Internet of Things Data Foundations for First Responders Fire Service

FY2020

May 6, 2020

Science and Technology Directorate

Homeland Security

Science and Technology

PSCR

Internet of Things Data Foundations for First Responders
Fire Service

FY2020

May 6, 2020

Work performed in support of the:
Department of Homeland Security
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 CONTENTS 3
TABLE OF FIGURES 4
1 EXECUTIVE SUMMARY 5
2 INTRODUCTION 5
2. 1 CHALLENGES IN A FIRST RESPONDER ENVIRONMENT 7
2.2 OTHER STANDARDS AND DATA SETS 9
3 PROCESS 11
3.1 INTERVIEW QUESTIONS 12
3.2 INTERVIEWS 13
4 RESULTS 13
4.1 TECHNOLOGY USE 13
4.2 INDICATIONS OF FUTURE TECHNOLOGY IMPLEMENTATION 14
4.3 INFORMATIONAL REQUIREMENTS 15
SOFTWARE DATA OBJECTS FOR FIRST RESPONDER IOT TECHNOLOGY 16
THE BRAVE NEW WORLD OF SOFTWARE OBJECTS 17
DATA OBJECT DEVELOPMENT FOR IOT SYSTEMS 18
DATA OBJECT ENCODING FOR COMMUNICATIONS 19
CONNECTIVITY AND MAPPING 21
REPRESENTATIVE SOFTWARE OBJECTS 23
FIRE SERVICE OBJECTS 24
GPS Location Schema (object name: *gps*) 24
Map Annotation Schema (object name: *map*) 26
Hydrant / Water Source Schema (object name: *h2o*) 29
Responder Health Schema (object name: *pHealth*) 32
Environmental Schema (object name: *env*) 34
SCBA Schema (object name: *scba*) 36
Vehicle Information Schema (object name: *veh*) 38
SYSTEM OBJECTS 40
Server Status Schema (object name: *serverStatus*) 40

Power System Schema (object name: *pwrStatus*) 42
Map Overlay Schema (object name: *mapOverlay*) 44
Map Polygon Schema (object name: *mapPolygon*) 46
CONCLUSIONS 48
APPENDIX A: INTERVIEW QUESTIONS USED DURING PHASE 1 49
APPENDIX B: FURTHER THOUGHTS ON TIMESTAMPS 50
REFERENCES 51

TABLE OF FIGURES

Figure 1: IoT Data Flow in a proprietary system 6
Figure 2: Sample CoAP Header 17
Figure 3: Software Object Representation 18
Figure 4: JSON versus XML encoding 20

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 Group (PSCR), in research sponsored by the Department of Homeland Security (DHS) Science and Technology (S&T) Directorate, has been studying IoT devices for first responders to determine how these devices can be utilized 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 fire safety professionals, 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

Many companies currently have a presence in the increasingly lucrative IoT industry. Because 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.

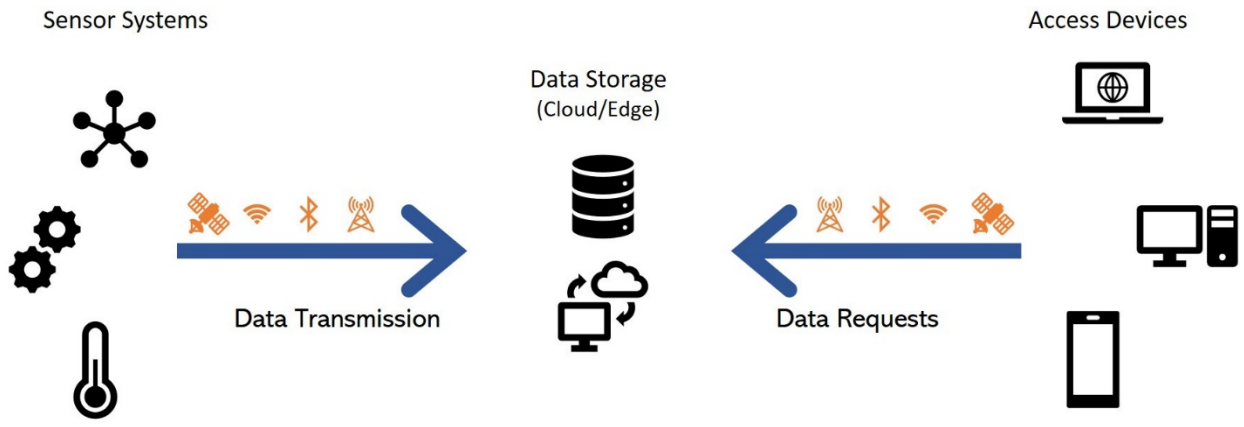

Fig. 1: IoT data flow in a proprietary system

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 wishing 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.

On the other hand, when we begin discussing commercial usage of IoT systems, such as within the Industrial IoT (IIoT) environment, the IIoT definition pertains to an industry or market. According to Hewlett Packard Enterprise (HPE), "While the word 'industrial' may call to mind warehouses, shipyards and factory floors, IIoT technologies hold a lot of promise for a diverse range of industries, including agriculture, healthcare, financial services, retail, and advertising" [1]. It is important to remember that these industries are defined by similar basic operations. Although individual businesses may differ, overall operations converge within an industry, and—most importantly—the businesses themselves will be the stewards of their own data. This approach gives businesses operating in the IIoT sphere a level of flexibility to acquire and control their own data, and larger businesses could additionally create custom platforms for their own environments.

From one perspective, public safety can be seen as 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 sort of public safety governance and building an IoT product to meet the needs of jurisdictions across the U.S. is a daunting prospect. But as with the industries mentioned above, their differences are transcended by a kernel of similarity. The aim of this project is to ascertain that similarity through discussion with public safety officials, and document objects that could be of use to public safety in future sensor systems.

2. 1 Challenges in a First responder Environment

When looking at 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 firefighting 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 as first responders, they 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 fact results in individual departments prioritizing different requirements and gravitating towards diverse solutions equipment to meet their needs.

Because of the wide range of activities undertaken by a first responder, and the relatively low market share that the first responder community provides for these IoT devices, there have been few devices developed for this unique environment. The consequential dearth of devices makes public safety a great potential market for expansion. However, there are additional requirements that must be met for an IoT device to be used successfully in the public safety environment. 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 the first responder's 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 that 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.

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." An analogy would be 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. But if they spoke the same language, they would be far more productive. The cost of communicating would go down because the savings produced by eliminating the translators would be very large. 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 United States 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 have a direct impact on battery life. Because 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 that is 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 that 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 could be several problems:

- The cost of systems will go up 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, upon 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 a lot of time and money dealing with 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 and formed 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 good 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.

That said, various standards in areas other than IoT information do and will 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)

Making the public safety IoT landscape more complex is the addition of standards being developed specifically for IoT devices. The Open Geospatial Consortium, with their Sensor Things specification (https://www.ogc.org/standards/sensorthings), is one of the most predominant when discussing 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 all this data work together is to realize that these existing sets of data have both elements that are common and elements that are unique to the field to which each standard belongs.

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.

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 [2]. 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

The first phase consisted of PSCR reaching out to the fire service arm of public safety and conducting 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. We then 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 will begin 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, focusing on 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. Responder input will be 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 will focus 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 (if present) will form a baseline for data that should be contained within a first responder IoT product. The unique information can then be refined with more focused responder feedback from each discipline.

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 thirteen potential participants.

3.1 Interview Questions

When developing our research, we wanted to look at 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. Follow up questions were asked when the question 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 the following 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 resulting interview recordings were transcribed, and themes were documented amongst the participants answers.

4 RESULTS

The following results account for the major themes extracted from the initial round of firefighter interviews. Because of the limited number of interviews performed in the initial phase, 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 in order 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 a number of smartphones to full-time employees and utilize tablet-based applications in addition to, or in place of, the traditional fire apparatus 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, firefighters 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 firefighters for utilization on the job. These applications served to supplement or enhance the information that firefighters 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 firefighter 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.

Firefighters 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. 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 they provide information that 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 and navigation)
- Incident type/pre-plan data/relevant information
- Hydrant and water source logistics
- Personal data
- Weather/external conditions

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. Because of 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 says 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 shows 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.

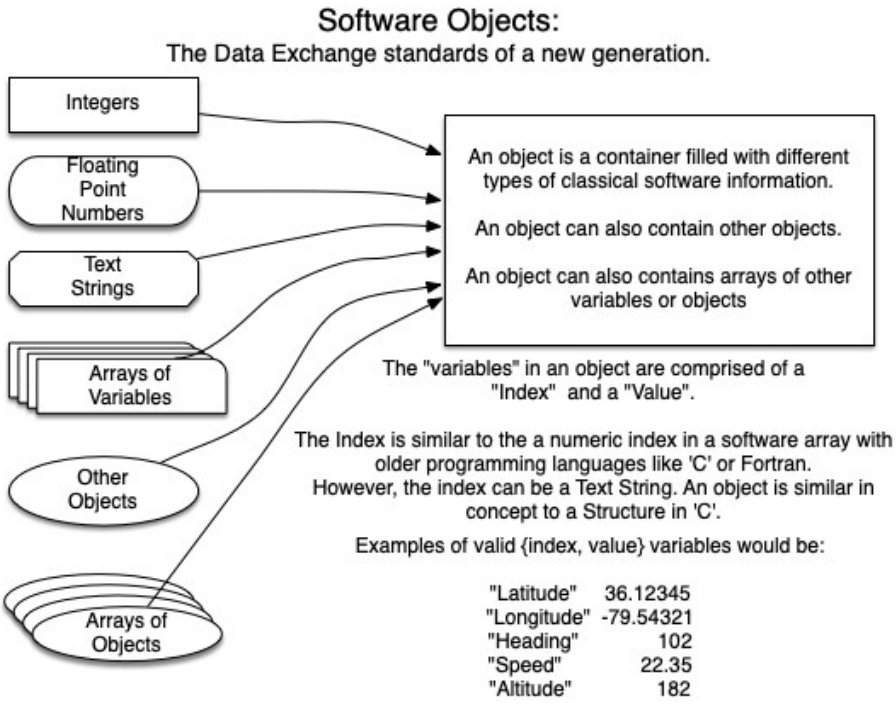

Fig. 3: Software object representation

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 deal with intermittent connectivity issues where 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 operation. Thus, this field should be present on most, if not all, data objects for IoT systems. Because of 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 that 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, due to the fact that 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.

JSON Example

```
{"employees":[
   {"name":"Bob", "email":"bob123@gmail.com"},
   {"name":"Alice", "email":"alice456@gmail.com"},
   {"name":"Carl", "email":"carl789@gmail.com"}
]}
```

The format for JSON encoding closely resembles how an object is entered in object oriented software.

That and the smaller size makes JSON very popular with web developers

XML Example

```
<employees>
  <employee>
    <name>Bob</name>
    <email>bob123@gmail.com</email>
  </employee>
  <employee>
    <name>Alice</name>
    <email>alice456@gmail.com</email>
  </employee>
  <employee>
    <name>Carl</name>
    <email>carl789@gmail.com</email>
  </employee>
</employees>
```

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 that 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.

A good 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 that 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 which 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 look at 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 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 big 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 usage. 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 do and will exist. With that in mind, this section will depict representative IoT software objects for use in first responder IoT solutions.

The documentation presented below 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 objects model implementations of the key objects outlined by our first responder interviewees, as well as additional objects implemented by PageTrack for use by professionals in the fire service. The second section lays out objects that would be of specific use to developers implementing a map-based situational awareness tool. Each schema object is shown as a table followed by notes on the individual elements within the object.

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.

6.1 Fire Service Objects

GPS Location Schema (object name: *gps*)

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

NOTES:

This schema has more global indices than are typically found in a data object. These additional indices allow other schemas, such as Hydrant / Water Source, to access the core location information for a static object as part of their definition, instead of having to use the full GPS data object. This schema also utilizes the convention that short indices beginning with the character 'l' (lowercase L) be reserved for location-based information.

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

For out of band null values (as indicated by the “OOBNV” column):

- -999 meters (-3,277 feet) works because there is no GPS receiver that could be at this low distance and still receive the satellite signal.
- "" denotes a null string. In this case, a source without a name means no source is known.
- As a reminder, OOBNV values are used to provide a hint to the receiver that the device can provide this type of information but currently has no valid value. Alternately, if a value is not provided by the transmitter, receiving systems should assume there is no change in the value.

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

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

NOTES:

This schema is for an icon to be placed on a map representing a fixed location resource.

The 'ts' or Timestamp index is for cache control.

The 'tsc' or TimestampCreated index indicates when the symbol was created.

The Deleted index is true if the symbol should be removed from the map.

The standard concept of Time to Live is represented by 'ttl' and says how long the map should be annotated. Note that the Deleted and ttl indices can both be utilized to control the removal of an icon.

The TextValid index is a string that can be used to provide information that controls the visibility of the icon. The IoT schema should define a format for this string, including 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 TextValid field to control when icons appear and prevent the first responder's view from becoming overrun with unnecessary information.

Ideally, there will 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 icon, the TypeOfIcon index could have a predetermined set of graphic images would be stored locally on the IoT device, and it would minimize the need for external connectivity to display information.

Additionally, if the filename of the graphic icon, such as "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)

The zoomVisible and zoomIndex indices are concepts taken from several different map tile (raster scan) based map systems. These are valuable in controlling map clutter at various levels of zoom, such as when several map icons are on top of each other at a specific zoom level.

If there is no value in the textHover field, then the hover text (a web concept that provides text information when a mouse is hovering over an icon) is the same as the value within the textInfo field.

6.1. 3 Hydrant / Water Source Schema (object name: *h2o*)

Indexes:	Short Form	Global?	Units	Required	Format	OOBNV
latitude	lt	Global	Degrees	✓ Yes	36.089	999
longitude	lg	Global	Degrees	✓ Yes	-79.162123	999
altitude	lz	Global	MSL	✗ No	182	-999
accuracy	la	Global	Meters	✗ No	4	-1
waterType	wt			✗ No	<string>	
gpm	gpm		Gallons/minute	✗ No	integer	-1
nimsStatus	nims	Global			integer	0
timestamp	ts		seconds	✗ No	1596014568	-2
timestampCreated	tsc		seconds	✗ No	1596014568	-2
url	url		Web Url	✗ No		""
textInfo	txti				<string>	
textHover	txth				<string>	
zoomVisible	zv				Integer, 1 to 20	
zoomIndex	zi				Integer, 1 to 10	

NOTES:

If there is no value in the textHover field, then the hover text (a web concept that provides text information when a mouse is hovering over an icon) is the same as the value within the textInfo field.

If there is no value within the nimsStatus field the status is assumed to be 'available'. The concept of the NIMS Status is described below.

The 'ts' or Timestamp index is for cache control.

The TimestampCreated field is for RMS use and would typically not be passed on to IoT devices.

Another obstacle for fire service iconology is determining how to incorporate 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 net. Alternately, a null url would mean use a generic hydrant symbol.

While map icons could be used here, it is not uncommon to have thousands of hydrant locations. We also introduce a new concept here: for some schema, if a value is missing there is a default value. For instance, if there is no waterType value, the default is a hydrant. Other types of water sources would be pond, draft, dry, and tank. A finite list of options here would be nice, and there is the possibility of using a number instead of text for the most common types.

Another new but common concept is introduced here: for the nimsStatus field we have an enumerated list of values that must be agreed on. 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 (like hydrants or hospitals), apparatus, and responders. A starting point for an expanded nimsStatus list is:

0	Unknown	Our 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.

A default value for the zoomIndex field could be up to the client-side system; a value of 3 would be a good choice.

A default value for the zoomVisible field could be 15. When zoomed out further (a lower zoom number), municipal hydrants will tend to be displayed on top of each other. The actual value is implemented by the software that displays the map to the user. Depending on the screen size and resolution, different default values may make sense.

Note that all these defaults and enumerated value lists serve to greatly reduce the size of the final encoded data object.

6.1.4 Responder Health Schema (object name: *pHealth*)

Indexes:	Short Form	Global?	Units	Required	Format	OOBNV
skinTemp	ps	Global	Degrees C	✕ No		-1
coreTemp	pt	Global	Degrees C	✕ No		-1
pulseRate	pp	Global	Beats/minute	✕ No		--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		-99
ambientLux	el	Global	lux	✕ No	integer	-1
ambientPressure	ep	Global	kPascal	✕ No	93.12	-99
ambientHumidity	eh	Global	Percent	✕ No	80	-1
accelerometerX	pax	Global	.01 G	✕ No		-9999
accelerometerY	pay	Global	.01 G	✕ No		-9999
accelerometerZ	paz	Global	.01 G	✕ No		-9999
accelerometerM	pam	Global	.01 G	✕ No		-9999
timestamp	ts		seconds	✕ No	1596014568	-2

NOTES:

This schema shows the value of the global indexes. We can take environmental measurements and include them in responder health to pass back the immediate environmental conditions at the responder's location.

Note the convention of a short index beginning in 'p' to indicate a field related to a responder's health. A short index beginning in 'e' will be used for environment telemetry. As before, a short index beginning in "l' indicates location information. This schema also makes use of a 3-character index for the accelerometer (fields accelerometerX, accelerometerY, and accelerometerZ). This produces a small increase in the size of the object, but makes the data significantly more readable.

Following these conventions will allow for many 2-character short indexes which should help in converting information from one schema to the next. It will also help when a few measured values need to be passed on to other areas and the overhead of placing them inside of an object can be avoided.

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:

1) The department a person is associated with at the time of reporting the health data. For fire and EMS, this may be the FDID
2) The PSAP or regional agency a person is associated with at the time of reporting.
3) A universal "stays with you for life" 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 sever. As such, a responder ID is often assigned. As an example, that 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.

6.1.5 Environmental Schema (object name: *env*)

Indexes:	Short Form	Global?	Units	Required	Format	OOBNV
ambientTemp	et	Global	Degrees C	✕ No		-99
ambientLux	el	Global	lux	✕ No	integer	-1
ambientPressure	ep	Global	kPascal	✕ No	93.12	-99
ambientHumidity	eh	Global	Percent	✕ No	80	-1
weatherWind	eww	Global	m/s	✕ No		
weatherGust	ewg	Global	m/s	✕ No		
weatherDir	ewd	Global	Degrees	✕ No	integer	-1
weatherRainRate	ewr	Global	mm/hr	✕ No		-1
weatherRainLast24	ew2		mm	✕ No		-1
weatherRainMidnight	ewm		mm	✕ No		-1
weatherConditions	ewc		<string>	✕ No		
weatherStationLat	lt	Global	Degrees	✕ No	36.089	
weatherStationLng	lg	Global	Degrees	✕ No	-79.162123	
weatherStationName	ewn			✕ No	string	“”

NOTES:

There is an astounding number of measurements for weather, and even more when you add forecast information. The best way to think of these values would be "What information can I get from a weather station, along with a bit of history on rainfall amounts?" These are values that responders can relate to and that they use to help them respond appropriately to a call. Objects for weather predictions, watches and warnings need to be established. They should coordinate with the National Weather Service's classifications and the OASIS data dictionary.

Note the use of the 'ew' prefix for clearly weather-related data in the short index, while the two-character indexes can 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.

The location of the weather station is also provided. If there are multiple weather stations in an area, the responders (or the software) can select the one closest to their location. Here we also see the use of the universal indexes discussed earlier; we didn't have to include a GPS location object to obtain the weather station latitude and longitude values.

The weatherStationName name field can be either the National Weather Service abbreviation or a local name. If a fire 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 can be used. Example: "EVFD Sta 1" for Efland Volunteer Fire Department, Station 1.

There is a relatively recent push to identify and track hazardous exposures that responders encounter, which could include hazardous weather incidents. The first letter of a two-character index for exposures should be reserved. That letter could be "x" to indicate e**x**posure.

6.1. 6 SCBA Schema (object name: *scba*)

Indexes:	Short Form	Global?	Units	Required	Format	OOBNV
scbaResId	si			✓ Yes	integer	-1
scbaStart	ss		timestamp	✗ No	timestamp	-2
scbaPressMain	sm		kPascal	✗ No	13789	-1
scbaPress1st	s1		kPascal	✗ No	1034	-1
scbaTempAmb	st		Degrees F	✗ No	180	-1
scbaTempMask	sk	Global	Degrees F	✗ No	130	-1
scbaTempRad	su	Global	Degrees F	✗ No	400	-1
scbaLux	sl	Global	Lumens	✗ No	integer	-1
scbaPressAmb	sa	Global	kPascal	✗ No	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:

When talking about a self-contained breathing apparatus (SCBA), there are two types of measurements being taken. One type is for the SCBA system itself, and the other measurements are for the responder using the system.

The scbaStart value is the time at which air was being drawn from the tank such that the pressure went down by 2%. This allows for pressure lost when the tank is turned on.

The scbaTempAmb value is the ambient temperature in and around the SCBA. It is a general indicator as to the environment the responder is in.

The scbaTempRad value is the temperature being radiated down on the responder. This might be measured with a thermocouple.

The scbaPass value is true when the tank's PASS alarm is on. To save on characters, like many Boolean values, a 0 or 1 is used.

The scbaToAlarm value is the number of estimated number of seconds until the responder's low air alarm will sound. This is an advanced value which would 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.

The scbaToZero value is an estimate as to how many seconds until there is no usable air left for the responder.

The scbaStill value is used to provide the amount of time (in seconds) that the responder has not moved.

The scbaTempMask value is the temperature inside of the responder's face mask.

The scbaResiprations value is the respiration rate for the responder. These values, along with scbaLux field (which indicates the brightness of the first responder's environment) can give a feel as to what conditions are like for the responder. The accelerometer values from the earlier Environmental Health schema (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.

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

Indexes:	Short Form	Global?	Units	Required	Format	OOBNV
vehId	vi	Global		✓ Yes	integer	-1
vehHorsepower	vh		HP	✗ No	integer	-1
vehRpm	vr		RPM	✗ No	integer	-1
vehTempOil	vo		Degrees C	✗ No	40.2	-1
vehPressOil	vn		kPa	✗ No	275.2	-1
vehTempCool	vc		Degrees C	✗ No	35	-1
vehLoad	vl		percent	✗ No	integer	-1
vehFuel	vf		percent	✗ No	integer	-1
vehFuelCapacity	vt		gallons	✗ No	40	-1
vehFuelConsume	vf		Gallons/Min	✗ No	0.31	-1
vehGVW	vgvw		pounds	✗ No	4200	-1
vehGndClear	vgc		inches	✗ No	14	-1
vehWidth	vmw		inches	✗ No	92	-1
vehLength	vml		inches	✗ No	384	-1

NOTES:

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 water tank size, pump GPM (gallons per minute), SCBAs carried, oxygen carried, etc.

Note than some of the information displayed in this object is static, such as fuel tank capacity. However, this static information when combined with the dynamic information of the current fuel level and current consumption rate 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 is included in this general vehicle object.

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

This information is presented in a format different to the previous data objects. This is 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

The Server Status object allows the 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 onto 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.

The timestamp field is used to indicate the last time something was changed and is used for cache control. 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 this information is not typically transmitted to the lower power, lower bandwidth IoT devices, and because of the lower rate of change of the status information, the short indexes are 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.

The current values for the various bit mask functions are listed above. This area is currently in development and testing. The relevance to IoT data is in the "bandwidth_available_world" and "connected" fields. These fields affect how often IoT data is sent and how often temporally stored IoT data is flushed back "upstream" in situations where connectivity is intermittent. There are real world IoT challenges.

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. This is a very classical example of taking a data table, encoding it, sending it, receiving it, and then updating whatever values are sent in your local data object.

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

This information pertains to the status of any source of power, such as a generator, uninterruptible power supply (UPS), or battery-UPS power system.

This information is presented in a format different to the previous data objects. All values must be presented inside of the pwrStatus object, 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.

Indexes:	Short Form	Required	Description
name	n	☑ Yes	Name of the power source. This would be understood by the server that is hosting/supplying the data.
type	t	✕ No	What type of power system this refers to. This is a text string, but there will need to be an agreed upon list for common systems "generator", "battery", "ups", "battery-ups", "solar"
voltsIn	vin	✕ No	For battery to UPS or solar, the input voltage
ampsIn	iin	✕ No	For battery to UPS or solar, the input current
voltsOut	vo	✕ No	output voltage of either AC power or battery voltage
ampsOut	io	✕ No	current being consumed at this point in time.
ampsOutMax	max	✕ No	Maximum current that can be drawn.
watts	po	✕ No	current output power in watts.
wattsMax	max	✕ No	Maximum power available.
remaining	t	✕ No	The amount of remaining time (in seconds) until this source runs out of power. Minus 1 means unknown.
fuel		✕ No	percent of "fuel" remaining. String. Could be "5 gallons", "20%". Battery systems use KWh.
fuelCapacity	ftank	✕ No	How much fuel the source holds. For engine power devices, this is in gallons. For battery systems, use KWh.
status		✕ No	"stoplight" status. Green, Yellow or "Red is Dead".
faults		✕ No	Text string with any problems, such as "redundant supply failure", "lower battery voltage", "low fuel warning"
info		✕ No	Short string with information, typically clickable
details		✕ No	Longer text description of the power source
dept_id		✕ No	
truck_id		✕ No	means to identify the engine associated with this generator.

NOTES:

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.

Note also that the units of some fields, such as fuel, have units appropriate to the type of power system.

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

This is information about images that can be overlaid onto a map. Overlays are a graphic image such as a .jpg or .png where the latitude/longitude of each corner is specified. The overlay can then be places on top of a map. 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.

Indexes:	Short Form	Required	Description
north	n	☑ Yes	Lat of "highest" North point
south	s	☑ Yes	Lat of "lowest" South point
east	e	☑ Yes	Long of the east-most point
west	w	☑ Yes	Long of the west-most point
url		☑ Yes	The URL to the overlay image
op		✕ No	Default Opacity, 0 to 1
info		✕ No	String with information about what this overlay is
address_object		✕ No	For building overlays, the address for this overlay
address		✕ No	Address for this overlay, if full address object not warranted
floor		✕ No	For building overlays, the floor of the structure
ts		✕ No	When this overlay was last modified by any process
enteredBy	eb	✕ No	Some means to identify the PSAP/Dept/User that created this.
ts_entered		✕ No	Last time this was created/edited/modified
cfsNumber		✕ No	If this overlay is associated with a call, this is the Call for Service (CFS) in the PSAP.
psapId		✕ No	The ID number for the PSAP
poly		✕ No	Name of the polygon for "hits" on this overlay
poly_psap		✕ No	The psapID for the poly
poly_deptid		✕ No	The department ID of the department that "owns" the polygon
poly_incAuto		✕ No	Bool. If true, if an incident is inside of the specified poly or bounds, then the poly is 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.

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

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.

Indexes:	Short Form	Required	Description
path		✓ Yes	A list of lat/longs in the form of: **"36.1234,-79.4321|36.5678,-79.9876| ... | 36.9999,-79.1111"** **"lat_1,$long_1$|lat_2,$long_2$|....|lat_{N-1},$long_{N-1}$[|lat_n,$long_n$]"** The separator is the vertical bar, ASCII code 124, hex 7C *Note: the path strings can be very long (many kilobytes)*
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 number of pixels in the outline
strokeOpacity		✗ No	The opacity of the outline
fillColor		✗ No	The color of the inside of a polygon
fillOpacity		✗ No	The opacity of the inside of a polygon
meta		✗ No	String that describes the polygon
ts		✗ No	Last time anything changed
enteredBy	eb	✗ No	Some means to identify the PSAP/Dept/User that created the object.
tsm		✗ No	Last time the object was modified by the user.

Notes:

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 (lat/long) is inside of a polygon or not. Any software utilizing this object should have the same flexibility.

7 CONCLUSIONS

Throughout this paper we have touched on many different areas, ranging from viewpoints directly from the fire service, to the needs 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 is still 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 good 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 still a chance to control some level of how their information is conveyed. There are many, 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. Future documentation covering law enforcement and emergency medical services will attempt to outline similar software building blocks for those first responder entities. We also hope to hear from the developer community regarding information that is missing. 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 is the Industrial Internet of Things (IIoT)?". HPE. Accessed on March 27, 2020. [Online]. Available: https://www.hpe.com/us/en/what-is/industrial-iot.html

[2] *"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/.