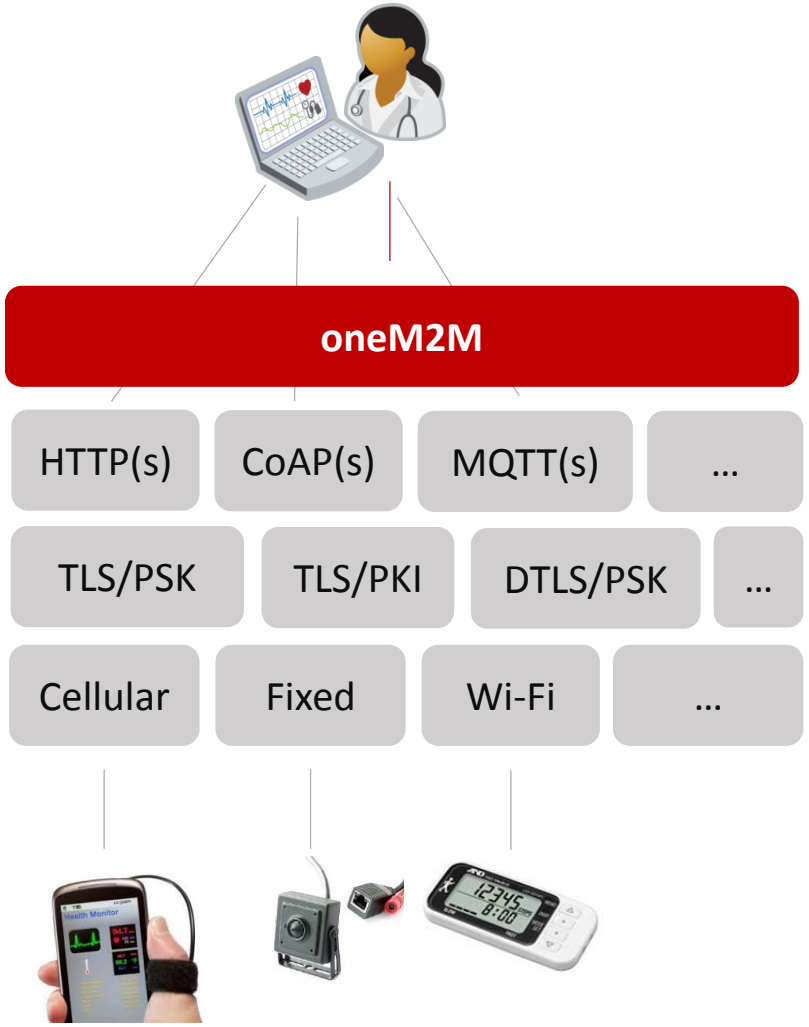
IoT device security abstraction

- oneM2M hides the different security frameworks of each IoT device technology from the App Developer.
- A Developer's app can establish a security association with the oneM2M service layer and via this security association, communicate securely with IoT devices
- The oneM2M service layer establishes and manages the security association with each of the IoT devices on behalf of the app
  - Enrolment, credential bootstrap/management, authentication, integrity, privacy, and authorization network connectivity of the devices from the app developer.



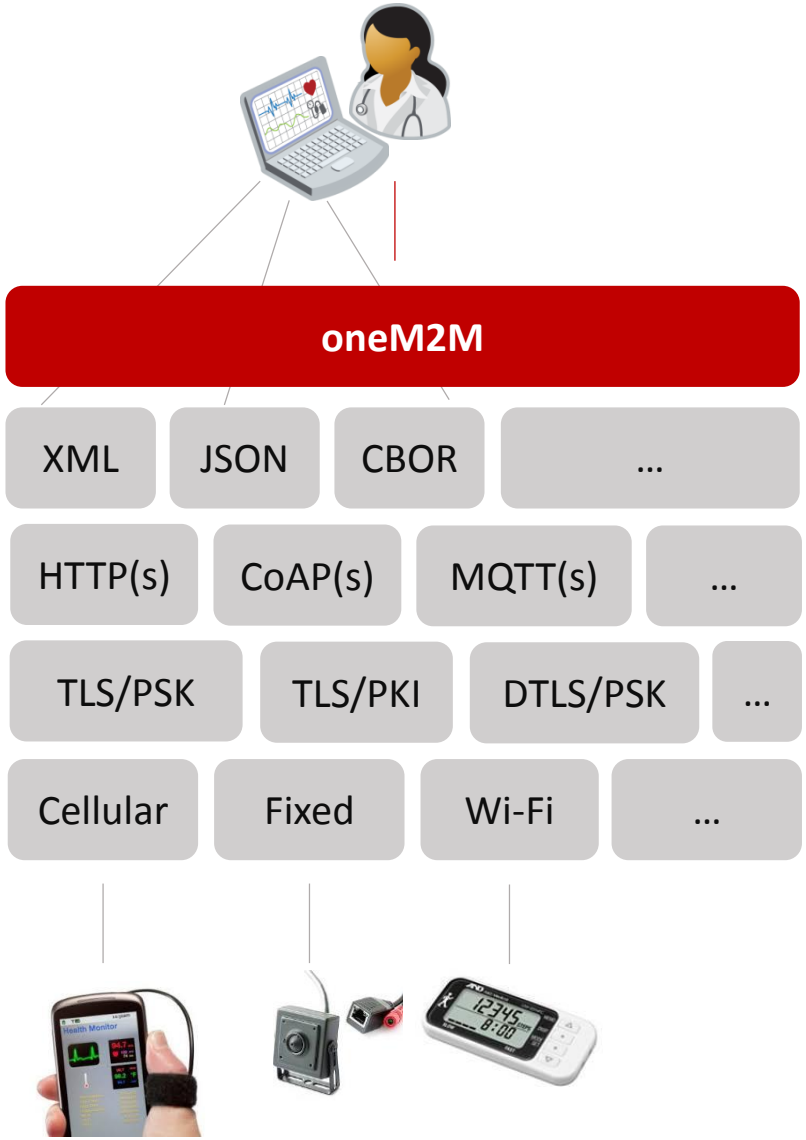
Transport protocol abstraction

- oneM2M hides the different transport protocols used by the devices from the App Developer.
- Applications can use different transport protocols than the different devices they choose to communicate with
  - E.g. HTTP(s), CoAP(s), MQTT(s), WebSockets
- oneM2M will handle converting the transport protocol so the App Developer does not need to



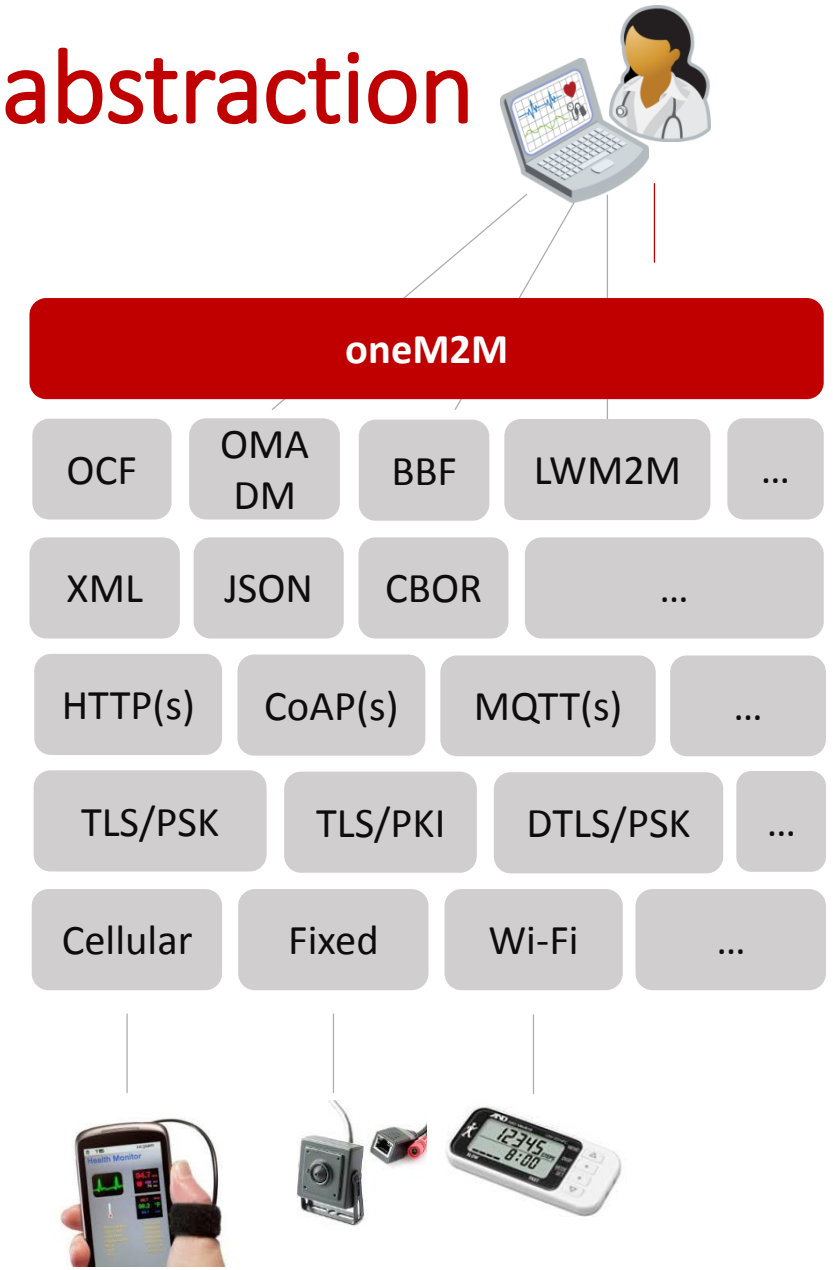
Content serialization abstraction

- oneM2M hides the different content serializations used by the devices from the App Developer.
- Applications can use different types of content serialization formats than the one or more devices they choose to communicate with
  - E.g. XML, JSON, CBOR, Plain-Text
- oneM2M will convert the content serialization format so the App Developer does not have to



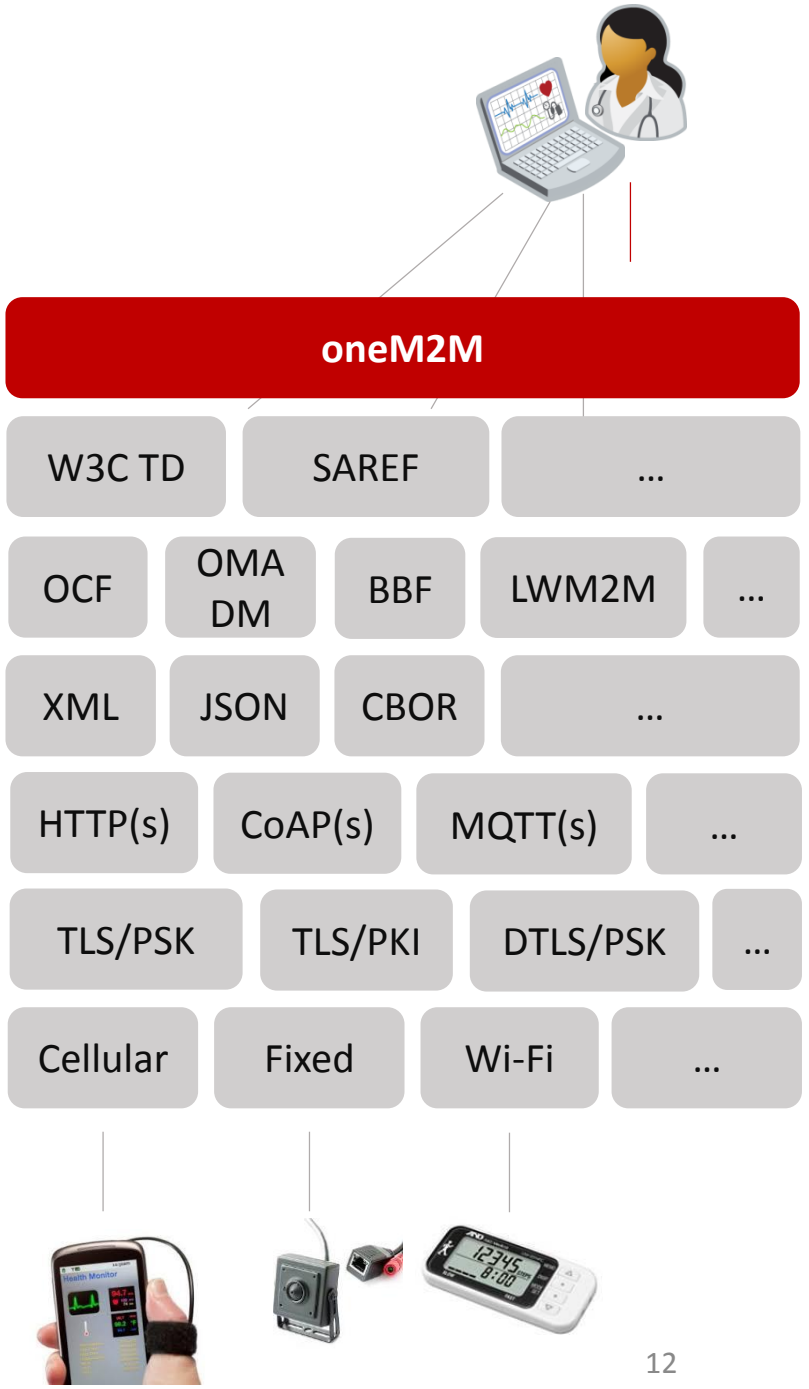
IoT device services and management abstraction

- oneM2M interworks with various IoT device service enablement and management technologies
  - E.g. LWM2M, OCF, OMA DM, BBF TR-069, …
  - All devices are presented to the App via oneM2M API
  - Via standardized oneM2M API, App developers can use device services and manage devices
- Once the data model is abstracted into oneM2M, App Developers can access all devices in a common manner and make use of oneM2M value-add capabilities such as
  - Resource Discovery
  - Generating Events via subscriptions and notifications
  - Grouping
  - Access Controls



IoT device semantics abstraction

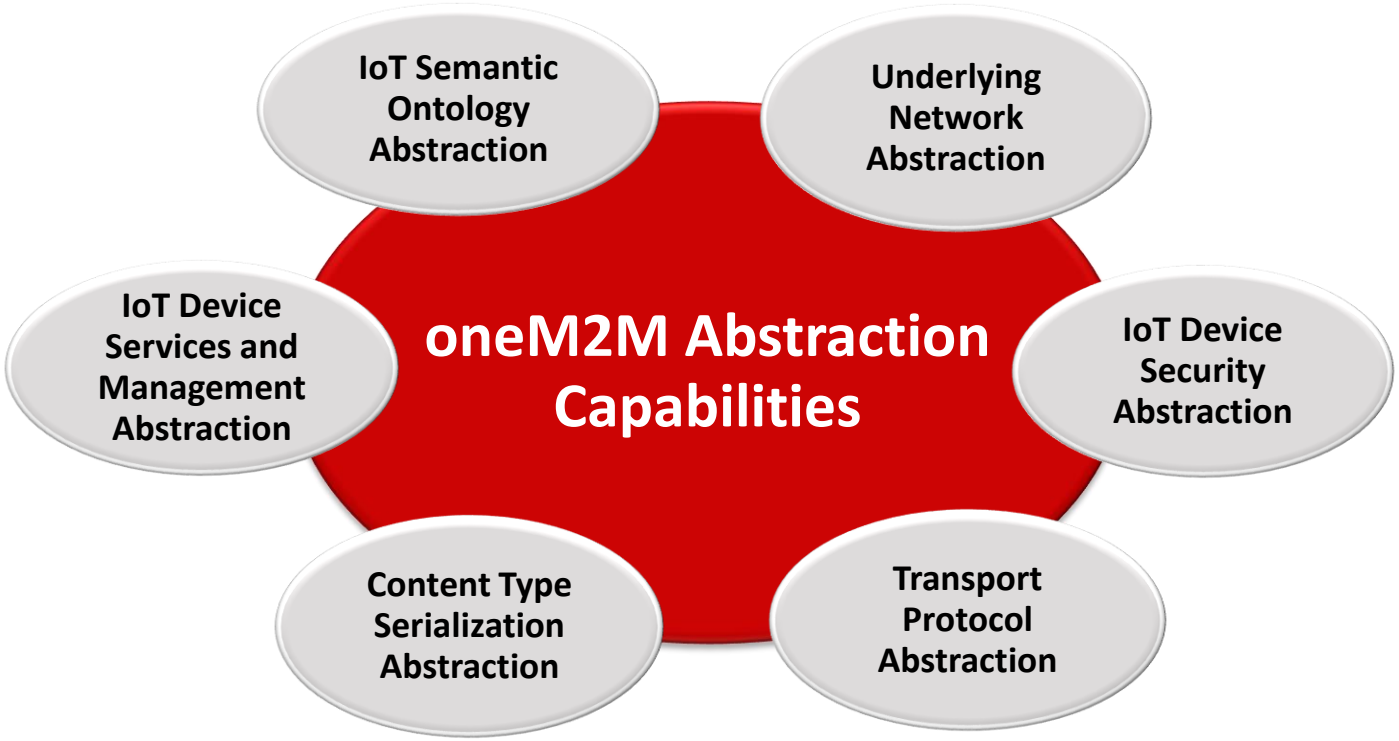
- oneM2M supports a semantic framework
- This framework supports semantic ontology interworking
  - Ontologies defined by other organizations can be used in the oneM2M framework to describe oneM2M resources in a semantic manner
- The framework supports semantic ontology abstraction
  - oneM2M defines its own technology that can be used to describe oneM2M resources in a semantic manner
  - Semantic descriptions expressed in terms of other ontologies can be interworked to oneM2M's Base Ontology to provide abstraction



Takeaways

→ IoT deployments can have diverse types of IoT devices that can cause complexity for IoT App Developers

→ Via its abstraction capabilities, oneM2M is able to hide this complexity from the App Developer!



Thank You!!!

Dale Seed

oneM2M ARC Chair

Principal Engineer

Convida Wireless

Seed.Dale@ConvidaWireless.com