Internet of Things Data Foundations for First Responders Fire Service

FY2020

May 6, 2020

PSCR

Homeland Security

Science and Technology

Internet of Things Data Foundations for First Responders

FY2020

September 30, 2020

Work performed in support of the:
Department of Homeland Security
Science and Technology Directorate
Office for Interoperability and Compatibility

U.S. Department of Commerce
Public Safety Communications Research Program

Contacts:
Alison Kahn
Personal Area Networks Project Lead
alison.kahn@nist.gov | 303-497-3523

Peter Hallenbeck
Softwhere Syzygy, LLC
Deputy Chief (Ret.) Efland Volunteer Fire Dept., Efland, NC
pete@eflandfd.org

Sam L. Ray
PSCR Portfolio Lead—DHS
samuel.ray@nist.gov | 303-497-3262

NIST-CTL PSCR Division
325 Broadway
Boulder, CO 80305

TABLE OF CONTENTS

TABLE OF FIGURES

1 EXECUTIVE SUMMARY

The term *Internet of Things* (IoT) has reached widespread use in the last several years. IoT systems consisting of small hardware devices capable of transmitting data wirelessly for informative or actionable purposes are present in many homes, appliances, cars and accessories. Additionally, the widespread usage of IoT devices and their data is promoting their increasing use in commercial buildings, parking lots, and city infrastructure.

For the past three years, the National Institute of Standards and Technology (NIST) Public Safety Communication Research division (PSCR), in research sponsored by the Department of Homeland Security (DHS) Science and Technology (S&T) Directorate, studied IoT devices for first responders to determine how these devices can be employed to meet the needs of the first responder in the field.

In April of 2019, PSCR held a Public Safety Internet of Things Roundtable at the NIST Laboratories in Boulder, Colorado. During this roundtable, several issues regarding the lack of IoT utilization in public safety came to light. Foundationally, there was a concern that the information required by public safety, as well as the information provided by these devices, was not well documented or consistent and, therefore, could not be reliably utilized between systems. These foundational concerns implied that there could be no expectation of reliable information provided by a system and consequently no widespread usage of IoT devices throughout public safety jurisdictions.

This document reports on interviews conducted with first responder professionals across all fields, in which their current informational and technological requirements were discussed. It will also present a potential model of the informational requirements as independent software objects to demonstrate how a baseline consistency in IoT product development could hypothetically be achieved.

2 INTRODUCTION

Currently, many companies have a presence in the increasingly lucrative IoT industry. Since the hardware in an IoT product is intended to be small and unobtrusive and to last for long periods of time with minimal configuration or maintenance, companies do not base their business models on the purchase of IoT hardware alone. Rather, the industry focuses on being long-term service providers. To that end, when customers purchase IoT devices for use in their homes or businesses, they are typically also purchasing data storage and visualization services from the same company. For simplicity and convenience, this is a very effective model. In the smart home industry and the emerging smart building and smart city industries, where the use cases can be easily defined, this business model works very well.

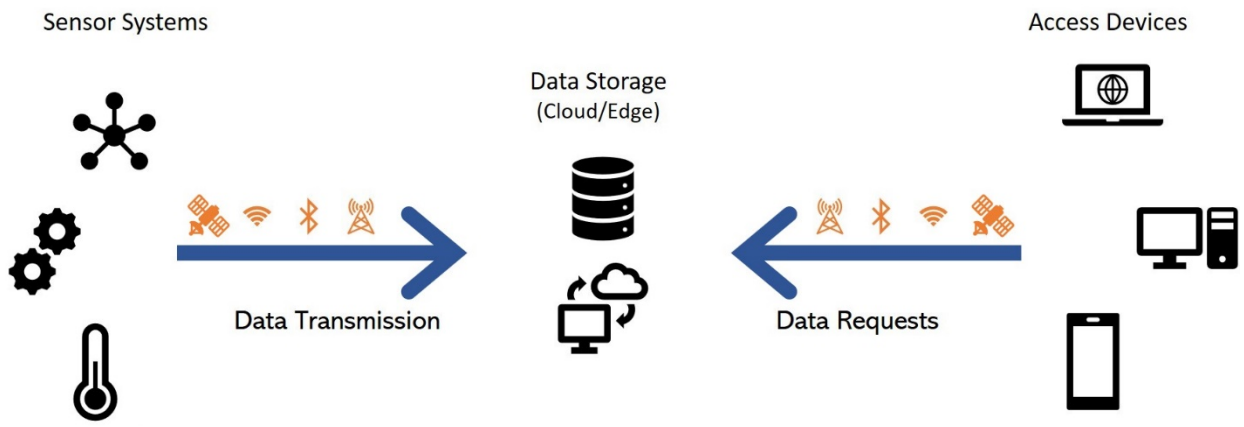

Fig. 1: IoT data flow in a proprietary system

However, the drawbacks to this particular system implementation include an inability for a group or individual to retain access to their own data, an inability to share or migrate data between systems, and limited ability to utilize or view the data—controlled by the developer's predefined use cases. Thus, a private citizen seeking to utilize an IoT application will have to make an informed decision as to which platform best meets his/her needs and decide which features and capabilities are required. These issues are shared by public safety departments and jurisdictions seeking to purchase IoT systems for field use. To perform tasks specific to their working environments, first responders will often need to operate outside the boundaries of general use cases developed for commercial use. Additionally, data access and availability will be critically important to the first responder in the field. (This will be discussed further later in the document.) These are some of the prime issues limiting the usage of IoT systems in the public safety arena today.

If the features of a commercially available IoT platform are not sufficient for first responder usage, the next obvious area to examine is industry-specific IoT applications. When we begin discussing industrial usage of IoT systems, it is important to remember that these industries are defined by similar basic operations. Although individual businesses may differ, distinct categories of operations converge within an industry, allowing for generalized IoT applications. Larger, well-funded companies have the ability to contract out proprietary systems, further customizing the data and operations and—most importantly—allowing the businesses themselves to be the stewards of their own data.

From one perspective, public safety may be considered to be its own industry, specializing in maintaining the health and safety of private citizens and society. However, its primary role is as a public government entity. This means that every locality across the country has some type of public safety governance, and building an IoT product to meet the needs of jurisdictions across the U.S. is impossible. But as with the industries mentioned above, a kernel of similarity is hidden among the differences between departments. The aim of this project is to unearth those similarities through discussion with public safety professionals, determine those which could provide situational awareness, and use that information to document software objects that could be foundational to the development of future public safety-specific sensor systems.

2.1 Challenges in a First responder Environment

When considering the first responder's environment, defining a use case or set of use cases that can be addressed by a single product is a difficult task. From department to department, the first responder's activities may differ based on a multitude of variables:

1. **Location**. The department's locale (e.g., urban, suburban, rural) may dictate the prevalence of different first response events.
2. **Geographical Environment.** The type of environment that the department exists in may play a factor. For instance, a more heavily forested area may be at risk for wildfire, while a coastal area may experience a hurricane risk.
3. **Incident Types.** The types of incidents that a department encounters most frequently will have an impact on the information and tools used. For example, if a department frequently attends to interior structure fires, the firefighters will have severe temperature challenges based on the building floor plan, contents, and the path of the fire itself. If a department frequently attends to automotive accidents, first responders will encounter medical challenges that require a different type of equipment.
4. **Society.** The societal factors surrounding a department will also play a role in the types of responses they may see daily. The economy and income level of the jurisdiction will play a part in the activities and infrastructure in the area.
5. **Department.** The type of department will dictate capabilities and budgetary restrictions. Certain areas will employ full-time departments and/or volunteer staff. Size and resources available to a department will dictate how they respond to incidents.

These variables indicate that although basic informational needs may be the same from department to department, there are specific needs that will vary greatly depending on regional differences. This results in individual departments prioritizing different information and gravitating toward diverse technological solutions to meet their needs.

If public safety is viewed at a departmental level, it can be seen as relatively small groups of people performing diverse activities, depending on the location within the U.S., the size of the population they serve, and other factors. Due to the wide range of these activities and the relatively low potential market share that a single department would provide for IoT devices, there have been few solutions developed for this unique environment. The relative openness of the field also makes public safety a great potential market for expansion if the appropriate requirements can be met. These needs, with respect to IoT devices, can be placed into three categories:

1. **Physical Requirements.** Traditional size, weight and power (SWAP) requirements for first responder equipment must also be met for IoT transmitters. The presence of IoT equipment must not obstruct the first responder's actions in any way.
2. **Operational Requirements.** The equipment must be able to be operated and maintained easily and efficiently. The hardware must be reliable and hardened enough that it will maintain functionality with minimal inputs from the first responder and avoid interfering with their mission.
3. **Informational Requirements.** The information provided by IoT devices must be presented to the first responder in an unobtrusive format. The information should only be displayed when relevant to the first responder's direct environment, and information should be consistent. So,

when multiple sensor platforms are present, either within a single department or in a mutual aid situation, a first responder can have an expectation of a certain level of information availability. Additionally, this consistency would allow certain informational generalizations to be made across first responder fields, rather than using customized information within disparate systems.

As technology--and perhaps more importantly the acceptance of technology--continues to grow in the emergency services world, there is a simple to understand but hard to solve problem. The amount of information and the number of different systems producing and processing information are growing. For these systems to work together, there must be agreement on what this information "looks like." To describe this concept with an analogy, these systems would be like a room full of people trying to work together while speaking different languages. Productivity would be terrible because you would need a second room full of translators. If they spoke the same language, they would be far more productive. The cost of communicating would decrease due to the large savings produced by eliminating the translators. Efficiency would improve because a simple message transmitted between two people would not have to go through one translator (best case) or many translators (worst case).

The environment for information in various first responder use cases is different from that of most other computer systems. The first responder environment is very much a mobile environment. Most of the information exchanged by various computers will be carried over wireless connections. While it is easy to assume that interfaces such as LTE and Wi-Fi can provide ubiquitous connectivity at high data rates, there are still large areas of the U.S. that have no such coverage. In disaster situations, wireless communication systems can fail. Satellite data systems have the limits of both significantly smaller available bandwidth and significantly higher costs. Battery-operated sensors such as health sensors or GPS location sensors that a person carries affect how a system is designed. There is a direct correlation between the battery life and the aggregate amount of information provided from those sensors. The quantity of information and frequency of updates provided have a direct impact on battery life. Since first responders must have access to these systems throughout their shifts and cannot depend on the ability to remove, replace, or recharge a piece of hardware, these considerations must all be considered during system design.

The considerations noted above for mobile systems contrast with non-mobile systems where fixed facilities have few power constraints and hardwired connections have minimal data speed limitations. A typical use case for responder information is one where information is received from the field and processed at a fixed location, and then processed output is sent back out to the mobile units in the field. The IoT information for responders under consideration in this document is often different from the much larger data sets in use at dispatch centers and for final incident reports. These differences can be attributed to several factors:

- Bandwidth limitations are imposed by power or cost concerns
- Information sent to a responder is often a subset of the total information available because the information provided is tailored to the responder’s role
- Much of the record management information--such as when information was received and who sent the information--has no value to responders in the field and thus will not be sent

These differences have a direct impact on what the information for the responders looks like. When using the term *looks like*, we are discussing the format of the information. Often, a term from databases is used to describe this format and we refer to this as the *schema* of the data.

The purpose of this document is to develop and distribute a consensus of the format or schema that can be used so responder IoT data is interchangeable and data systems are interoperable. If the various groups working on technology systems for responders fail to accomplish these goals, there will be several problems:

- The cost of systems will increase due to the need for a great deal of software to translate information between all the systems.
- Responders will be frustrated, as much of their time and money will be spent configuring the generic translators which most vendors will provide. Although interoperability is shown as a feature in many systems, after purchasing it is often up to the customer to develop the translation or pay additional money to have it developed. This can cause issues for departments and jurisdictions that do not have appropriately skilled technical resources available.
- Responders will be frustrated in the decision and purchasing processes as they try to navigate interoperability. As lack of resources leaves them with a system that only partially meets their needs, first responders will instead revert to more stable, (if not less automated) known processes. This regression will slow the rate of adoption of technology.
- Innovation will be hampered as groups trying to bring niche products into the marketplace will spend time and money creating interfaces to existing data systems. Piecemeal integration of individual systems will only work in specific cases, limiting the ability for departments to find solutions that work for their situations.

2.2 Other Standards and Data Sets

It is likely that at some point in time a group will be needed to vote on and resolve differences of opinions. Failure to do this would prevent development of standards, and adoption of technology would take 10 to 20 years longer than if there were viable standards. A look at the history of networking standards during the 1970s and 1980s shows what happens when every player in an industry comes out with its own standard. Most notably, disagreement occurred amongst governments and businesses across the U.S. and Europe regarding the communication protocol that would provide the backbone of the internet. Eventually, the TCP/IP protocol that we utilize now became standard, but the effort required to get to that point could have been minimized if earlier action had been taken to identify global requirements.

That said, various standards in areas other than IoT information already exist. In the first responder ecosystem alone, examples of existing standards are listed below:

- OASIS standards for emergency notification (https://www.oasis-open.org/)
- APCO NENA 2.105.1-2017 NG9-1-1 Emergency Incident Data Document (EIDD) for dispatch centers (https://www.apcointl.org/download/apco-nena-2-105-1-2017-ng9-1-1-emergency-incident-data-document-eidd/)
- National Fire Incident Reporting System (NFIRS) incident reporting standard for fire (https://www.usfa.fema.gov/data/nfirs/support/documentation.html)

- NFPA 950: Standard for Data Development and Exchange for the Fire Service (https://www.nfpa.org/)
- NFPA 951: Guide to Building and Utilizing Digital Information (https://www.nfpa.org/)
- EMS based NEMSIS Data Dictionary NHTSA V3.4.0 (https://nemsis.org/)
- Integrated Reporting of Wildland-Fire Information (IRWIN) (https://www.forestsandrangelands.gov/WFIT/applications/IRWIN/index.shtml)

Links are provided on all standards for those seeking to find more in-depth information. However, standards such as OASIS Emergency eXchange Data Language (EXDL) and National Information Exchange Model (NIEM) are two open standards which span the various public safety and emergency management organizations. So, it makes sense to examine sensor data objects in relation to these two standards in more detail. The EXDL creates a well-defined data sharing mechanism for emergency management. Within the specification, there are several elements, but for the general purpose of transmitting sensor data, the EDXL Distribution Element (EDXL-DE) is most applicable. The EDXL-DE specification describes a container for the routing of emergency messages. The details surrounding the information element being exchanged are very clearly outlined, so the information's metadata (such as sender, recipient, urgency, incident and target area) are clear-cut, but the contents of each message are left open. This is beneficial when considering the diverse types of data that are transmitted during an emergency management event. For the research that is being described in this document, the focus is on defining software objects that represent data for sensors. The sensor data object could easily integrate with the EXDL-DE format as a content object, especially considering that as defined, the sensor software object is flexible enough to be formatted with any encoding method needed by the encapsulating standard. Thus, utilizing XML to interact appropriately with the EXDL standard would be no issue.

If we look at standards such as NIEM, they are specifying how to present the information to public safety entities so the information can be interpreted regardless of regional or other specialized terminology. Currently NIEM is expanding its utilization of JSON objects and has successfully integrated geospatial information through the Geo4NIEM initiative in conjunction with the Open Geospatial Consortium. Since NIEM was not developed with the limited transmission or bandwidth requirements of sensors in mind, there may be additional accommodations that would need to be made to consider the sensor use case. However, since public safety sensor systems are still not widely implemented, addressing this topic now allows standards development organizations to have a framework in place so the sensor systems can accommodate public safety needs. By standardizing the format of the sensor data, existing standards will have expectations and understanding of the elements of information available to them, developers will be able to integrate standards-based transmission earlier, and first responders will understand and determine the data that is being sent to them.

The addition of standards being developed specifically for IoT devices is making the public safety IoT landscape more complex. The Open Geospatial Consortium, with their Sensor Things specification (https://www.ogc.org/standards/sensorthings), is one of the most predominant sensor standards available today in regards to standardization of sensor information. However, organizations such as the Institute of Electrical and Electronics Engineers (IEEE) and the International Organization for Standardization (ISO) are undertaking tasks related to standardization of sensors themselves. Although this document focuses on the standardization of the sensor information, like the SensorThings

specification, the scope for this project is limited to information specifically for first responders, and we do not provide a direct methodology for accessing the data, as the Sensor Things specification does. The area of sensor systems for first responders has, for the most part, been set aside, while standards for more commercial systems and general sensor standards have been developed.

One unique exploration of first responder sensor systems is DHS S&T's Next Generation First Responder (NGFR) Integration Handbook (https://www.dhs.gov/science-and-technology/ngfr/handbook), which serves as an approach for commercial sensor technologies to use as they integrate with existing first responder equipment. This guide was developed as part of DHS's NGFR Apex program and is published as a three-part document. The first document, "Part 1: Introduction," gives an overview of the framework and functionality of the design. The second document, "Part 2: Engineering Design," gives specific guidance towards development of solutions within the framework. The final document, "Part 3: Technical Supplement," presents system requirements and additional architecture details for the components of the framework. It is intended to assist commercial entities in developing modules within the framework.

The Integration Handbook is an ideal complement to this project, as it focuses on bridging the current hardware requirements of first responders' personal protective equipment with hardware and communications requirements of forward-looking sensor technologies. The document proposes specific sensor categories that would be of most use to first responders in the field and provides supplemental strategies to allow the sensors to operate efficiently in the field. It uses general specifications, supported by open standards, to allow commercial industries to build modules that support the first responder use case. This document aims to encapsulate the data in such a system.

The key to making these various collections of IoT data work together is to realize that each standard has both elements that are common with other collections of data and elements that are unique to that IoT measurement or information.

One way to make sense of these standards and IoT data exchange schema is to view them as being like pieces of fabric that are stitched together to make a quilt. We see the pieces of information (the data itself) as the fabric of the quilt. The key to putting all these pieces of fabric together to see the big picture (the quilt) is to understand which values in the data objects would be used to connect to other pieces of data. These shared values are the common pieces of information that form the seams where the swatches of fabric are connected.

For instance, think of an individual "swatch" of the quilt as being a piece of the first responder's biometric data (e.g., a heart rate). In order to connect this data to areas outside the first responder's personal equipment (other parts of the quilt), we would need a common piece of data to act as a seam and tie the areas together.

An example of these common pieces of data would be information that identifies a person or responder. All health information ties back to a person. Responder location information ties back to a person. Exposure information tracks back to a person. Exposure information also connects to an incident report. Incident reports have some sort of unique identifier that can connect a call to a dispatch center with the dispatch of units and then connect to a wildland fire report.

Information often involves specific agency identifiers. There is an existing set of numbers for Fire or EMS departments (the FDID). Counties have a FIPS code, dispatch centers have a unique number, we have standard state abbreviations, and the U.S. Post Office has a standardized way to write an address including standardized street abbreviations. Departments often have employee ID numbers to identify people, as well as a *Radio Number* that each person uses. Many of these agency identifiers are numeric; they were created decades ago when computing resources tended to classify things with a number because that took less storage and was easier to process than a text string. The key concept is that when creating an IoT schema, one must be aware of existing classification systems to avoid re-inventing the wheel and creating a second set of classifications. When developing data schemas, it is also important to focus on the pieces of information which are used to connect or link to other groups of information. For that reason, it is important to understand the standards which are related to the information within the schemas. The intersecting pieces of information are the seams of the quilt.

3 PROCESS

The goals of this research project are widespread. The project aims to look at the usage of information and technology across the entire first responder genre. This is a broad group encompassing approximately 2.8 million individuals around the U.S. across fire, law enforcement, and emergency medical services [1]. Interfacing with many of these individuals would be an all-encompassing effort, and due to the technical nature of the material, we wanted to interface specifically with those public safety professionals with an interest and background in mission-critical technology. Therefore, we decided to gather this information through a multi-phased approach.

Phase 1: Fire Service

In the first phase, PSCR contacted the fire service arm of public safety and conducted interviews with members of the fire service, focusing on the most used pieces of information in their day-to-day activities, as well as the technology and how it is currently used. Then, the research team analyzed the outputs from these interviews to determine information trends amongst fire service professionals. These trends were then compared to software objects in existing situational awareness platforms, and software objects were developed based on the example platform.

Phase 2: Continued Fire Service, Law Enforcement and Emergency Medical Services

The second phase began with PSCR reaching out to members of the fire service, law enforcement and emergency medical services (EMS), initially via survey and then via one-on-one interviews, and establishing the most used information in their day-to-day activities, as well as the technology present in their fields and how it is currently used. The responder inputs were then used to develop software objects for law enforcement and EMS, as well as to refine the fire service information.

Phase 3: Determine Commonalities, Refine and Expand

The final phase focused on determining areas that are common to multiple, or all three, first responder entities and areas that are unique to a given discipline. The common objects form a baseline for data that should be transmitted within a first responder IoT product and general characteristics on how that data can be interpreted. The information unique to each discipline was refined based on responder

feedback. The final objects outlined in this document remain an early interpretation of how IoT data can be transmitted amongst public safety-centric devices. Although PSCR had an amazing amount of feedback from participating first responders, continued participation and feedback will be necessary to establish globally functional guidelines for first responders.

To begin the first phase, we posted an open invitation via the 2019 PSCR Stakeholder Meeting, inviting attendees who wished to share their thoughts on technology for first responders to sign up in the meeting application. By utilizing an open method, we ensured that our respondents did not feel coerced into participating in the interview, and we also ensured that we received respondents who had an interest in first responder technology. From this method, we received 13 potential participants.

From the participants that contributed to the first phase of interviews, the first set of elements for the fire service were developed. Beginning in January 2020, PSCR prepared for Phase 2 of the data foundations interviews. Although numerous in-person interviews had been planned originally, our efforts were shifted to 100% virtual communication in March 2020.

A Google Forms-based survey was sent out to the PSCR mailing list, and an additional 72 respondents communicated with us regarding their technology preferences. The following section outlines the content of the interviews along with a brief synopsis of the findings based on participant data.

3.1 Interview Questions

When developing our research, we wanted to examine how first responders were currently doing their jobs. We wanted to know what information they used daily, what tools they used, and how these tools moved them beyond the verbal communication process traditionally used by public safety. Additionally, we wanted to have baseline demographic information available to analyze trends that might appear due to external circumstances such as geographic location or incident type. To that end, the following list of questions was presented to each interviewee. The interviewers asked follow up questions when warranted, but no questions were asked outside of this list.

The full list of questions can be found in Appendix A. The list was approved by the federal Office of Management and Budget and met the requirements of the federal Paperwork Reduction Act and NIST Research Protections Office. The questions covered two sections: the first section, “Basic Demographic Questions,” covered information regarding the first responder’s department, position, environment and technology awareness. The second section, “Setting the Scene,” focused more on the types of responses that first responders encounter in a daily environment and how they utilize technology or want to utilize technology to aid these activities.

3.2 Interviews

The initial 13 participants were contacted individually in order to schedule an interview session, and the interviews took place over a month. At the onset of the interview, in accordance with NIST’s Research Protection Office, each participant was informed of the following:

- Data collected will not be kept in a personally identifiable format
- Records kept will be stored on a secured NIST computer drive
- Attendance and participation in the survey is equivalent to consent to contribute to the research

- At any point in time, the participant may decline to answer a question
- At any point in time, the participant may remove himself or herself from participation
- If participants wish to remove themselves, their previous interview content will be destroyed, and their answers will not be used in research

The research team transcribed the resulting interview recordings, and documented themes amongst the participants answers.

4 RESULTS

The following results account for the major themes extracted from all rounds of first responder interviews. Due to the limited number of interviews performed in this initial research effort, qualitative coding techniques were not deemed necessary. As greater numbers of responses are accumulated in ongoing phases, more traditional techniques will become relevant.

There were three major areas that will be discussed during these results findings, as listed below:

- Technology use
- Indications of future technology implementation
- Informational requirements

These three items all have importance when discussing IoT technology and future states and how technology for first responders will evolve.

4.1 TECHNOLOGY USE

Considering the first responder's current use of technology is vital to determine where the most feasible evolutions in technology will occur. Traditionally, first responder communications have relied heavily on voice communication, receiving transmissions via land mobile radios (LMR) from dispatchers or teammates on locations. Thus, transitions to newer forms of technology, especially now, where there are few dedicated pieces of technology for first responders, can show where benefits lie.

During the interviews, it was apparent that most communications still rely heavily on LMR systems. However, the greatest emerging trend was that of smartphone and tablet utilization while on the job. Many larger departments were able to provide smartphones to full-time employees and utilize tablet-based applications in addition to, or in place of, the traditional vehicular mobile data terminal (MDT). In instances where a department could not provide a smartphone to its employees, for instance in a smaller or volunteer department, several interviewees indicated that they would use personal devices for certain activities. Personal device usage had to be carefully implemented, however, as there were inherent issues with utilization of a non-department device for department activities. These included the following:

- Possible damage to the smartphone while on the job
- Avoiding use of applications that required personally identifiable information
- Possibility of phone needing to be used in evidentiary proceedings

Because of these issues, first responders largely refrained from using personal devices in a work capacity, except for using their devices at time of dispatch to view dispatch information. Their personal devices were then put aside. Within the smartphones and tablets, several categories of applications emerged as most important to first responders for utilization on the job. These applications served to supplement or enhance the information that first responders across categories received from radio and dispatch:

- Navigation applications (e.g., Google Maps, Waze) - These applications served to provide first responders with the quickest and most efficient routes to a scene
- Dispatch information (e.g., Active 911) - These applications would allow firefighters to receive dispatch information plus additional preplan information on their smartphones
- Incident command (e.g., Tactical Application Kit) - Users are enabled to visualize and move designations for apparatus and units
- Hazmat information (e.g., WISER) - These applications could be used to look up chemicals for potential HAZMAT situations, bypassing traditional paper-based information and allowing firefighters to have the information on their smart devices

4.2 Indications of Future Technology Implementation

Larger departments also indicated the importance of access to newer technologies, such as utilization of drone imagery for building preplan information or smartwatch utilization for responder information. However, much of this technology appears to be either in beta testing stages and provided only to a select number of departments, or to be custom built for the department itself. These examples indicate that much of the newer technology is still dependent on a large budgetary expense and/or a large population on which to test the products.

Participating responders were also asked about indicators of future technology evolution in their surroundings, such as the implementation of smart building or smart city features that may aid response in coming years. At this point in time, there were few examples of smart building or smart city applications, particularly those that were accessible to first responders. Many interviewees talked about HVAC controls but noted that during an emergency, first responders are more likely to cut power to an entire building and remove all doubt about whether electricity has been cut than to use automated controls to cut access to one area of a building.

Those first responders who did have buildings with smart features in their areas noted some problems with the technology in its current state. These issues were reported primarily by responders in the fire service, who often need access to building layout and equipment features during a structure fire or similar event. First, the building owner or manager of a smart building is the entity that owns the data generated by this building. For privacy concerns, security concerns and other reasons, this data is often not made available to firefighters in any broad format. The building owner may let firefighters know some of the information when necessary, but that access requires direct contact with the building operators, and the information is generally not accessible. Additionally, with the addition of “smart” elements, traditionally mechanical elements of a building become highly computerized. An example given by one of our interviewees was that of an elevator. In a traditional building, a first responder will have a key for manual access to a malfunctioning elevator. In a smart building, a technician must be called in to override an elevator when it needs to be accessed. This requirement takes a key capability

out of the hands of first responders and puts it into the hands of a third party, adding an element of risk during an emergency scenario.

Ultimately, it is the perceived dependence on an additional piece of equipment that was the most problematic for first responders we interviewed. The additional effort to bring equipment online, maintain it, and then have the confidence that it would work throughout the duration of a mission-critical event was a matter of concern for most first responders. Solutions that are provided for the first responder community will need to take these concerns into consideration, in addition to ensuring that information provided is relevant and usable for the first responder.

4.3 Informational Requirements

After transcribing the interviews, we isolated the interviewees' answers to determine theme words related to first responder data. These theme words were used to determine the most important informational requirements for the first responders. From the first round of interviews, the following topics emerged as data deemed important to a first responder in the field:

- Location (including outdoor location, indoor location, z-axis indoor location, and navigation)
- Incident type/pre-plan data/relevant information
- Personal biometric data
- Weather/external conditions
- Field-specific information (e.g., hydrant and water source logistics, patient health data, etc.)

These topics were consistent across the initial number of first responders targeted, although the type of information that would be included for each category differed depending on the exact capabilities for the department. For example, hydrant and water source logistics could be as simple as locations of all hydrants in a jurisdiction. However, for a department that would need to bring water on site, it could include the location and capacity of available tanker trucks. These crucial differences exemplify the difficulties that arise when attempting to create a solution for first responders across disciplines. Due to the wide range of information that can be encapsulated in one topic, solutions for first responders must be able to capture the vital information, and additional information as necessary, whether it is regional information that may not be applicable to all departments, or proprietary information that allows a developer to set its product apart from others in the market. However, we must also provide a foundational set of information that a first responder can depend on. The following sections will serve to outline key software objects that may be utilized to develop IoT products for first responders.

5 Software Data Objects for First responder IoT Technology

In the 1970s and 1980s, computers were much different from those of today. Everything was slow, small and expensive, relative to modern standards. In that environment, the schema and protocols used to exchange data were very low-level and binary, using a fixed format. For networking, IP protocols were similar. In fact, it is typical for network protocols today still to use these fixed formats to deliver a payload.

One can think of the networking aspects of data exchange as being the *envelope* and the IoT data as being the *content in the envelope*. In general, we will not worry about the envelope because there are

so many ways to send the information. In our IP-centric world, we use UDP and TCP, with other protocols that are built on top of that such as HTTP, email, CoAP and many more. In the IoT environment, we also have wireless networks, such as Bluetooth, Zigbee, Thread and more. The common aspect of all these information delivery systems is there is a payload—the content of our envelope. The goal of this document is to define best practices for the schema of the payload so that regardless of the transmission protocol, the data is understandable. We will look at one such delivery system: the CoAP protocol. Note that a protocol is often a blend of the nature of the envelope, which is often influenced by the underlying delivery system (hardwired, LTE, Bluetooth) and a concept of the payload.

For example, the CoAP protocol is often used for sending IoT data. It is rooted in 32-bit systems. While it can be very space-efficient, it is an example of older packet formats, as illustrated in Fig. 2.

CoAP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
4	32	VER		Type		Token Length				CoAP Request/Response Code								Message ID															
8	64	Token (0 - 8 bytes)																															
12	96																																
16	128	Options (If Available)																															
20	160	1	1	1	1	1	1	1	1	Payload (If Available)																							

Fig. 2: Sample CoAP header

This older format can still have value in systems that are extremely sensitive to either transmission bandwidth or battery consumption. Receiving software would need to know how to interpret the payload, which most likely would have a “hard,” binary-like format, and translate that into a data object.

A key concept here is that in modern systems, the payload is treated as a *stream of bytes*. This approach allows the payload to traverse all the different networks involved when two systems are communicating. In the rest of this report, we will be discussing the IoT schema and only the payload part of system communications.

5.1 The Brave New World of Software Objects

Not only are computers today more powerful in all regards than in the past, but the programming languages used are also more powerful. Often these languages are called *object-oriented* languages. Examples of these languages are Java, JavaScript, PHP and Python. The older traditional languages such as COBOL, Fortran and even C had the concept of variables that held data, as well as arrays of data, which are collections of similar types of data. Data may be collected and organized into a rigid form called a *structure* that is comprised of various types of data. Definitions of structures cannot be changed without recompiling and updating all the software in a system that uses the structure.

Unlike a data structure, a modern software object is like a box filled with various types of information. Each piece of information comes with a "note" attached that defines what it is. The note is called the *index*, a leftover term from the arrays of older systems. So, if the index/note was "Jeff’s Phone Number," and the information (or value) was "919 555 1212," you would know what that information was. The box can contain many different types of information. Integers, floating point numbers and text strings (like the phone number above) are examples of data types. The box can also hold arrays of

values, like the arrays of older languages. The box can also contain other boxes, or more correctly, a software object can contain other objects. Finally, the box can also contain arrays of values.

Inside of an object-oriented language, these objects are stored and represented in a computer. If the program can get information from an object and create other objects, the end user remains unaware of the existence of the objects.

The image in Fig. 3 depicts a representation of a software object, the elements that could be contained within it, and an example of how the data may be represented as an index and value pair.

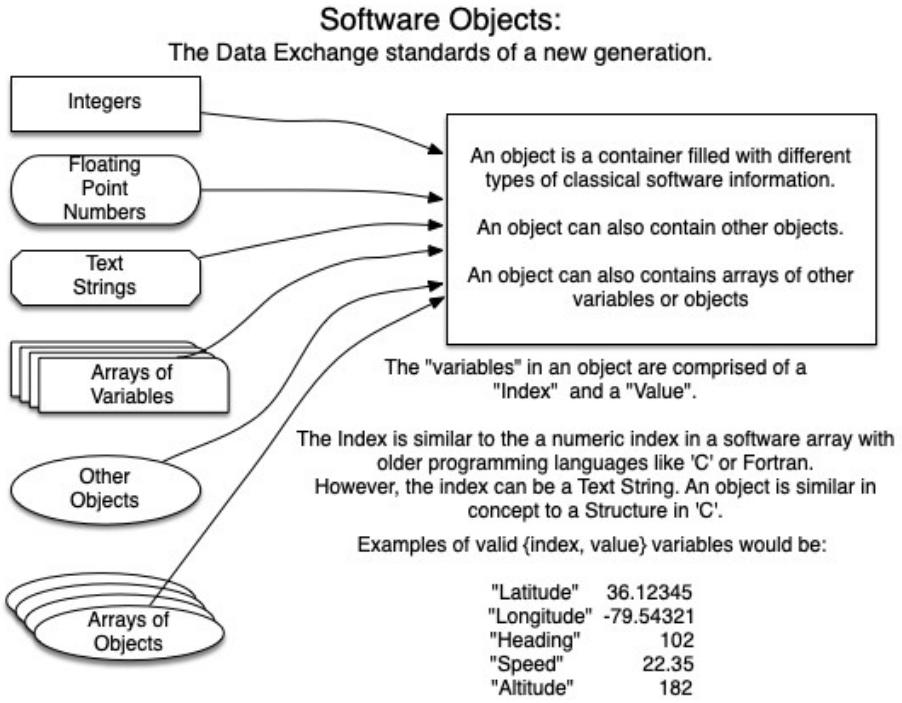

Fig. 3: Software object representation

All the above variables could be put into an object called, for example, “gpsLocation.” A data exchange schema would define the name of the object and what information (e.g., variables) is contained within the object. The text string that is used as the index for a variable is an important part of the schema. The type of the variable (e.g., integer, floating point, text) would be a part of the schema, and when appropriate, the units for the variables would also be a part of the definition. The benefit of object-oriented programming is that software objects are very similar to the structures we use to deal with information in the real world. An object is simply a collection of related information.

5.2 Data Object Development for IoT Systems

As stated above, a data object schema will ideally define the name (or index) of the data elements or variables which comprise the object, the data type that each variable will utilize, and where appropriate, the units that shall be used. In an IoT schema specifically, where we will be looking at data traveling across low bandwidth networks for potentially extended periods of time, some additional considerations should be put into place.

- **Indexes**. A long index name is important for people who work on creating these systems and may need to look at the pieces of information and know what they are. But in the environment of a working system, particularly one with power or bandwidth constraints, a short index is equally important to reduce the amount of data and the power expenditure of an IoT device. Thus, the object definition would ideally have both a long index (for clarity) and a short index (for efficiency).
- **"Required" field**. This field will be specified to inform the user whether an element must be present or is optional. The reason for including this field traces back to the concept that a system using an object's information must "rummage around" and see what it can find. Here we are introduced to one of the best features of a data object: it can have varying amounts of information. This method also provides the ability to add new information to an existing object. As systems evolve, new information and functionality can be added to the existing information, and everything still works, even without a software update. Old systems do not need to consider the new information, and new systems can access the updated information and perform new functionality when applicable. This concept also allows companies to add proprietary features to a system and still adhere to global best practices without fear that their products will become generic.
- **Timestamps.** Almost every time you transmit information in an object, a timestamp will be included. If the information is location information, it is important to know how recently the location has been updated. But even with static information, such as the hydrant location, you will often see a timestamp. Here the timestamp is used to assist in caching and synchronizing information between systems, allowing one system to send another system only the information that has changed since the last time the two systems exchanged information. This approach can result in a reduction of bandwidth that is often on the order of a factor of 50 to 100--a critical factor for potentially limited bandwidth situations such as those experienced by first responders.

  Timestamps also help manage intermittent connectivity issues where the time the data was received cannot be correlated to the acquisition time. Finally, timestamps are essential when creating a final report or log of events that occurred during an emergency. Due to the importance of timestamps in systems for first responders, additional information regarding timestamps and reporting are in Appendix B.

5.3 Data Object Encoding for Communications

Once we have established software objects and a defined schema, the next challenge is turning the objects into a stream of bytes that can be sent over any communications channel, on any system, using any communications protocol. This stream of bytes is called a *string*. It is the encoding of a data object, and this encoding transforms the established software objects into usable information, regardless of the platform utilized.

Currently, there are two dominant encoding techniques: eXtensible Markup Language (XML) and JavaScript Object Notation (JSON). XML is used heavily by Microsoft systems, while JSON is used heavily in web-based systems. Both encoding systems are bi-directional. A software object is converted to a string, and then on the receiving end it is converted from a string back to a software object. All the

object-oriented programing languages have library routines to convert between strings and objects. They also have library routines for both XML and JSON.

Another interesting point is there are library routines for converting from XML to JSON and vice versa. As a rule, all JSON can be converted to XML. However, there are rare cases where XML cannot be converted to JSON, because XML can encode very complex data objects that are virtually never seen in IoT environments. Addressing this limitation, the IoT schema data structures can be constructed so that XML-encoded objects are always simple enough to convert to JSON. As such, either encoding format would be usable.

When defining an IoT schema, it is important to ensure that the schema can be encoded in both XML and JSON so that a developer could use either approach without considering encoding conflicts between the two formats. However, it is not uncommon to have someone insist that XML must be used. From a software point of view, if the schema is properly developed, it will not matter. Further, examining the encoded string and determining which way it was encoded makes the reception process agnostic and automatic. If a developer insists on one encoding protocol or another, it may not be a true requirement.

There is another consideration at the heart of the reason web environments and embedded system environments gravitate toward JSON encoding: JSON encoding takes significantly less space than XML. Fig. 4 illustrates this point with an example showing that the number of characters required to encode an object into JSON is less than that for encoding the same object into XML.

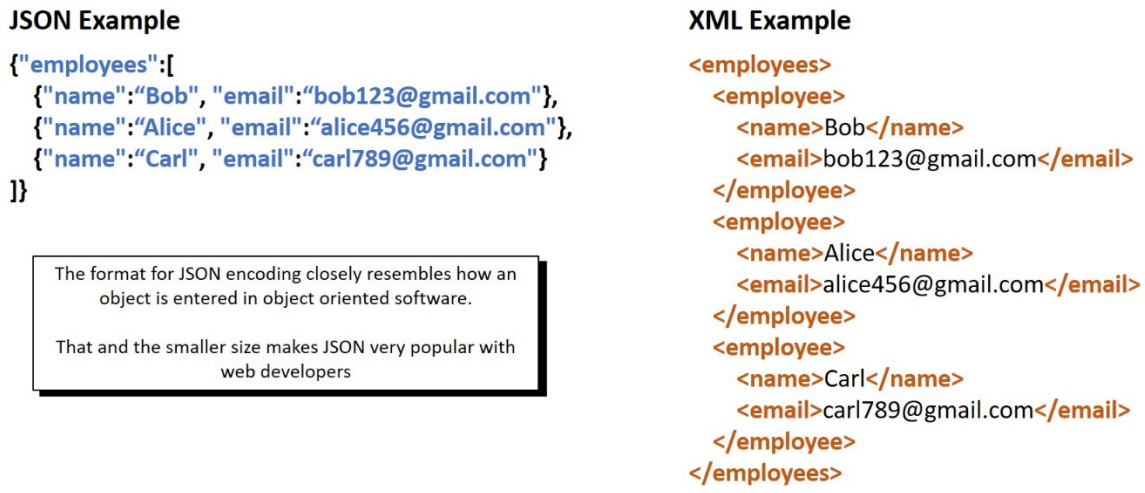

Fig. 4: JSON versus XML encoding

Encapsulating the JSON encoded information required about 192 bytes of memory. The XML encoded information took about 419 bytes. As you can see, the XML format has the index both before and after the value, whereas the JSON format only has it once. Note: the information is an object called "employees" that contains three similar objects containing the name and email of the employee.

Comparing the results using short indexes and JSON versus full indexes and XML reveals a significant difference. Especially for the purposes of IoT-type sensor systems, it makes sense to advocate for a blanket policy requiring JSON encoding in all environments where information could pass on either wireless or low bandwidth communications channels.

5.4 Connectivity and Mapping

When creating IoT exchange schemas, it is important to identify what information is used to connect or relate the schemas to existing standards. This connectivity is analogous to a relational database system. The IoT objects and data objects from other existing standards are the tables in the database. Key fields in each table or indexes in the objects relate that table to other tables to connect the information.

When deciding on index names, care should be taken to determine whether a value is one that will connect the IoT information to information that is external to its object. For instance, timestamps are pervasive in a complex system, and tying information to the time at which it occurs is critical for situational awareness and post-event reporting. Accordingly, the characters 'ts' in a system would ideally have special meaning. In a representative system, the following timestamp (ts) information could exist for a resource (apparatus or person) responding to an incident:

- *Communications ts*: time of last communication from resource
- *Location ts*: time of last reported location of resource
- *Status change ts*: time of status change of resource (e.g., from "En Route" to "On Scene")
- *Update ts*: time of last update given to the resource (for caching management)

Another example is indexes for identifier information. Identifiers often use the characters "id." A system might have a *DepartmentId*, a *PsapId*, a *ResponderId*, an *ApparatusId*, and *IncidentId*, among others.

An effective IoT schema will use the same indexes in different object definitions to help identify the common information that "stitches" all the pieces of the system together. These indexes will also be able to tie into other existing standards with minimum effort. However, if the relationships are documented, the information interaction will remain the same. In many regards, this is a matter of style.

When connecting collections of information such as the values in an object or connecting information in different standards, we run into a new problem: mapping information between two systems. A real-world example of this problem occurs where one system is used for the real-time mapping and response system, and another system is used for long-term incident records, often called a Records Management System or RMS. The real-time mapping system must have a unique identifier associated with each responding resource (person or apparatus). There is also a unique identifier used in the RMS system so incident information gathered during the incident can be associated with a resource. The RMS identifier is used to supply information to the NFIRS RMS system for incident reports that need to have a list of responding apparatus and personnel. Connecting these identifiers in some manner is important. This approach allows the two systems to share data about when apparatus went en route, arrived on scene, and when the scene was cleared. It also shares which responders were on each apparatus.

For people, the federal government solves the problem of a global unique identifier with a Social Security Number, but that cannot be used outside of the government due to privacy concerns. For paid departments an employee number can be used, but this does not work for long-term exposure records. Additionally, the National Fire Operations Reporting System (NFORS) that is used for long-term exposure records has its own unique person identifier that differs from departmental identifiers. Responders with volunteer departments may not have employee IDs at all.

Resource identifiers have an entirely different set of problems. For example, water systems may have a unique identifier for hydrants that may be different from what the fire department uses. EMS ambulances have a number for each physical vehicle, but the vehicles may take on different radio numbers, depending on where they are assigned on any given day. Even with these limited examples, it is obvious that the issues with information mapping are numerous.

When we consider the problem of mapping similar identifiers between different objects or data sets, a fun piece of math comes into play. If you have N different systems that will need to translate their identifiers between them, the number of unique one-to-one pairs of all the identifiers with N systems is:

# of unique pairs of identifiers between N systems = (N x (N-1)) / 2

This equation indicates that the number of pairs of identifiers between systems is on the order of N^2 and demonstrates why it is so critical that there be a consensus standard. Failure to achieve this consensus will result in an exponential data mapping effort that will overwhelm IoT systems, and most of the information technology will simply fail. All in all, developers must avoid overcomplicating things, and managers need to avoid oversimplifying things.

A variant of the mapping problem is the classification problem: determining a common classification language among first responder entities, or even among jurisdictions within the fire service. An example is creating a finite list of incident types, such as "EMS", "fire" or "accident" so that analytics on information, such as a year's worth of emergency calls, can occur. Each dispatch center decides what list of codes describing the nature of an incident (often called "nature codes") will be used. Very seldom do two public safety answering points (PSAPs) agree on either the length of this list or the textual descriptions of the incidents. Given that disparity, classifying an incident is difficult. The same is true for apparatus types, such as engines, tankers, tenders and ambulances. The skills of responders are another area. Often the list of skills or training for a responder matches the state list of courses, but the list differs between states. The classification problem is almost as large as the IoT schema definitions problem.

6 REPRESENTATIVE SOFTWARE OBJECTS

The previous section explained the theory behind the software object model and the reasoning for its use. By developing a best practice for encapsulating the data in first responder systems, we take steps toward making systems that can not only provide data for first responders when needed but can interoperate with the multitude of other situational awareness and reporting systems that exist today and will exist in the future. With that in mind, this section will depict representative IoT software objects for use in first responder IoT solutions.

The documentation presented within the sections “Common Objects,” “Fire Service Objects,” “System Level Objects” and “Server Level Objects” was developed by Peter Hallenbeck, founder of Softwhere Syzygy, LLC, and retired Deputy Chief of Efland Volunteer Fire Department, Efland North Carolina. These objects are the result of both his experience in the fire service and his efforts developing the PageTrack situational awareness platform for first responders. The information within “Fire Service Objects,” “Law Enforcement Objects,” and “EMS Objects” was based on the structure developed by Hallenbeck and outputs received during the interviews held by PSCR during Phases 1 and 2 of this project. The System Level and Server Level sections describe objects that would be of specific use to developers implementing a map-based situational awareness tool.

Again, these objects are meant to be representative objects to form best practices around developing IoT devices for first responders, specifically fire service professionals. The objects do not comprise a standard, and they do not contain any information that may be specific to a proprietary solution. However, as explained above, the benefit from utilizing these schema objects is that additional information can be added without changing the functionality of legacy systems, thereby promoting information exchange without limiting functionality.

Each schema object within the Common, Fire Service, Law Enforcement, and EMS Objects sections contains the following information:

- **Indexes.** These long form indexes provide a descriptive name for the data value and can be used where payload size is not a limiting consideration.

- **Short Form.** The short form name provides a two to four-character representation of the data value name and can be used in instances where the payload size is reduced (e.g., when transmitting data to/from an endpoint IoT device). Commonly used objects will also have a designated initial character (e.g.,” e” to represent an environmental field.

- **Global?** This field indicates whether a data value is Global. If designated as Global, the value can be used outside of the original software object, to allow a developer to use the most common fields in an object without having to transmit the full object. Latitude and Longitude are examples of fields commonly reused outside of the GPS object.

- **Units.** When applicable, units are provided within the schema to provide a baseline understanding of the data being transmitted and stored via this design. The ability to convert to different units for display or storage is available to the developer.

- **Required.** This field indicates whether a field is required within the object, allowing first responders a baseline expectation of the information that will be provided to them. It also allows developers to add proprietary elements to the existing schema without having to make them available to competitors.

- **Data Type.** A data type may be suggested for the indicated value.

- **Sample Val:** When necessary, a sample value is provided, showing the proposed data format.

- **OOBNV.** The field name stands for “Out of Band Null Values,” and the content represents the value transmitted when the device does not receive a valid reading or connectivity is lost. Alternately, if a value is not provided by the transmitter, receiving systems should assume there is no change in value.

Each schema object is shown in a table followed by notes on the individual elements within the object. There is also an “Additional Perspective” section that highlights additional information based on Hallenbeck’s experience in embedded software design and situational awareness tool development.

6.1 Common Objects

6.1.1 GPS Location Schema (object name: *gps*)

Describes the GPS location information and its metadata

Indexes:	Short Form	Global?	Units	Required	Data Type	Sample Value	OOBNV
Latitude	lt	Global	Degrees	☑ Yes	<float>	36.089	999
Longitude	lg	Global	Degrees	☑ Yes	<float>	-79.162123	999
Altitude	lz	Global	MSL	✕ No	<integer>	182	-999
Speed	ls		M/S	✕ No	<float>	22.35	-1
SpeedMph	lm		Miles/Hr	✕ No	<integer>	50	-1
Heading	lh	Global	Degrees	✕ No	<integer>	359	-1
Accuracy	la	Global	Meters	✕ No	<integer>	4	-1
Numsats	lns		Integer	✕ No	<integer>	7	-1
Timestamp	ts		seconds	✕ No	<integer>	1596014568	-2
Source	src			✕ No	<string>	"GPS" or "WiFi"	""

NOTES:

This schema utilizes the following naming convention for the beginning character(s) of the Short Index name:

- 'l' fields related to a GPS location

This schema has more global indices than are typically found in a data object. The global indices allow other schemas (e.g., Hydrant / Water Source) to access the core location information for a static object as part of their definitions instead of using the full GPS data object.

⇒ **Additional Perspective**

The parameters encompassed in the above object are the most commonly used attributes of GPS data. There are many other parameters associated with GPS systems that are not included here. If those parameters are passed in a system, it would be best to have a new object, such as "gpsParams" to encompass those values.

For the **Heading** value, software systems should be tolerant of the value 360 (referring to 360 degrees) and treat it as zero.

6.1.2 Map Annotation Schema (object name: *map*)

Describes an icon to be placed on a map representing a fixed location resource.

Indexes:	Short Form	Global?	Units	Required	Data Type	Sample Value	OOBNV
Latitude	lt	Global	Degrees	☑ Yes	<float>	36.089	999
Longitude	lg	Global	Degrees	☑ Yes	<float>	-79.162123	999
Altitude	lz	Global	MSL	✕ No	<integer>	182	-999
Accuracy	la	Global	Meters	✕ No	<integer>	4	-1
Timestamp	ts		seconds	✕ No	<integer>	1596014568	-2
TimestampCreated	tsc		seconds	✕ No	<integer>	1596014568	-2
Deleted	del		boolean	✕ No	<Boolean>	1 or 0	
Ttl	ttl		seconds	✕ No	<integer>		3600
Url	url			✕ No	<string: url>		"“
TextInfo	txti				<string>		
TextHover	txth				<string>		
TextValid	txtv				<string>		
TypeOfIcon	toi				<string>		
ZoomVisible	zv				<integer:1-20>		
ZoomIndex	zi				<integer:1-10>		

NOTES:

This schema utilizes the following naming conventions for the beginning character(s) of the Short Index name:

- 'l' fields related to a GPS location – uses Global elements from the GPS object

Within the schema, certain indexes may require additional clarification. The following list gives additional information for these elements:

- **Timestamp:** used for cache control of the map annotation object
- **TimestampCreated**: indicates when the symbol was created
- **Deleted**: True if the symbol should be removed from the map; False otherwise
- **TTL:** represents the standard concept of "Time to Live"; indicates how long the map should be annotated. [Note: both the "Deleted" and "TTL" indices can be utilized to control the removal of an icon].
- **Url:** a web URL linking to additional information pertaining to the icon
- **TextInfo:** a string describing the map annotation icon
- **TextHover:** a web concept that provides text information when a mouse is hovering over an icon. If there is no value in the TextHover field, then the value within the TextInfo field is used as hover text.
- **TextValid:** a string used to provide information that controls the visibility of the icon. The schema should define a format for this string, using concepts such as:
  - days of the week
  - start visibility time
  - end visibility time
  - some manner of specifying periodic patterns of days (e.g., shift schedule)

  These values will allow the field to control when icons appear and prevent the first responder's view from becoming overrun with information.
- **Type of Icon:** a string designator for common icon types across first responder departments. See "Additional Perspective" box for more details.
- **ZoomVisible**, **ZoomIndex**: concepts taken from several different map tile (raster scan)-based systems. Both are valuable in controlling visibility at various levels of magnification, such as when several map icons are on top of each other at a specific zoom level but spaced appropriately when the zoom index is increased (the map is zoomed in further).

⇒ **Additional Perspective**

There will eventually need to be agreement on a basic list of icon types to be used within the schema, denoting items such as:

- stations
- schools
- hospitals
- staging areas

and other entities that are common across departments. Achieving consensus amongst different groups will be difficult, but if there is agreement on both a list of icons and the symbology used for the icons, the **TypeOfIcon** index could have a predetermined set of graphic images stored locally on the IoT device, and it would minimize the need for external connectivity to display information.

Additionally, if the filename of a graphic icon (e.g., "hospitalInService.png") could be agreed upon, then in cases of no network connectivity, the server could easily access the file locally. This means that there would be agreement on the name of the symbol, but the image itself would not necessarily have to be standard.

Below is a sample list of basic icon types that could be used to initially populate the schema:

'f'	Fire
'e'	EMS
'l'	Law Enforcement
'r'	rescue
'h'	Hazmat
's'	shelter (civilian)
'e'	Energy (fuel locations)
'l'	Logistics (e.g., staging, base camp, food, sleeping area)
'a'	Aircraft landing zone (e.g., airport, heliport, or drone landing area)
'z'	Hazards
'A'	Always show this icon
'O'	Other (this category acts as an open label for unspecified symbols)

6.1.3 Responder Health Schema (object name: pHealth)

Describes the health and direct environmental information of a responder

Indexes:	Short Form	Global?	Units	Required	Data Type	Sample Value	OOBNV
SkinTemp	ps	Global	Degrees C	✕ No	<float>		-1
CoreTemp	pt	Global	Degrees C	✕ No	<float>		-1
PulseRate	pp	Global	Beats/minute	✕ No	<integer>		-1
PulseOx	po	Global	Percent	✕ No	<integer>		-1
Respiration	pr	Global	Breaths/minute	✕ No	<integer>		-1
BpSystolic	pbs	Global	mmHg	✕ No	<integer>		-1
BpDiastolic	pbd	Global	mmHg	✕ No	<integer>		-1
AmbientTemp	et	Global	Degrees C	✕ No	<float>		-99
AmbientLux	el	Global	lux	✕ No	<integer>		-1
AmbientPressure	ep	Global	kPascal	✕ No	<float>	93.12	-99
AmbientHumidity	eh	Global	Percent	✕ No	<integer>	80	-1
AccelerometerX	pax	Global	.01 G	✕ No	<float>		-9999
AccelerometerY	pay	Global	.01 G	✕ No	<float>		-9999
AccelerometerZ	paz	Global	.01 G	✕ No	<float>		-9999
AccelerometerM	pam	Global	.01 G	✕ No	<float>		-9999
Timestamp	ts		seconds	✕ No	<integer>	1596014568	-2

NOTES:

This schema utilizes the following naming conventions for the beginning character of the Short Index name:

- 'p' fields related to a responder's health
- 'e' fields used for environment telemetry
- 'l' fields related to a GPS location—uses Global elements from the GPS object

This schema makes use of a 4-index grouping for the accelerometer:

- **AccelerometerX**: acceleration along the X axis
- **AccelerometerY**: acceleration along the Y axis
- **AccelerometerZ**: acceleration along the Z axis
- **AccelerometerM**: magnitude of the X, Y, and Z axes.

This structure produces a small increase in the size of the object but makes the data significantly more readable.

⇒ **Additional Perspective**

This schema shows the value of the global index parameter. With these global values, the developer can take environmental measurements and include them in a responder health object to transmit the immediate environmental conditions at the responder's location.

As referenced in the body of the document earlier, there is a significant problem in identifying an individual in a consistent manner. Attributes for a responder could include identifiers such as:

- The department a person is associated with at the time of reporting the health data. For fire and EMS, this may be the FDID.
- The PSAP or regional agency a person is associated with at the time of reporting.
- A universal, lifelong identifier for responder health and exposure information. This may be something along the lines of a 64-bit unique number that is randomly generated the 1st time a person is involved with any agency that tracks or accumulates health and exposure information.

In IoT systems, the data values obtained tend to be sent to a larger server. In such systems, a responder ID is often assigned. As an example, this may be the primary key for the responder table in that server's database. The burden is then on that server to use information about the role that responder has in the incident to provide better and more universal identification.

6.1.4 Environmental Schema (object name: env)

Describes the environmental information of an area surrounding a responder. Includes weather-related information.

Indexes:	Short Form	Global?	Units	Required	Data Type	Sample Value	OOBNV
AmbientTemp	et	Global	Degrees C	✕ No	<float>		-99
AmbientLux	el	Global	lux	✕ No	<integer>		-1
AmbientPressure	ep	Global	kPascal	✕ No	<float>	93.12	-99
AmbientHumidity	eh	Global	Percent	✕ No	<integer>	80	-1
WeatherWind	eww	Global	m/s	✕ No	<integer>		-1
WeatherGust	ewg	Global	m/s	✕ No	<integer>		-1
WeatherDir	ewd	Global	Degrees	✕ No	<integer>		-1
WeatherRainRate	ewr	Global	mm/hr	✕ No	<integer>		-1
WeatherRainLast24	ew2		mm	✕ No	<integer>		-1
WeatherRainMidnight	ewm		mm	✕ No	<integer>		-1
WeatherConditions	ewc			✕ No	<string>		“”
WeatherStationLat	lt	Global	Degrees	✕ No	<float>	36.089	999
WeatherStationLng	lg	Global	Degrees	✕ No	<float>	-79.162123	999
WeatherStationName	ewn			✕ No	<string>		“”

NOTES:

This schema utilizes the following naming conventions for the beginning character(s) of the Short Index name:

- 'l' fields related to a GPS location—uses Global elements from the GPS object
- 'e' fields used for environment telemetry
- 'ew' fields used for weather-related data

The two-character indexes apply to the responder's immediate area and current values.

The National Weather Service (NWS) has many standard string values for weather conditions; they can be passed back to responders wanting weather information using the NWS data object.

Within the schema, certain indexes may require additional clarification. The following list gives additional information for these elements:

- **WeatherStationLat, WeatherStationLng:** the location of the weather station. If there are multiple weather stations in an area, the responders (or the software) can select the one closest to their location. This is another example of the universal indexes discussed in the GPS object; the Global GPS object elements can be used for latitude and longitude in place of a full GPS location object.

- **WeatherStationName**: can be either the National Weather Service abbreviation or a local name. If a department has a weather station at their location that provides local conditions (often on a timelier basis than once an hour), a name that makes sense to the user of the data could be used. Example: "EVFD Sta 1" for Efland Volunteer Fire Department, Station 1.

6.1.5 Vehicle Information Schema (object name: *veh*)

Describes the attributes of a general responder apparatus

Indexes:	Short Form	Global?	Units	Required	Data Type	Sample Value	OOBNV
VehId	vi	Global		☑ Yes	<integer>		
VehHorsepower	vh		HP	✗ No	<integer>		-1
VehRpm	vr		RPM	✗ No	<integer>		-1
VehTempOil	vo		Degrees C	✗ No	<float>	40.2	-1
VehPressOil	vn		kPa	✗ No	<float>	275.2	-1
VehTempCool	vc		Degrees C	✗ No	<integer>	35	-1
VehLoad	vl		percent	✗ No	<integer>		-1
VehFuel	vf		percent	✗ No	<integer>		-1
VehFuelCapacity	vt		gallons	✗ No	<integer>	40	-1
VehFuelConsume	vf		Gallons/Min	✗ No	<float>	0.31	-1
VehGVW	vgvw		pounds	✗ No	<integer>	4200	-1
VehGndClear	vgc		inches	✗ No	<integer>	14	-1
VehWidth	vmw		inches	✗ No	<integer>	92	-1
VehLength	vml		inches	✗ No	<integer>	384	-1

NOTES:

This schema utilizes the following naming convention for the beginning character(s) of the Short Index name:

- 'v' fields related to a general emergency vehicle – no responder category specified

The elements in the Vehicle Information schema can apply to almost any vehicle. It is anticipated that there will be additional specific schemas for fire apparatus, EMS apparatus, rescue apparatus, etc. Those schemas would have additional fields for category-specific information (e.g., water tank size, pump GPM (gallons per minute), SCBAs carried, oxygen carried, etc.).

> ⇒ **Additional Perspective**
>
> Note than some of the information displayed in this object is static, such as fuel tank capacity (**VehFuelCapacity**). However, this static information when combined with the dynamic information of the current fuel level (**VehFuel**) and current consumption rate (**VehFuelConsume**) allows software systems to provide more advanced data, such as the amount of time remaining until the fuel tank is empty. This type of information is universal to all vehicles, and therefore it is included in this general vehicle object.

6.2 *Fire Service Objects*

6.2.1 Hydrant / Water Source Schema (object name: *h2o*)

Indexes:	Short Form	Global?	Units	Required	Data Type	Sample Value	OOBNV
Latitude	lt	Global	Degrees	☑ Yes	<float>	36.089	999
Longitude	lg	Global	Degrees	☑ Yes	<float>	-79.162123	999
Altitude	lz	Global	MSL	✕ No	<integer>	182	-999
Accuracy	la	Global	Meters	✕ No	<integer>	4	-1
WaterType	wt			✕ No	<string>		
GPM	gpm		Gallons/Minute	✕ No	<integer>		-1
NimsStatus	nims	Global		✕ No	<integer: enum>		0
Timestamp	ts		Seconds	✕ No	<integer>	1596014568	-2
TimestampCreated	tsc		Seconds	✕ No	<integer>	1596014568	-2
Url	url			✕ No	<string: url>		“”
TextInfo	txti			✕ No	<string>		
TextHover	txth			✕ No	<string>		
ZoomVisible	zv			✕ No	<integer: 1-20>		
ZoomIndex	zi			✕ No	<integer: 1-10>		

Describes the attributes of hydrant and water source locations

NOTES:

This schema utilizes the following naming convention for the beginning character(s) of the Short Index name:

- 'l' fields related to a GPS location – uses Global elements from the GPS object

Within the schema, certain indexes may require additional clarification. The following list gives additional information for these elements:

- **WaterType:** the type of water source. If there is no value in the field, the default is “hydrant”. Other types of water sources could be “pond”, “draft”, “dry”, and “tank”. (*A finite list of options here would be preferable and would allow for the possibility of using an enumerated value (a number in place of text) for the most common types).*
- **NimsStatus:** an enumerated list of values outlining the key NIMS status categories that are relevant in the field. If there is no value within the field the status is assumed to be 'Available'. The concept of the NIMS Status is described in the “Additional Perspective” box below.
- **Timestamp:** used for cache control of the map annotation object
- **TimestampCreated**: for RMS use and would typically not be passed on to IoT devices.
- **Url:** a web URL linking to additional information pertaining to the icon
- **TextInfo:** a string describing the map annotation icon
- **TextHover:** a web concept that provides text information when a mouse is hovering over an icon. If there is no value in the TextHover field, then the value within the TextInfo field is used as hover text.
- **ZoomVisible**, **ZoomIndex**: concepts taken from several different map tile (raster scan) based systems. Both are valuable in controlling visibility at various levels of magnification, such as when several map icons are on top of each other at a specific zoom level but space appropriately when the zoom index is increased (the map is zoomed in further).

Note that all these defaults and enumerated value lists specified here will greatly reduce the size of the final encoded data object.

⇒ **Additional Perspective**

While map icons could be used for the hydrant and water source elements, it is not uncommon to have thousands of hydrant locations, and this could quickly overwhelm a map view.

Another obstacle for fire service iconology is the incorporation of NFPA standards. One possibility is to have a unique URL of "nfpa" that would indicate that the icon for the hydrant should be a color-coded hydrant based on the GPM/Available Flow. This would also save having to retrieve a URL from the network when in a reduced connectivity environment. Alternately, a null URL could indicate that a generic hydrant symbol should be used.

The NimsStatus field described above would allow departments to easily represent the most used NIMS statuses. The difficulty here is that these elements must be agreed upon. Many systems expand on the basic NIMS concepts of status which are "Available", "Assigned" and "Out-of-Service." These statuses can be used for facilities (e.g., hydrants or hospitals), apparatus, and responders. An example starting point for an expanded nimsStatus list is shown below:

0	Unknown	Out of band null value (OOBNV), for when data cannot be obtained
1	Out of Service	Resource cannot be attached to calls, often associated with mechanical issues or reasons that will "take time" to fix
2	Available	Resource can be attached to calls
3	En Route	Resource is on the way to the scene
4	On Scene	Resource is on scene at the incident location
5	Off Scene	Resource still attached to the call, but not at the incident location. Could be going to a hospital, could be going to get water
6	Unavailable	Resource not out of service, but having equipment cleaned or put back on after a call. Sometimes called the "In Quarters" status.
7	Deploying	Resource driving a long distance to get to a staging area. Often used for disaster assistance or coverage to a mutual aid coverage area. The resource is in transit and not available to be assigned to a call.

Note that water sources, like most fixed location resources, can be either "Available" or "Out of Service".

6.2.2 SCBA Schema (object name: *scba*)

Describes the attributes related to fire service Self Contained Breathing Apparatus (SCBA) units

Indexes:	Short Form	Global?	Units	Required	Data Type	Sample Value	OOBNV
ScbaResId	si			☑ Yes	<integer>		-1
ScbaStart	ss		seconds	✗ No	<integer>	1596014568	-2
ScbaPressMain	sm		kPascal	✗ No	<integer>	13789	-1
ScbaPress1st	s1		kPascal	✗ No	<integer>	1034	-1
ScbaTempAmb	st		Degrees F	✗ No	<integer>	180	-1
ScbaTempMask	sk	Global	Degrees F	✗ No	<integer>	130	-1
ScbaTempRad	su	Global	Degrees F	✗ No	<integer>	400	-1
ScbaLux	sl	Global	Lumens	✗ No	<integer>		-1
ScbaPressAmb	sa	Global	kPascal	✗ No	<integer>	99	-1
ScbaRespirations	sr		Breaths/min	✗ No	<integer>		-1
ScbaToAlarm	sn	Global	seconds	✗ No	<integer>		-1
ScbaToZero	sz	Global	seconds	✗ No	<integer>		-1
ScbaPass	sp	Global	Boolean	✗ No	<integer>		-1
ScbaStill	sg	Global	seconds	✗ No	<integer>		-1

NOTES:

Within the schema, certain indexes may require additional clarification. The following list gives additional information for these elements:

- **ScbaStart:** the time at which air was drawn from the tank such that the pressure went down by 2%. This allows for pressure lost when the tank is turned on.
- **ScbaTempAmb:** the ambient temperature in and around the SCBA. It is a general indicator as to the environment the responder is in.
- **ScbaTempRad**: the temperature being radiated down on the responder.
- **ScbaPass**: a Boolean value of "True" when the tank's PASS alarm is on, "False" otherwise
- **ScbaToAlarm**: the number of estimated number of seconds until the responder's low air alarm will sound. *(This is an advanced value which could exist in systems where the SCBA had information on the tank pressure, the respiration rate, the air consumption history, and the amount of movement base on an accelerometer.)*
- **ScbaToZero**: an estimate as to how many seconds until there is no usable air left for the responder
- **ScbaStill:** the amount of time (in seconds) that the responder has not moved
- **ScbaTempMask:** the temperature inside of the responder's face mask
- **ScbaResiprations**: the respiration rate for the responder.
- **ScbaLux:** the brightness of the first responder's environment

⇒ **Additional Perspective**

When discussing a self-contained breathing apparatus (SCBA), there are two types of measurements being taken:

- Measurements for the SCBA system itself
- Measurements for the responder using the system

The SCBA object uses a combination of these measurement types to display the first responder's direct conditions. The accelerometer values from the earlier Environmental Health schema (Short Forms: *pax*, *pay*, *paz*, *pam*) can be used to judge the current level of physical exertion. Because they are global indexes, they can be easily accessed in this object.

6.2.3 Fire Service Vehicle Schema (object name: *fVeh*)

Describes the attributes distinct to fire service apparatus

Indexes:	Short Form	Global?	Units	Required	Data Type	Sample Value	OOBNV
VehId	vi	Global		☑ Yes	<integer>		
WaterTank	fvwt			✗ No	<boolean>	0 or 1	-1
TankSize	fvts		Gallons	✗ No	<integer>		-1
PumpGPM	fvg		Gallons/Min	✗ No	<float>		-1
LightsEnabled	le			✗ No	<boolean>		-1
SirensEnabled	se			✗ No	<boolean>		-1

NOTES:

This schema utilizes the following naming conventions for the beginning character(s) of the Short Index name:

- 'v' fields related to a general emergency vehicle – no responder category specified
- 'fv' fields related specifically to a fire service vehicle

6.3 *Law Enforcement Objects*

6.3.1 Weapon Discharge (object name: *wDis*)

Describes the location of an officer's weapon being discharged

Indexes:	Short Form	Global?	Units	Required	Data Type	Sample Value	OOBNV
DischargeLat	lt	Global	Degrees	☑ Yes	<float>	36.089	999
DischargeLong	lg	Global	Degrees	☑ Yes	<float>	-79.162123	999
Timestamp	ts		seconds	✗ No	<integer>	1596014568	-2
EquipmentID	eq			✗ No	<string>	Weapon ID	-2
Deleted	del		boolean	✗ No	<Boolean>	1 or 0	
Ttl	ttl		seconds	✗ No	<integer>		3600

NOTES:

This schema utilizes the following naming conventions for the beginning character(s) of the Short Index name:

- 'l' fields related to a GPS location – uses Global elements from the GPS object

Within the schema, certain indexes may require additional clarification. The following list gives additional information for these elements:

- **DischargeLat/DischargeLong**: gives the location of the shot being fired using location elements from GPS object
- **EquipmentID:** the official identifier of the equipment being fired. This may vary based on department. Could also be associated with the individual who has been officially issued the equipment in question.

6.3.2 Law Enforcement Vehicle (object name: lVeh)

Indexes:	Short Form	Global?	Units	Required	Data Type	Sample Value	OOBNV
VehId	vi	Global		☑ Yes	<integer>		
LightsEnabled	vle			✗ No	<boolean>		-1
SirensEnabled	vse			✗ No	<boolean>		-1
VehInPursuit	lvp			✗ No	<boolean>		-1

NOTES:

This schema utilizes the following naming conventions for the beginning character(s) of the Short Index name:

- 'v' fields related to a general emergency vehicle—no responder category specified
- 'lv' fields related specifically to a law enforcement vehicle

This schema assumes that the globally relevant vehicle information will be held in the general vehicle object, while law enforcement specific parameters will be held in the Law Enforcement Vehicle object.

6.4 EMS OBJECTS

6.4.1 Patient Health (object name: ptHealth)

Indexes:	Short Form	Global?	Units	Required	Data Type	Sample Value	OOBNV
PtSkinTemp	pts	Global	Degrees C	✕ No	<float>		-1
PtCoreTemp	ptt	Global	Degrees C	✕ No	<float>		-1
PtPulseRate	ptp	Global	Beats/minute	✕ No	<integer>		-1
PtPulseOx	pto	Global	Percent	✕ No	<integer>		-1
PtRespiration	ptr	Global	Breaths/minute	✕ No	<integer>		-1
PtBpSystolic	ptbs	Global	mmHg	✕ No	<integer>		-1
PtBpDiastolic	ptbd	Global	mmHg	✕ No	<integer>		-1
Timestamp	ts		seconds	✕ No	<integer>	1596014568	-2

NOTES:

This schema utilizes the following naming convention for the beginning character(s) of the Short Index name:

- 'pt' fields related to a patient under transport's vital readings

6.4.3 EMS Vehicle (object name: *eVeh*)

Indexes:	Short Form	Global?	Units	Required	Data Type	Sample Value	OOBNV
VehId	vi	Global		☑ Yes	<integer>		
LightsEnabled	vle			✗ No	<boolean>		-1
SirensEnabled	vse			✗ No	<boolean>		-1
EmsVehType	evt			✗ No	<integer: 1-4>		-1

NOTES:

This schema utilizes the following naming conventions for the beginning character(s) of the Short Index name:

- 'v' fields related to a general emergency vehicle—no responder category specified
- 'ev' fields related specifically to a law enforcement vehicle

This schema assumes that the globally relevant vehicle information will be held in the general vehicle object, while EMS specific parameters will be held in the EMS Vehicle object.

6.5 System Level Objects

6.5.1 Power System Schema (object name: *pwrStatus*)

Describes the status of any source of power, such as a generator, uninterruptible power supply (UPS), or battery-UPS power system

Indexes:	Short Form	Required	Description
Name	n	☑ Yes	Name of the power source.
Type	t	✗ No	Type of power system.. Text string with predefined agreed-upon values (e.g. “generator”, “battery”, “ups”, “solar”, etc.)
VoltsIn	vin	✗ No	Input voltage (only for systems requiring this information)
AmpsIn	iin	✗ No	Input current (only for systems requiring this information)
VoltsOut	vo	✗ No	Output voltage (AC power or battery systems nly)
AmpsOut	io	✗ No	Instantaneous current consumption
AmpsOutMax	io_max	✗ No	Maximum current draw
Watts	po	✗ No	Instantaneous output power (watts)
WattsMax	po_max	✗ No	Maximum power available
Remaining	t	✗ No	Time until power depletion for the current power source (in seconds). -1 indicates unknown time.
Fuel	f	✗ No	Percentage of fuel remaining (for power sources utilizing fuel).
FuelCapacity	fcap	✗ No	Fuel capacity of the power source. Implicitly utilizes the units of the power source (e.g. engine devices: gallons, battery systems: KWh)
Status	vmw	✗ No	“Stoplight status” (green, yellow, “red is dead”)
Faults	vml	✗ No	Text string with errors (e.g. “redundant supply failure”, “lower battery voltage”, “low fuel warning”
Info		✗ No	Short string containing additional information regarding power source
Details		✗ No	Longer text description of the power source
Dept_id		✗ No	Department Identifier of the power source owner
Truck_id		✗ No	Vehicle identifier of the power source owner

NOTES:

This information is presented in a format different to the previous data objects. All values must be presented inside of the Power Status object, and there are no global indexes. It is anticipated that an IoT object like this would be presented at a lower data rate than many of the above objects, such as at a rate of once per minute.

Within the schema, certain indexes may require additional clarification. The following list gives additional information for these elements:

- **Fuel, FuelCapacity:** the units of fields have units appropriate to the type of power system.

> ⇒ **Additional Perspective**
>
> For engine powered generators, there could additionally be Vehicle Information telemetry data with information about the engine itself. This would be information about the generator attached to the engine.

6.5.2 Map Overlay Schema (object name: *mapOverlay*)

Details pertaining to images that can be superimposed on a map.

Indexes:	Short Form	Required	Description
North	n	☑ Yes	Latitude of Northernmost point of map view
South	s	☑ Yes	Latitude of Southernmost point of map view
East	e	☑ Yes	Longitude of Easternmost point of map view
West	w	☑ Yes	Longitude of Westernmost point of map view
Url		☑ Yes	The URL of the map overlay image
Op		✕ No	Default opacity of the overlay (range: 0 to 1)
Info		✕ No	String with information describing the overlay
Address_Object		✕ No	Address of the overlay (building overlays)
Address		✕ No	Short address of overlay (if full address object not specified)
Floor		✕ No	Floor of the structure (building overlays)
Timestamp	ts	✕ No	Most recent modification timestamp for the overlay
EnteredBy	eb	✕ No	Identification of the entity that created the overlay
Ts_Entered		✕ No	The time that this was created
CfsNumber		✕ No	Call for Service (CFS) in the PSAP related to this overlay (if applicable)
PsapId		✕ No	The ID number for the PSAP associated with this overlay
Poly		✕ No	Name of the polygon to be used for adding to this overlay.
Poly_Psap		✕ No	The psapID for the polygon
Poly_DeptId		✕ No	The department ID of the department related to the polygon
Poly_incAuto		✕ No	Boolean value. If true, an incident inside the specified poly or bounds will automatically be displayed on the map.

NOTES:

In the Geographic Information Systems (GIS) environment, there is the concept of a "shape file" which is a collection of polygons or poly lines. A map overlay is an image file (e.g., .png, jpg) that is placed on top of a map. Typically, the shape file has a lot of transparent pixels. It also has opacity, a common feature of most map display systems, and it has metadata.

An example of an overlay is one that was created for the trail system in a state park. The original GIS supplied data was about 1 MB. The overlay that was created out of that was 16 KB, a compression of over a factor of 60 times. The concept of an overlay exists in HTML5, Google Maps and Open Layers.

This information is presented in a format different to the previous data objects because it is not a typical IoT schema. It is presented here as an example of a schema for a server-side data resource.

6.5.3 Map Polygon Schema (object name: *mapPolygon*)

Describes a shape outlining an area of interest on a map

Indexes:	Short Form	Required	Description
Path		☑ Yes	A list of latitude/longitudes creating the outline of the polygon (e.g.: "36.1234,-79.4321|36.5678,-79.9876|…|lat_1,$long_1$|lat_2,$long_2$|…")
Type		☑ Yes	"polygon" or "polyline". Can use 'g' or 'l' as abbreviations
StrokeColor		✗ No	The color of the outline (HTML5 or "#123456" RGB format)
StrokeWeight		✗ No	The thickness (number of pixels) of the outline
StrokeOpacity		✗ No	The opacity of the outline
FillColor		✗ No	The color inside the polygon
FillOpacity		✗ No	The opacity of the color inside the polygon
Meta		✗ No	A string that describes the polygon
Timestamp	ts	✗ No	A timestamp indicating the last time there was a change to the polygon information
EnteredBy	eb	✗ No	Identification of the PSAP/Department/User/etc. that created the object
Tsm		✗ No	A timestamp indicating the last time the object was modified by the user.

Notes:

Polygons are areas enclosed by a set of closed loop line segments. Polylines are a set of line segments that are typically not closed and as such do not have to enclose an area (but they may encompass an area). An example of a polygon might be a fire district or other response district. An example of a polyline might be a line on a map showing the route of a utility line. A boundary for an area, such as the outline of a county, could be represented with either a polygon or a polyline, the key constraint being that it is a closed loop.

HTML5, Google Maps and Open Layers all support polygons and polylines. The name of this object is "polyImage."

This schema is presented differently from the above sections, because it is not a typical IoT schema. It is presented here as an example of a schema for a server-side data resource.

⇒ **Additional Perspective**

Following the principle that software should be liberal in what it accepts, and rigid in what it provides, it is strongly suggested that for closed loops (polygons) any software should be able to deal with two cases of closed loop data: the first case of data where the last point is the same as the first point, creating the closed loop; and the second case of data where the last point is not the same as the first and therefore the first point must be reused as the final point to draw the closed polygon.

Many software systems will have a concept of being able to determine whether a point (latitude/longitude) is inside of a polygon or not. Any software utilizing this object should have the same flexibility.

6.6 Server Level Objects

The last several sections have explored how IoT data for first responders can be managed within unique data objects, taking into account public safety's need to perform disparate jobs (fire, law enforcement, medical) within close proximity to each other and agencies' occasional need to interact with each other. Another aspect of public safety's mission, which makes it unique among many other industries utilizing IoT, is that they must work in remote, disaster-prone, and often network-degraded environments. In that regard, special consideration must be given to the conditions under which first responders are required to transmit data, and how data can continue to be transmitted despite insufficient operating conditions.

Below, the Server Status object allows server status information to be obtained by any computing platform in the system. Typically, endpoint IoT devices with limited battery power will never request it. Personal Area Network devices that receive information from IoT devices and pass the information on to the fixed-location servers will utilize the server status. After an initial load, the devices that have this information can request updates if anything has changed.

Of special note are the bitmasks which allow a server to classify its current operating conditions and specify the level of transmission available to devices transmitting to the server. These bitmasks are:

- **Isolated:** indicates the level of connection or isolation for the system
- **Connected:** indicates level of connectivity on system
- **Bitmask banned:** indicates banned data types (due to bandwidth or similar issues)

This area is currently in development and testing. However, these pieces of information, in conjunction with related server-level data, aim to provide the recipient devices with an understanding of a server's conditions and prevent overconsumption of limited bandwidth by data that may not be mission critical. Further details are presented on the following page.

6.6.1 Server Status Schema (object name: *serverStatus*)

Details the schema of the database table containing server status information. The "Field" element below is equivalent to the Index field in previous images.

Field	Type	Key	Default
server_id	int(11)		1
isolated	int(11)		0
isolated_last	int(11)		0
connected	int(11)		0
connected_last	int(11)		0
stability	smallint(3)		-1
bandwidth_available_world	int(7)		1000
mtu_available_world	smallint(4)		127
bandwidth_minimum_local	int(7)		1000
mtu_available_local	smallint(4)		127
bitmask_banned	int(11)		0
time_til_down	int(11)		-1
time_booted	int(11)		0
latency_ms_world	int(11)		10
timestamp	int(11)		2
timestamp_last_rt_stream	int(11)		2
ipV4	int(11) unsigned		0
ipV4_netmask	int(11) unsigned		65535
ipV6hi	bigint(20) unsigned		0
ipV6lo	bigint(20) unsigned		0
lattitude	float		0
longitude	float		0
name	varchar(63)		NOT SET
gateway_source_name	varchar(31)		NOT KNOWN
gateway_type	varchar(15)		NOT KNOWN
google_client	varchar(31)		
google_version	varchar(15)		quarterly
google_crypto	varchar(47)		
local_map_root	varchar(31)		

Definitions for '`server_status`' table:

Notes:"timestamp" updated on any change, used for cache coherency control.

isolated bitmask:
Indicates level of connection / isolation for system

0 System has full LTE Internet connectivity
1 System is isolated and has no internet access
2 System has Iridium connection
4 System has connection to local isolated server that has no Internet connection (standalone server)

connected bitmask:
Indicates level of connectivity on system
Note: any bit set means there is some level of connectivity

0 System has full bandwidth connection (**Default**).
1 System has full bandwidth connection to Internet (LTE connection)
2 System has 88Kbps-ish connection to Internet (Iridium Direct Internet)
4 System has Iridium SDB messaging (1960 byte max payload size)
8 Reserved
0x10 System has mesh network connectivity, about 50Kbps.
0x20 System has mesh network connectivity, about 10Kbps.

bitmask_banned:
Indicates banned data types (due to bandwidth or similar issues)
Note: "Map Tiles" implies images < 40 KB

1 Map Tiles from local server
2 Map Tiles from the Internet
4 Audio
8 Images over 5 KB (allows buttons)
0x10 Images over 100 KB
0x20 Polygons/Polylines
0x40 Overlays over 100 KB
0x80 Video

Notes

Timestamp: Indicates the last time an element was changed and is used for cache control.

Within the schema, certain indexes may require additional clarification. The following list gives additional information for these elements:

- **Bandwidth Avail World, Connected:** affect how often IoT data is sent and how often temporally stored IoT data is flushed back "upstream" in situations where connectivity is intermittent

⇒ **Additional Perspective**

In a fixed location environment, the server status information can be stable for many hours on end. In the deployed scenario where connectivity is more sporadic, the server status may change every few minutes.

Because the server information is not typically transmitted to the lower power, lower bandwidth IoT devices, and because the status information will have a lower rate of change than typical IoT data, the lack of short indexes is not a major concern.

In the web environment, the system status values are populated into variables on page load. As the web page interacts with the server, values that change are provided automatically along with other information.

From a usage point of view, software on systems will have a data object that looks exactly like the above table. Any information that comes over the communications link to that device updates whatever values are provided in the software systems data object.

7 CONCLUSIONS

Throughout this paper we have touched on many different areas, ranging from viewpoints directly from all categories of public safety professionals to the need for, and usage of, software objects in first responder technology. Ultimately, all the topics relate to either the need for consensus standards or guidance on how to create them. There are many nuances that are beyond the scope of the paper, and although they were not included in this document, their importance remains great. These elements will be understood by the people involved in the consensus process.

Every journey begins with the first step (or sometimes a stumble). A useful first step here is to define several very basic, common IoT objects. Many of these basic objects, such as location, will be part of larger more complex objects. By defining these basic "building block" objects now, we increase the odds of having some degree of interoperability in the future. These objects also begin to define a style and best practices for index names, preferential units, etc. that will guide developers in creating more complex IoT objects. What must not be overlooked is the need for everyone to work together. Public safety must be able to articulate their needs, and system developers must be able to listen and interact with public safety to implement the right information. Public safety and the technology industry are at a point where there is a chance to control some level of how information is conveyed. There are many other challenges that will be faced before IoT technology may be used widely, but that does not mean that either entity should not take the time now to ensure that baseline information is well defined. The more time that passes without a consensus definition, the more likely that the implementations of individual companies will never utilize a common specification.

As stated initially, this document is intended to be the start of a conversation, not the end of it. Ultimately, we intend for this document to be the start of a much more robust set of information, and hopefully this will lead to a faster, more effective technology implementation for the first responder community. With technology implemented that truly meets the needs of first responders, we can enhance their communications, their situational awareness and their safety.

8 APPENDIX A: INTERVIEW QUESTIONS USED DURING PHASE 1

Section I: Basic demographic questions

1. What is your jurisdiction? (City, County, State, Federal)
2. What is your role in your department?
3. Where do you work? (geographic location)
4. What type of geographic environment do you work in?
5. What types of data do you work with?
6. Does your department / jurisdiction utilize software-based record management services?
7. How familiar are you with data technologies for work purposes?
8. Does your department / jurisdiction utilize software-based record management services?

Section II: Setting the Scene

1. Describe the most common working environment. When you think of a standard day, what are you doing?
2. Think about an emergency where you did not feel you had the adequate information to do your job. This includes high-risk situations in which many people were inputting data, and in which the data could have a greater impact on emergency response outcomes. Describe the situation.
3. What communications challenges do first responders experience when using IoT devices?
4. Which IoT devices commonly used in civilian life, such as a smartwatch or a smartphone, do you use in the field?
5. What type of data collection and storage could be offloaded from the first responders and put onto devices?
6. What information about a building do you have or receive en route?
7. What information do you gather upon arrival at a building?
8. Do you require information from a building manager/landlord during pre-planning or during an event itself?
9. Do any of the buildings in your jurisdiction utilize “smart” elements?
10. Do you have any other thoughts about IoT technologies in your work to share?

9 APPENDIX B: FURTHER THOUGHTS ON TIMESTAMPS

Almost every time you have information in an object, you will have a timestamp. If the information is location information, it is important to know how "fresh" the location is. But even with static information, such as a hydrant location, you will often see a timestamp. Here the timestamp is used to assist in caching and synchronizing information between systems. This timestamp enables one system to send another system only the information that has changed since the last time the two systems talked, resulting in a reduction of bandwidth that is often on the order of a factor of 50 to 100. Why send all the information about a system when you can just send what has changed? Timestamps also help deal with intermittent connectivity issues where you cannot use the time the data was received to know when the data was acquired. They are critical when creating final reports or a log of events that occurred during an emergency operation.

There is no shortage of examples on how to "print" a time; i.e., to create a text string that represents a time. Note that all timestamps have the ultimate goal of producing a "date and time." We see in the world today that a date can be "Month/Day/Year" (U.S. format) or "Day/Month/Year" (military and most other parts of the world). There are also the issues of what time zone the user was in and whether Daylight Savings Time (DST) is observed or not, and times may be in 12-hour or 24-hour format.

A common standard for printing the time is ISO 8601. There have been many other ISO standards, such as 2014, 2015, 2711, 3307, and 4031. There have also been revisions on 8601. The good thing about standards is there are so many from which to choose.

On many computer systems a timestamp first seen on the Unix and Linux operating systems called the "Unix Timestamp" is used. It is the number of seconds since midnight on January 1, 1970 in the UTC time zone (also called GMT for historical reasons). As you can see in the example, a typical Unix timestamp today looks like "1596014568."

There are two "wins" with the Unix timestamp. The first is that it makes computations about time very easy because the timestamp increases monolithically with time. The second is that, for the purpose of our schema, it takes far fewer characters than the ISO 8601 standard. The downside is that if you want to convert it back to local time, you have to know the time zone and DST information associated with that timestamp.

Here we see another difference between IoT data and fixed computing system information. It is typical for a mobile device to provide only the Unix timestamp, and then when the information arrives at a larger hosted server at some fixed location, that server needs to be sure that the time zone and DTS observance information is saved in some manner where the big system can export data in a full ISO 86501 format. This practice lets small IoT devices deal with time in a manner that is very easy and efficient in terms of power and bandwidth.

10 REFERENCES

[1] *"What do first responders do to preserve our freedom?"*. American Security Council Foundation. Accessed on June 9, 2020. [Online]. Available: https://www.americansecuritycouncilfoundation.org/first-responders/.