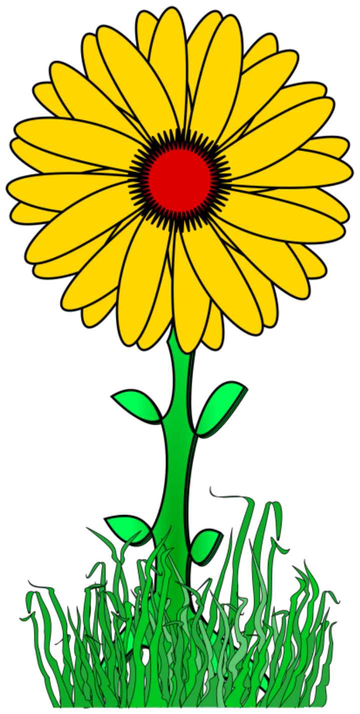
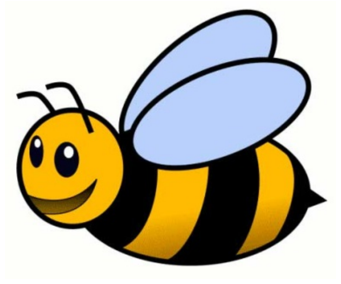




I^3

Intelligent IOT Integrator (I^3)
University of Southern California
A joint project: Marshall and Viterbi

Requirements
Feb 19 2018

Bhaskar Krishnamachari (bkrishna@usc.edu)
Cyrus Shahabi (shahabi@usc.edu)
Jerry Power (jerry.power@marshall.usc.edu)
Seon Ho Kim (seonkim@usc.edu)

1. Introduction

The Internet-of-Things (IOT) is in its infancy. As with any emerging technology, one should expect a heighted level of evolution in this space during these formative years. Such an evolutionary trajectory will make it especially difficult to build and operate an IOT system because product lifecycles will be short and there will be a high frequency of updates.

Additionally, while a university-focused IOT system that supports the operational needs of a university may be similar to a smart-city or a smart-factory although smaller in scale, the university-focused system carries the incremental task of providing an IOT educational environment for students and a research environment for faculty. These incremental needs can be characterized by the nature of the short-term application requirements that disappear once the academic year or research project is complete.

The envisioned I^3 system seeks to make it much easier to develop and deploy IOT applications and sensor network by maximizing the level of reuse between different applications. The modularized nature of the architecture simplifies the process of adding new sensors and applications as the system evolves over time.

Many companies and universities are doing research in the area of IOT but most such efforts are targeted at a specific application or use case and assumes that the necessary sensors will be driven by the application provider. This creates a 1:1 relationship between sensor and application that reduces sensor reuse and increases application operational costs. It also serves to inhibit the emergence of an independent IOT application development market. While some IOT living labs have appeared at some university campuses that allow IOT testing in live environments, they do not address themselves to systemic technology reuse and the emergence of a vibrant market for IOT data exchange.

The I^3 system platform can be used in conjunction with other platforms to meet specific needs. For example when the I^3 platform is used in conjunction with a data analytics platform or a web-services platform, the I^3 system serves to manage sensor access while the web services platform manages the application proper and the data analytics platform manages analysis of the resulting data.

The USC I^3 system envisions a world where application developers can gain access to the myriad of sensors that others have deployed and connected to the network. The I^3 system envisions a world where sensor owners can take the initiative and deploy intelligent sensors in anticipation of an emerging and independent application market that will utilize their data for the benefit of its users. Under the I^3 vision, an IOT network will allow easy integration of different kinds of sensors from many different vendors to create an application support environment that advanced analytics to be developed and supported in a multivendor environment. The I^3 platform is the information bus that allows the level of integration the future world needs to occur.

The I^3 system is an IOT domain controller will support the following goals:

- Support research needs of all USC schools (a multidisciplinary research platform).
- Support educational needs of all USC schools (a multidisciplinary educational platform).
- Support the needs of the USC service organizations (all USC operational departments)
- Provide testing facilities to businesses so they provide funding for opex and capex expansion
- Provide IOT data of value to businesses so they provide funding for opex support

The I^3 system expected to be reused by government and business entities as an IOT domain controller which supports the development of a local data-based ecosystem:

- Support protected and managed access between applications and an IOT domain
- Support sensor owners seeking to sell their device data in an open data marketplace
- Provide the administrative support so the system can sustainably support itself based on local traffic (sustainable support include technical sustainability, economic sustainability, and social sustainability).

The I^3 will allow the university, government entities, and large corporations to control energy costs based on intelligent inputs. For example, whereas Nest-type systems allow energy management based on time of day and Nest-detected trends, the I^3 system will support Nest-like thermostats that can also adjust environments based on the number of people in a building, scheduled events, and emergency situations. Vehicle maintenance sensor systems are no longer limited to managing vehicle system based strictly on vehicle sensors but can now be integrated into maintenance shop schedules so parts are pre-ordered and vehicles maintained operations are scheduled during low demand periods and after the needed parts have been delivered. First responders (police, fire, ambulance) can be not only integrated based on alarms and communications but now can be integrated into infrastructure systems allowing them to automatically order building shutdowns, divert traffic from dangerous situations, and manage the orderly evacuation of civilians. Student and staff tracking systems that provides oversight of people in public and private areas with video recognition of unknown/uncensored individuals.

The I^3 ecosystem allows application developers to provide remuneration to the sensor owners based on their data needs (the value of the data). The application developers and the sensor owners have no direct interaction with each other – the I^3 system acts as a mediator between these two parties. The I^3 system provides privacy for the sensor owners by only revealing the data the owners wish to reveal to any specific application entity. The sensor owners must trust the I^3 system to act on their behalf so the systems have a large degree of control.

2. Benefit of I^3

The benefits of the I^3 are many and can have wide ranging benefits for universities, business environments, government, and any other entities anticipating a future where integrated IOT concepts will be at play. For researchers, I^3provides a platform to efficiently test IOT adoption motivators, inhibitors, societal impacts, and business models. The I^3 platform also allows new application and devices to be developed and tested in order to evaluate potential users (performance, features, design, etc) issues. Without a platform concept, development of temporal applications and sensors would be prohibitively expensive discouraging the level of research needed to face this rapidly evolving world. For commercial IOT developers, the I^3 system creates a large scale, multidevice, multiapplication testing environment. These I^3 platforms provide a vehicle that allows these companies to perform market testing of IOT applications and sensors in a managed but complex environment. Further, the existence of I^3 platforms allow the creation of operational environments where the costs associated with a specific IOT application can be shared thereby improving application prove-in metrics. For students, the I^3 system provides a unique environment for learning about engineering, business, and societal impacts of IOT, a formative factor that will have significant impact to future generations. Finally, for the university and other such large institutions, the I^3 platform provides an evolvable platform to support the forward looking needs of the Department of Public Safety (DPS) and other campus services. It also creates a proactive business environment for independent business entities looking to operate in a business climate with a proactive operational business support infrastructure. Finally, the I^3 platform provides a vehicle that may allow marketing groups to highlight an increased number of IOT applications thereby drawing increased attention to efforts to shape the next industrial revolution.

The overriding goal of the I^3 system is to create IOT communities where device owners are an active part of the community. These IOT communities can link themselves together to create larger communities. Applications and data brokers access and provide services to these communities that would not be possible if the communities were not organized by the I^3 system

3. General Requirements

The word 'platform' is an overused word that is often misunderstood. A platform is a system that can be reused for multiple purposes. The costs associated with a platform can be amortized across different applications allowing platform systems to make more economical use of the shared core. Platform systems can be linked and layered allowing platforms to contain other platforms as needed to maximize reuse.

- As a platform system, the value of the platform is increased by encouraging wide scale reuse of the platform. The architecture should be built to encourage the members of the user community to wide-scale deployment of platforms, sensors, and applications. The intent is that platform deployments will go viral and once the platform is deployed, sensor and application deployments should increase rapidly. This implies the need for easy deployment, easy addition of sensors, and easy addition of incremental applications.
- The application developer/owner may be independent from the sensor owner/deployer. In the I^3 system, a single sensor which is collecting desirable data may be providing that data to multiple applications which are making use of the same data in different ways. Therefore applications should be linked to sensors through an abstraction layer. Further a single application can access multiple sensors of different types and a single sensor can support multiple applications of different types and from different developers.
- The cost of deploying and maintaining a network of sensors can be financially supported by multiple applications, this allows per application cost of sensor deployment to be reduced and this should stimulate accelerated sensor deployment. There should be a mechanism that allows each application to track its use of each sensor in order to support proportionate billing of platform utilization. Ultimately, each application would only pay for access to the data they specifically need, further reducing an applications operational costs thereby stimulating accelerated application deployment.
- The I^3 system should to support business models that allow sensors owners to resell their data thereby allowing sensor owners to recover the cost of deployment. This means the I^3 system has to track which applications access data from specific sensors. Sensor owners can trade these tracked usage credits for money or for application usage offsets.
- Initially, the expectation is that access to the I^3 IOT data will be traded for application access (a freemium type model) but longer term the platform system could serve to create a marketplace for exchange/sale of IOT data. For example, proactive system users may deploy sensors and sell/rent their IOT data as marketing fragments in order to generate funds that may be traded for internet access services.

- Sensor owners need to have complete control as to which applications have access to their sensor data (privacy). The I^3 system should allow sensor users to configure different privacy settings for each application with access to their sensors identified by the registry. Support for privacy at the registry layer reduces the complexity of the software contained within the sensor and allows a single user to make wholesale security changes to all owned sensors from a single operator console.
- The I^3 system has to place a high level of emphasis on trust (security) and take extreme efforts to safeguard the data from hackers and other nefarious users. Security software must guard against virus, worms, and denial of service (DoS) attacks. The I^3 system should also prevent a valid sensor owner from gaining access to data or configuration data of other sensor owners. In addition, the I^3 should prevent application owners from gaining access to data or configuration data of sensors that the sensor owner has not explicitly granted.
- A wide range of sensors may be owned, deployed, and managed by a wide range of users. Multiple users may own similar sensors that are deployed in different places. It should be possible for two or more owners to deploy the same sensor in the same location. The I^3 system provides a central point where data of different types can be integrated and used for systemic intelligence. Some sensors will report binary conditions (on/off), others digital counters, others text, and others video or picture images. The I^3 collects all such data and allows applications to take action based on the entire range of information available to it.
- Video based sensor data is very different from structured sensor data. The expectation is that unstructured data, such as video and image data, is stored in a related but different storage area than the structured data. The system should be optimized for near real time support of structured data but these performance requirements do not hold for unstructured data supported by the I^3 system. Video captured from IOT devices may be streamed (continuous) or sporadic (non-continuous, where video stream is auto-activated upon motion detection and stopped when motion halts). Video that is delivered to the I^3 system can be sent to one or more subscribing applications; applications can be added or deleted from the video subscriber list without requiring a reconfiguration of the ongoing streaming applications.
- In this document the term sensor is used generically and it is understood that sensors can sense and provide data to the registry. Additionally, the register can set a wide range of configuration parameters in the sensors. For example, a sensor can report the current temperature in a building and the target temperature can increased, decreased, or set to a specific target value by changing configuration parameters.
- The system should minimize the need to manually download applications to sensors and smartphones. For example, instead of requiring that each IOT application be downloaded to a smartphone, a single application should be downloaded and this application periodically reports sensor data to the central registry. The many applications that use this sensor data retrieve the data from the registry and do not directly interact with the smartphone.
- It is impossible to envision all future sensor types; the expectation is that new sensor types will continually become available. Support for new sensor types can be added over time as incremental sensors are deployed by expanding the functionality of an existing sensor driver module or by creating a new sensor driver that supports the registry by collecting the sensor data and properly filing it in the registry.
- In some cases, sensors will be physically deployed and then configured in the central registry. In other cases, the sensors will be defined in the registry and left in an inactive state awaiting some future physical deployment. Sensor owners can turn-on or turn-off sensors which prevents the I^3 system from attempting to communicate with the sensors. State settings allow inactive

sensors to report most-recently collected data to the permitted applications or cause the system to report an inactive state.

- There should be audit functions that the legitimate operation of all software processes on I^3 servers (downloaded applications, central system applications, and sensor driver applications). Since applications and driver applications may come from third parties this implies an ability to characterize the operational parameters of unknown software so that alerts can be raised if the operational parameters exceed expected bounds.
- External parties are expected to hear about I^3 system via social networking. Social networking references are expected to point interested parties to an I^3 website. The website has to have pages to provide an overall system description, provide design guidelines to potential implements, support an FAQ page, provide a download page, instructions for implementers and developers, a support page, a contribution page, and a means to announce upcoming events such as new releases, virtual events, and conferences.
- Running systems should auto-check for updates and be able to actively update themselves when an update is available. All systems should periodically check to make sure their operating software is the current. Operational checks should be more exhaustive than simply checking a version release number as this would allow people to hack/edit operational systems improperly.
- The I^3 system should be opensource so there is no need for deploying entities to purchase incremental software licenses and so the I^3 user community will be able to actively participate in the I^3 development process.
- As an opensource system, ultimately there may be multiple development and test systems spread throughout the globe. There should be a means for developers and testers to communicate and share bug reports, feature requests, to post development messages, and to pose questions to the larger community (and get answers).
- Continuous Integration (CI) and Continuous Delivery (CD) practices should be followed. Development should be not done on live/operational systems but on dedicated development systems. Administrators should be able to flag whether a system is an operational system, a development system, a test system, or a beta-test system. Using CI processes, developers can submit their software to an automated integration system which creates temporal system builds (giving the developers feedback). Periodically, a temporal system build is advanced as a candidate release and it is promoted to a laboratory test system. Candidate releases which pass laboratory testing are promoted to beta-test status and released for testing by I^3 multiple testing partners. Software that passes beta-testing becomes the latest general release software. Under CD concepts, there will always be different iterations of the software in development, test, beta, and in general release.
- It is possible that after a candidate software release has entered into the testing, a significant error will be discovered in the current general release of the software. The configuration management software (GitHub) should be able to reconstruct the current general release, allow the faulty software module to be corrected, and suspend the candidate software testing process so a point release against the current general release can be fast tracked through the testing process. One the point release passes testing, the point release becomes the latest general release and testing of the new candidate release resumes. (The point release can be distributed to field systems as either a complete new load or as a minor software module update.
- Near term, I^3 systems will be deployed by IT people with different skill sets, different abilities, and access to different tools. Ultimately, it is possible that I^3 systems will be a part of every residence. Therefore, the installation process should be simple, straightforward, and self-

contained so people can install the system without problem at another campus, at a small office, and at their personal residence.

- Registry has to be operationally very fast (low latency) – it must add 'almost' no delay to the system so applications are operating in response to sensor changes in near real-time. For small systems performance targets can be met with an integrated I^3 system that is configured so the registry, applications, and sensor drivers are all running on a single machine. For larger systems, this may mean the registry has to operate on a dedicated machine, the admin interfaces, logs, etc are then stored on a networked machine while the sensor drivers and applications running on even different machines. There should be documented operational guidelines so the administration can plan an I^3 system architecture that will meet their specific needs. The guidelines may preclude use of virtual machines and may demand hardware specifics (e.g. SSD, minimal CPU/memory specs) based on expected demands of the performance targets of that administrator.
- The system is scalable allowing it to support both large and small institutions. Most administrators will want to specify their system needs in terms of the number of supported sensors. Sensor complexity can vary widely with some simple sensors reporting simple data and some complex sensors such as a smartphone reporting a complex set of information. Therefore, scalabity measures may even vary within any targeted configuration capacity.
- There is an expectation that sensor multiplexers will need to be supported by the I^3 system. A sensor multiplexer is an intelligent remote that can collect sensor data from numerous sensors and report them to the I^3 system as a composite data stream.
- It is possible the I^3 system will eventually exceed the parameters that were used to design the system on initial deployment. An integrated system can be easily migrated to a distributed system without taking the system off-line. Communications modules and applications can be migrated between client machines by the system operator to facilitate system maintenance.
- The system has to be extremely reliable and should operate 24x7 without manual operator assistance. Philosophically, the design of the system should expect that operators will only be physically available during working hours but the demand of application and sensor owners are must be progressed 24x7.
- As the I^3 system registry grows, at some point it will exceed the performance ability of a single machine and the registry will have to split so half the registers are moved to another machine and these two machines operate as peers. There should be a smooth and near seamless way to manage this transition.
- Sensors can be read sensors (e.g. location reporting sensors), write sensors (e.g. electronic signs), or read/write sensors (resettable counters that might count cars entering/exiting a parking lot).
- The IOT world is in its early days and the expectation is that new sensors will be invented over time making it impossible to anticipate all future sensor types. The I^3 system internal APIs should be open and extendable (e.g. XML or Restful API) allowing software module to be upgraded sequentially to support new functions without requiring a complete system reset.
- An I^3 system has multiple log files where each log file reflects a specific type of activity. For example, there may be a billing log file that tracks all billable activity, there may be a system administration log file that tracks the action of local I^3 system administrators. There may be a sensor manager log file that tracks the actions of the sensor managers. . There may be an application manager log file that tracks the actions of the application managers. Individual log files can be examined separately; all log files have a similar structure to make it easy for

administrators to track overall activity. In addition, there is a utility that that can interleave specific log file actions allowing operators to see the sequencing of different event types.

- The I^3 system supports email and other messages between application and sensor managers but the sender and respondent user identity is masked from the other party (e.g. local I^3 user names may be used) to provide anonymity until the user wants to disclose their true identity. The I^3 system and I^3 system managers can email sensor and application managers and these managers can reply to these message (without attachments) using the logical identifiers but these external managers cannot initiate an email an I^3 system entity. Any attempt by an external party to initiate an email sequence should be blocked as a security measure in ensure viruses cannot enter the system through email. The intent is to block unexpected and unnecessary transmittal of electronic information into the I^3 system as most cyber-attacks begin with seemingly innocuous transmittals of digital data. For this same reason, support of USB connected devices should be rejected by default; supervisory overrides are needed to get such devices recognized by the system and only after these devices pass any needed security scan.
- Security is expected to be a major concern needed to win the trust of the sensor owner community. Security experts agree that the greatest security facing any administration is the malicious insider and operator errors. To minimize the potential impact of such issues the log files and backup files should be encrypted. The registry content should be strongly encrypted by the sensor owners (the I^3 system managers do not have the key. There should be no root access and insiders should not be able to obtain root access by rebooting or restarting any system.
- There should be different layers of system administration. For example, the lowest layer of system admins should have read only access. Specific layers should have permission to manage application managers, another layers should be able to manage sensor managers, another layer should be able to manage the registry, and another layer should be able to manage processes. A human operator may have different logins so they can log out of their read-only account and log in again to manage the registry. Any logged in sys admin should be auto logged-out after some defined period of time (e.g. 60 minutes) forcing them to periodically authenticate themselves. A period of inactivity should trigger an autologout process.
- Sysadmin operators cannot email files out of an operational system. Files cannot be exported/taken from the system by copying them to USB (or other removable media devices). When new USB systems are detected, the files on those devices are isolated from the remainder of the system so viruses cannot be brought into the system via an unknown USB stick.
- Detection of any changes in the hardware configuration is logged. Hardware and software configuration reports are periodically logged and compared against expected templates to determine if new hardware or software has been added to the system.
- Multiple storage options should be supported and mechanisms to manage different file types should be supported. For example, backup files (used for recovery) may be stored locally, on network attached storage (NAS), or on a cloud server. Non-current log files may be stored on networked computers. Defined (and tested) configurations need to be documented so entities can easily replicate the targeted operating environment.
- In the I^3 system, there is a directory of sensors and a directory of applications that can be searched and used as the basis for reports. This implies there is a naming standard for sensors and for applications. It may be possible to reuse standard directory structures such as a DNS system to manage these systems. See the discussion section on reconciliation for elaboration.
- The communications channels between the I^3 system and the IOT devices can be secured to prevent monitoring, to prevent insecure devices from masquerading as secure devices, to prevent illegitimate I^3 systems from masquerading as legitimate, and to prevent valid devices

from accessing the I^3 system with invalid software. Securing the link between the devices and the I^3 system will draw mischievous users to the I^3 system where presumably significant attention will be applied to preventing systems level attacks.

- Each I^3 system is an autonomous system with links to other I^3 systems. Each I^3 system administration decides how to 'pay' device owners for the data their devices produce and payment means may vary between I^3 systems. Systems can vary by receiver allows a corporate system to collect all payments, a charity based system can use the proceeds to fund a charity. Systems can vary by currency which payments are made.
- A hub I^3 system only accepts mirrored references from subordinate systems allowing a hierarchy of I^3 systems to be constructed.
- Smartphones are considered one of the most significant types of IOT device because the smartphone can create IOT data of its own accord, the Smartphone can act as a Bluetooth hub and collect I^3 data from other devices, and the smartphone can support internet access to allow the smartphone to access applications that reside on the northbound side of an I^3 system.
- I^3 systems can be run on a variety of machine types. For instance, web based servers and on virtual machines as a convenience for the system administrators, however, it is difficult to provide security assurances for systems running on hardware/software run by other administrations. Therefore, there are different hardware/software configurations of the I^3 system that provide a defined level of security assurances for the sensor owners that decide to connect their devices to a specific administration.
- I^3 systems can be upgraded in place without requiring a complete system shutdown or reboot in order to support 24x7 operational support. When software is to be upgraded, individual software modules/processes are upgraded sequentially so the system is not ever completely taken off-line.
- IOT devices are outside the control of the I^3 platform, however it should be possible to have an absolutely secure link between the IOT device and the I^3 system. If the IOT device meets a defined set of I^3 requirements so that the device will only communicate with the I^3 platform (and nobody else) and the I^3 system can validate the device identity, the IOT device becomes directly unhackable. This will divert hacker attention to the I^3 platform as the gateway to the IOT device but it is assumed that the I^3 systems can mount a more rigorous hacker defense than an IOT device.
- When a sensor manager defines a new sensor to the system, a message is sent to all application managers so they are made aware of the new sensor. When a new application is defined by the system, a message is sent to all sensor manager making them aware of the new application's data needs. These messages are stored in a directory so the sensor manager and application managers can browse the available sensors/applications at their leisure. Monthly (or another time as defined by theI^3 system manger) the application manager or the sensor manager can trigger a manual re-advertise message.
- "IOT Systems" are closed IOT environments where many sensors communicate to a local IOT controller and the local controller acts as the gateway to the local sensor network. As an example, consider an airplane which has thousands of sensors that are managed by an IOT control system in the cockpit. For the purposes of the I^3 platform, the airplane is consider a single IOT system since any interaction with the IOT sensors is managed by the control system gateway. From the I^3 perspective, all managed IOT devices have a port to a public communications network. In general, the I^3community must be aware of local communications systems such as ZigBee, BlueTooth, NFC, and RFID but these technologies will generally not be used to access the I^3 system.

- IFTTT are not considered an analytics network but does provide a programmable means to directly react to sensed conditions. Analytics are not considered part of the I^3 system but Analytics programs can access and use the I3 system for easy access to a network of IOT devices. At present, the I^3 system would expand the concept of any IFTTT system by making it aware of a larger network of IOT devices; at present such an expanded definition of IFTTT is considered outside the scope of I^3 but that decision is subject to future review.
- I^3 will provide full support of both IPV4 and IPV6 in order to allow the I^3 system to communicate with devices and applications using IPV6 (preferred) and alternatively legacy IPV4 systems.
- Following the SDN philosophy, the administrative applications that provide the functions that manage the registry, generation reports, manage device-users, and application developers can be run remotely from the I^3 core (connected to the core via the internet).
- Message flows between devices and applications are metered for billing purposes and so the flows can be compared against expected norms. If these message flows begin to exceed expectations, the I^3 system will warn of a potential security breach. If these deviations are extreme, it is possible to configure the I^3 system to auto-shut down the application and/or device message flows until the situation can be investigated.

4. Marketecture

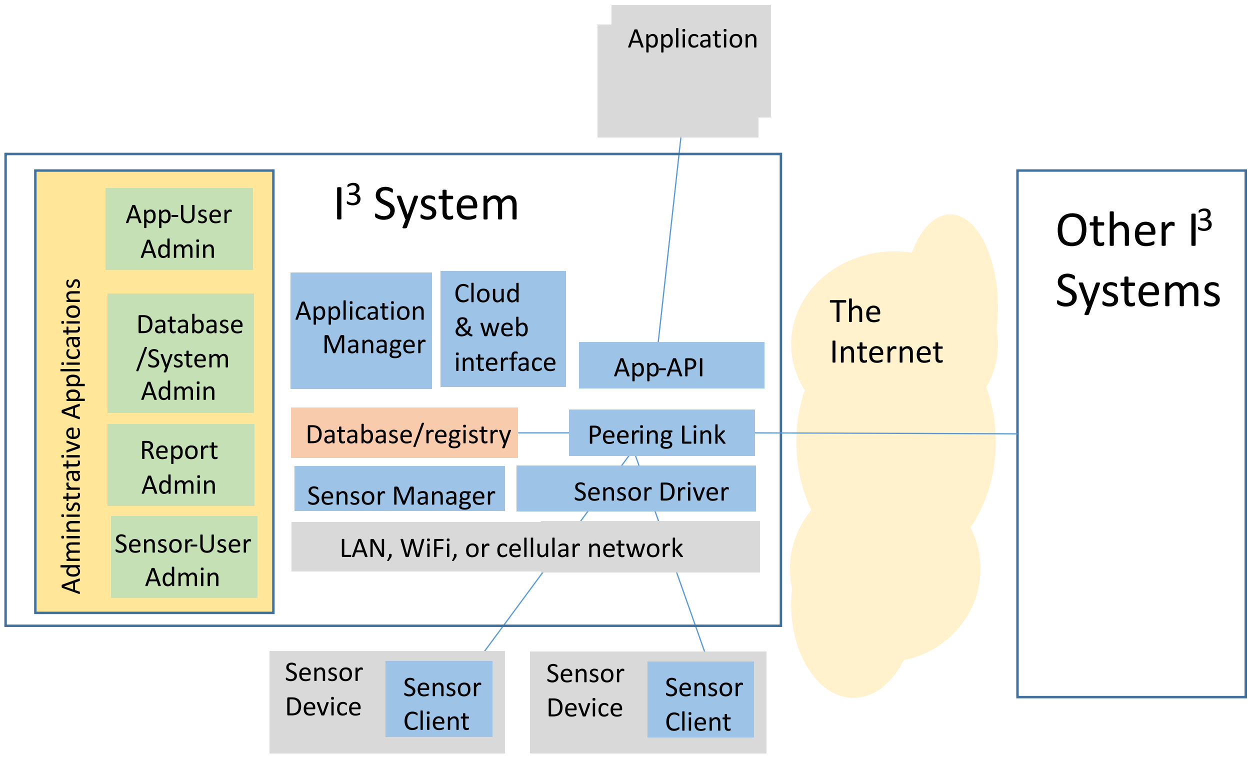
Core of the I^3 system is an active registry. As a registry, it contains a description of all the sensors, beacons, and applications visible to the system. The active registry points to a virtual representation of the IOT device that is constantly updated to reflect the current state of the physical sensors. The I^3 system simplifies the IOT systems in the registry/database provides privacy features for the user so that only applications which the sensor has approved has visibility to the data from the sensors owned by an individual. Centralization of this software reduces the complexity of the software located in the sensor. Additionally, because the registry/database allows applications visibility of the sensor data as standard database calls, the complexity of the applications is reduced because each application does not need to understand the unique features of the sensors they are accessing.

While beacons are not sensors in themselves, they do act to trigger activity in an IOT devices when these devices come near the beacon. There are two kinds of beacons, App beacons and Physical Web beacons. App beacons transmit a device Id to the IOT device and the device is expected to have sufficient intelligence to make use of that ID. The beacon IDs may be configured in a registry/database which would allow a smartphone to use the ID to retrieve information from the registry and then act based on the registry supplied information. In contrast, the Physical Web beacons can transmit a URL to the IOT device; these URLs can reference a web site (possibly with beacon IDs as a parameter) eliminating the need for beacon specific applications in the IOT device. While these examples consider beacons as fixed devices that are detected by mobile IOT devices, beacons are small enough they could be the mobile devices (e.g. mounted to a bike) and their presence could be detected by an fixed detectors as well.

In this document, sensors are assumed to be bidirectional devices. Strictly speaking, sensors are often used to refer to devices that report data and accentuators are used to control a report device. Sensors are read-only devices and accentuators are write-only devices. In this document, the term sensor is used more generically in that it is assumed that sensors can be remotely managed (polled for data, operational parameters set, etc)

The I^3 active registry is application independent and is designed to allow many applications to share access to a specific sensor. Sensor sharing requires that the point of sharing is moved away from the sensor where it can provide common services for multiple applications. This same philosophy streamlines the operation of complicated sensors. For example, instead of having to load multiple IOT specific applications to a cellphone, a single generic sensor is loaded to the cellphone and that application uploads sensor data to the active registry allowing the registry to manage application access to the sensor data.

A marketecture is a logical description of a system's components that is used to better describe the functionality of the system. As a logical depiction, the actual physical realization may be significantly different from the marketcture.



Logically the on I^3 system consists of the following functional modules

- Application Manager: A single application developer can own/control multiple applications and this module allows a developer to manage their applications. The manager can start, stop, restart, delete, and load new applications to on I^3 system.
- Applications: Multiple Applications can be supported by each user of an I^3 system. Applications are not a formal part of the I^3 system but the I^3 system is aware of local applications running on its server.
- App-API: This module manages application access to the I^3 system to ensure that the application is properly behaved and does not compromise the I^3 system integrity. In the client/broker or the publisher-subscriber sense of the word, there is a 1:1 relationship between the application API and the remote application.
- Cloud & Web Interface: I^3 applications can be loaded on the local I^3 system or they can be remote from the I^3 system (e.g. on a public or private cloud). Remote applications can access

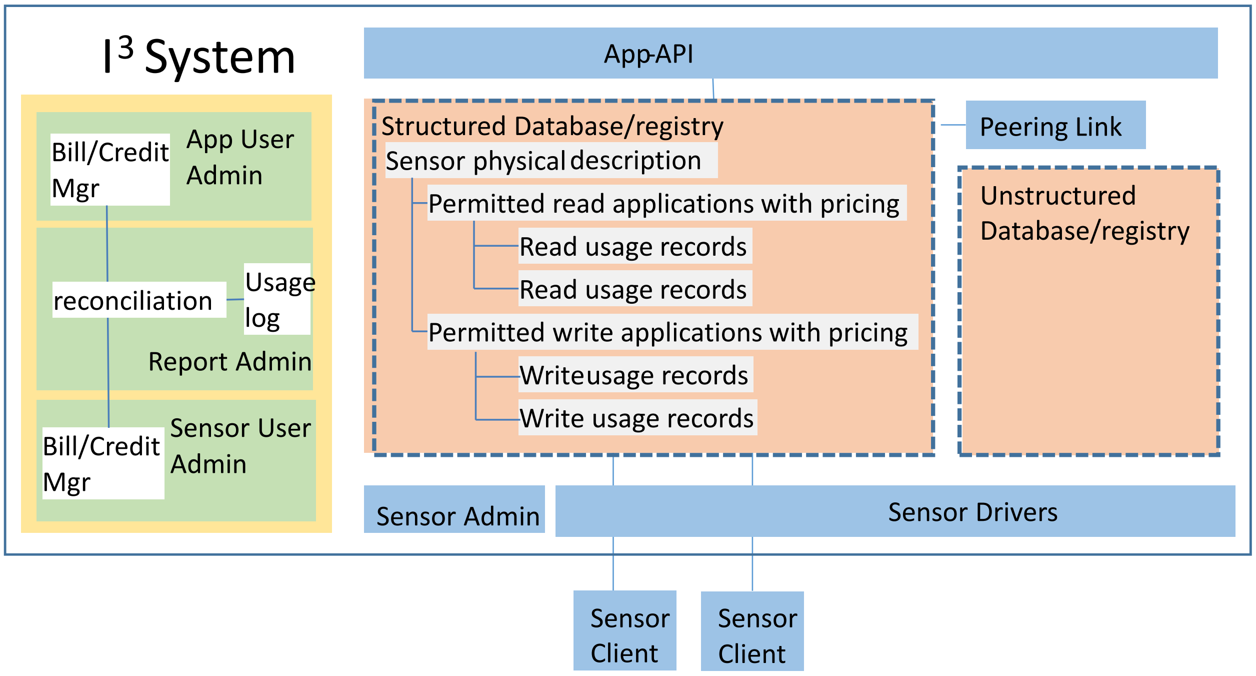
the I^3 data exchange via a fully functional API. In addition, there are basic HTML applications that developers can used to manually probe the I^3 message exchange in order to validate the status of its register.

- Database/registry: The database/registry contains descriptive information about beacons and sensors distributed throughout the I^3 system's span of control It also contains a collection of registers that reflects the current status of the sensors in the field. These registers are automatically updated as the sensors change state so the cache contains an accurate representation of the sensor status. In the client/broker or the publisher-subscriber sense of the word, there is a 1:N relationship between the database/registry and the application API modules. Additionally, there is a there is a 1:N relationship between the database/registry and the sensor driver modules.
- Peering Link: Multiple I^3 systems can be tied together. When two I^3 systems are linked either system can create a register in its local cache that is not tied to a sensor but instead reflects a register in the remote system; the local database/registry is updated as the remote register changes state.
- Sensor Manager: A sensor manager is responsible for configuring all the sensors in the registry that are owned/managed by a single person or entity. A single sensor manager may be responsible for multiple sensors. The sensor manager module is used to create, delete, and to set the permissions flags that provide application specific visibility of a specific sensor register.
- Sensor Driver: The sensor drivers interact with the remote sensors clients to determine (or set) register values. The sensor driver can accept autonomous indicators from the sensor client and changes the cache registers to reflect the status of the physical sensor. The sensor driver can also poll sensor clients if the client cannot create autonomous alerts or when a validation action is being performed. In the client/broker or the publisher-subscriber sense of the word, there is a 1:1 relationship between the sensor and a sensor driver.
- App-User Administration: The I^3 systems administrator uses this module to configure the application owners so they can use the application administration module to manage their applications.
- Database/System Admin: The I^3 systems administrator can use this module to manually manipulate the cache registers to adjust the configuration of the registry and to manage I^3 upgrades.
- Report Admin: The I^3 systems administrator can use this module to create, schedule, print, or save system reports.
- Sensor-User Admin: The I^3 systems administrator uses this module to add sensor owners to the system so that they can begin defining new sensors in the registry.
- LAN/WiFi network: The LAN/WiFi network is used to connect sensors to the IP network that supports the I^3 system.
- Cellular Network: The cellular network is used to connect sensors to the I^3 system which cannot connect to the system via a local LAN/WAN.
- Sensor Device: Sensors are remote devices that detect local conditions. There is an I^3 sensor client that must be loaded to the in the sensor device that reports these conditions back to the I^3 system.
- Sensor Client: Sensors clients are device specific software modules that must be loaded to the sensor device. These modules report sensor specific information to the sensor driver in the I^3 system.

The cache structure is central to the I^3 concept. A single user (which may be an individual or an employee of a company) can be the designated owner of multiple applications and sensors. In the I^3 system, a user is responsible for configuring and defining the operational parameters (including privacy parameters) for the applications and sensors they own.

To drive viral deployment, the concept of a IOT data marketplace is integral to the system so each application is tracked according to the sensor data they consumer and each sensor is tracked based on the data they provide (with proper approval) to the various applications.

A key component of the I^3 concept is the idea that application development and sensor deployment should be independent market activities. There should be incentives for sensor owners to deploy sensors which serve to make the I^3 environment attractive to independent application developers. In addition, the application developers should be able to gain access to sensor data without having to deploy dedicated sensors or sensor specific software applications. To accomplish this, the I^3 system will keep detail usage information so it is possible to track which sensor data is used by which applications. Ultimately, the applications should be able to 'purchase' access to the sensor data they need as needed and sensor owners should be compensated based on the value of the sensor data they produce. Note: in this context the term 'purchase' may imply a monetary transaction, a barter agreement that trades data for application access, or peering relationship where on parties data need is offset by the needs of another party.



In the I^3 system there are both read and write sensors which are reflected in the read and write cache. Access to read and write registers may be priced differently. The I^3 system also allows users to grant different sensor access permission to different applications. The sensor owner may set sensor pricing differently for different applications.

While the I^3 system tracks usage and pricing, the currency unit is virtualized within the I^3 system and prices only have meaning relative to other sensor data. The translation from a relative price to a real price is managed by an administrative billing process outside the I^3 system. This billing process runs on a scheduled basis and will attempt to manage usage offsets before billing or crediting either sensor owner or application owners. Credits are only issued from the I^3 system after billing statements have cleared.

5. Specific Requirements

5.1. Application Manager

Application Manager: A single application developer can own/control multiple applications and this module allows a developer to manage their applications. The manager can start, stop, restart, delete, and load new applications to on I^3 system.

1. Certified user managers can add applications to their account
2. An application can only see sensors data residing in the Database/registry that the sensor manager has granted permission to access
3. An application manager can ask the system to forward a message to targeted sensor manager who have granted access permission. These messages allow application manager to send informational messages about applications which have permission to access their sensor data. Responses to such messages have owner identity masked so the true sensor manager is hidden from the application manager.
4. Application managers can request an administrative report (printed, text file, CVS) covering the sensors they have access to, who manages the sensors (logical system name only), and how often the applications have accessed each sensor or all sensors in the last X days (total and access per day).
5. Application managers can request the I^3system send permission requests to the sensor owners that they desire access to but have not yet been granted. The system will hide sensor owner identification information for the application owners. A clocking mechanism prevents an application from sending more than one such request to the same sensor owner to one request per month if the owner has not denied the request so the owner does not perceive these requests as spam. If the owner does deny the request, the time-out period until the next permitted access request becomes 3 months.
6. An application manager can ask for a list of sensors where permission has been granted, denied, or not-yet asked. Lists will show sensor types and identify owners by index (hiding their real identity and any means of direct contact).
7. There is an automated process for application managers to recover from lost passwords that is difficult to be hacked by those with malicious intent.
8. For each application manager account, the system tracks whether that account has generated a current credit or balance due since the last billing cycle. It also tracks the last billing cycle and whether the balance or credit has been cleared or is still open.
9. Application manager accounts are tracked to determine when they have last updated their password. Managers that have not updated their passwords are given a gentle reminder and if the password remains updated for a significant period of time, the account and all application access associated with that user are locked.
10. When the sensor manager grants an application manager access to a sensor register in the database/registry, the application has a subscription to the register. If the register is configured to notify the subscribing applications of register state changes, autonomous messages will be sent from the database/registry to the App-API when a state change is detected. The application

manager can discontinue the subscription by issuing a disconnect command. The application can also suspend a subscription if the application only wants to temporarily suspend message flow.

11. Application managers can edit their account information if that information changes. For example, it their external email, billing information, phone number, or social network account references change, the application manager can self-edit this information.
12. Application managers can send messages to sensor manager using a sensor manager's logical reference. Messages can be delivered as to external users as email, TXT, linked in messages, facebook messages, instragram messages, twitter, or snapchat messages. (The system does not support pop-up messages which can be an annoyance for the sensor managers). Each platform supports a different message format so the application manager can specify which message format should be used for the desired message.
13. Messages sent from/to the sensor manager are logged so they can later be tallied for billing purposes.
14. When the application manager sets up a relationship between the App-API and the Database/registry, the relationship is a publish/subscribe messaging relationship. A single register in the database/registry can support have many application side publish/subscribe relationships.
15. Application managers should have the potential to rate the quality of the data from sensor managers and from specific sensors. A high rating of a sensor manager may encourage other application managers to seek out that sensor manager's data. Sensor owners should have the ability to comment on the ratings that they are given by application managers (comment but not change)
16. Sensor data from different manufacturers will be stored in the common cache for easy reference by applications from different application developers. TheI^3 system will be able to produce a data dictionary for the developer reference so they know the kinds of information available to them.
17. Sensor owners who are in arrears (behind in billing) can be administratively locked so the no longer have access to their sensors. Sensors owned by a locked sensor owner can be frozen (all or individual sensors can be frozen). Frozen sensors no longer report sensor changes to applications. All data for locked or frozen accounts are maintained so the sensors can be reactivated after the billing issue is corrected.
18. Application owners who are in arrears (behind in billing) can be administratively locked so the no longer have access to their applications. Applications owned by a locked sensor owner can be frozen (all or individual applications can be frozen). Frozen applications no longer have access to the I^3 system cache or registry. All data for locked or frozen accounts are maintained so the applications can be reactivated after the billing issue is corrected.
19. Before a new application owner is accepted by the system, it will send a validation message to the user's email address to validate the user can indeed be contracted via that email address. When a user adds a new application to the I3 system, it will send a validation email to the application owner to make sure that user did indeed intend to add the device.
20. Message flows between devices and applications are metered for billing purposes and so the flows can be compared against expected norms. If these message flows begin to exceed expectations, the I^3 system will warn of a potential security breach. If these deviations are extreme, it is possible to configure the I^3 system to auto-shut down the application and/or device message flows until the situation can be investigated. The definition of expected message flow volumes can be left undefined (no message volume monitoring), they can be manually specified, or they can be auto-generated based on historic traffic volumes.
21. If an application (data buyer) wants to buy data, they browse the available data and 'apply' to purchase the data. The seller has to approve the sale before the data is released. When the seller applies to buy the data, they have to share things like their data privacy policy, description of why

they want the data, etc. The more trustworthy, the more noble the use, the less invasive the application, the lower the price needs to be to convince the seller to 'sell'. Of course, the seller could be lying about their privacy policy etc so there needs to be a community mechanism to rate the sellers as to whether the sellers can trust the documentation their receive.

22. The application developers will be more interested in a sellers data if they know it is accurate, kept up to date, etc. To enable this, when the application developers browse available data there will be a numeric score that indicates the quality of the data as rated by other application developers.
23. Application developers can browse the application data they have purchased in the past and can rate the quality of the data they have experience with. When rating the quality of the data, the I^3 system will provide a report card that allows the application developers to rate the data in different categories. The I^3 system will compile the rating factors into a single score which will be averaged with other developer scores. The algorithm for converting the report card into a single numeric score will weigh the different criteria. The algorithm will not be made public and it can be modified as part of a system update if, over the course of the I^3 operational life, the weightings are changed.
24. Developers will not be able to comment on specific data beyond the numeric ratings. The fact that there are no comments will prevent developers from flaming IOT device owners and avoid fake comments. Developers owners cannot rate device owners unless they have had experience with that device owner and there are time limits that prevent developers from rating very old experiences.

5.2. App-API

App-API: This module manages application access to the I^3 system to ensure that the application is properly behaved and does not compromise the I^3 system integrity.

1. To ensure security, there should be a 1:1 link between the application on a remote server and the App-API module. Remote applications cannot directly access the I^{3}system without passing through the security assurance features provided by the App-API module.
2. Applications can read or write sensor data registers that is recorded in registry through the App-API module calls.
3. There is an HTML application that can be used for testing systems and to provide I^3 system administrators a local diagnostic support resource.
4. All App-API requests are logged by application, sensor ID, time, date and other identifying information. Log files can be later processed by a scheduled billing process (off-line) to support pay-per-register-access models.
5. Applications are monitored so that if an application goes silent for an excessive period of time or if the application exceeds expected activity levels in a period of time, the application is flagged as potentially having gone rogue. (Silent sensors may be sent a wake-up packet before declaring an error condition).
6. While the I^{3}system may support local applications for testing, application developers cannot download software to the I^3 system.
7. The application can request the creation of a group sensor reports which can later be retrieved via CSV or SQL request.
8. Applications can interact with the I^3 system using native I^3 messages, APIs, and SQL calls. If the application is a legacy application built to expect custom messages and message sequences, it is

possible for the application developer to create an application proxy that translates the legacy interface protocol into I^3 standard operational sequences and messages.

9. Applications can query the log file to get a report of state changes (flagged by time of state change) for sensors they access to over time.

10. The application link can be configured to accept only one access so the link from the app-api is secure and the I^3 system knows who it is communicating with. If the sensor driver and the app-api is secure to a validated user elsewhere in the network, the I^3 system is said to be completely secured.

11. The I3 system will adhere to the NISH cyber security framework (https://www.nist.gov/cyberframework). The framework assumes that the system is capable if identifying potential risks and taking steps to minimize their impact, protecting the system from unanticipated attacks, detecting threats, providing tools to respond to active threats (including user notification), and recovering from cyber attacks.

5.3. Cloud & Web Interface

Cloud & Web Interface: I^3 applications can be loaded on the local I^3 system or they can be remote from the I^3 system (e.g. on a public or private cloud). Remote applications can access the I^3 cache via a fully functional API. In addition, there are basic HTML applications that developers can used to manually probe the I^3 cache in order to validate the status of its register.

1. The local I^3 system administrator has a user friendly means of manually doing everything that can be done by an application through the App-API or through a sensor driver. This allows the administrator to manually correct errors in the system. For example, the local I^3 system administrator can:
   1.1. Create new app users
   1.2. Create new sensor owners
   1.3. Add apps under a defined app user
   1.4. Add a sensor under a sensor owner
   1.5. Can give or deny an app access to a register
   1.6. Can set up a mirrored register on a local machine
   1.7. Can set up a mirrored register on a remote machine
   1.8. Can manually change the state of a register
   1.9. Can do an upgrade of the operations software (all or part of the software)
   1.10. Can create log reports, app or sensor status reports,
   1.11. Can create reports showing the state of all sensor registers
   1.12. Can manually query the state of any sensor register
   1.13. Can change sensor configuration settings (location, polling frequency,…)
   1.14. Can save or restore the status of an individual sensor, all the sensors belong to a sensor owner, all the sensors of a specific type, or all sensors (a complete sensor state backup).

5.4. Database/registry

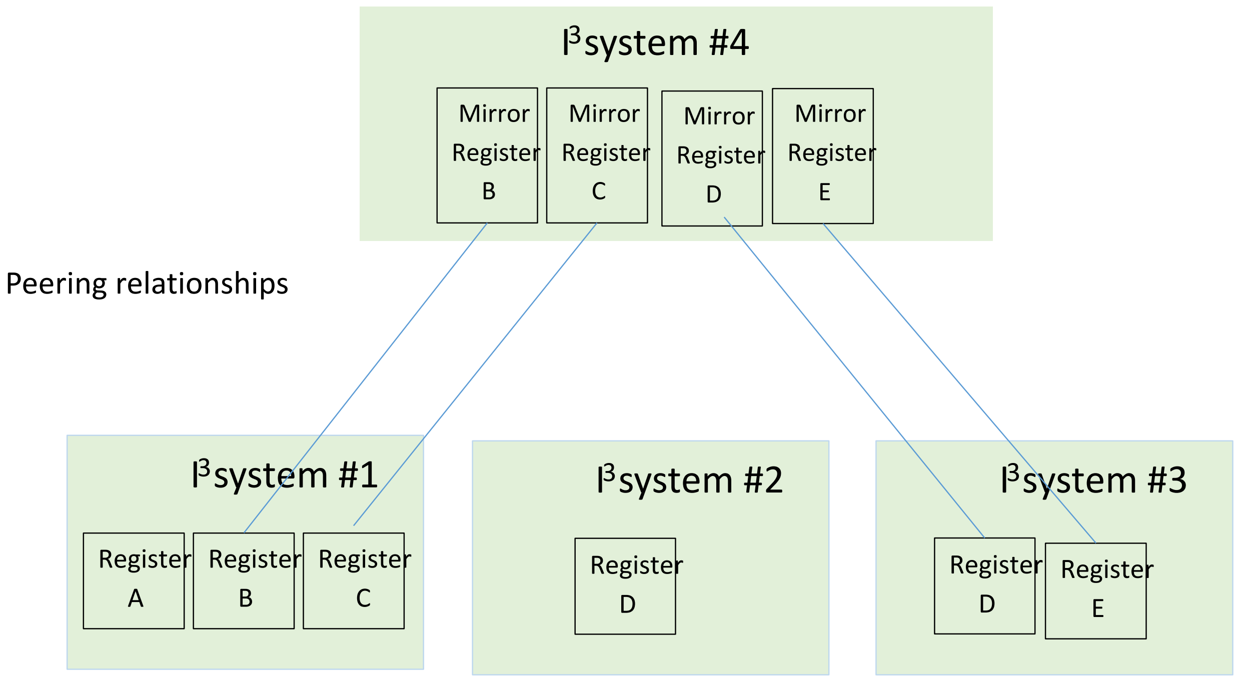
Database/registry: The database/registry contains descriptive information about beacons and sensors distributed throughout the I^3 system's span of control It also contains a collection of registers that reflects the current status of the sensors in the field. These registers are automatically updated as the sensors change state so the cache contains an accurate representation of the sensor status.

1. Sensor registers reflect status of sensors connected to the I^3 system. Registers can be binary (on/off), text, numeric, or up/down counters.
2. A single register may be visible to multiple applications based on the permissions given by the sensor manager. The database/registry will to track each application's permission independently and track how often each application accesses a specific sensor.
3. There may be one or more read registers, write registers, or read-write registers associated with each sensor. When a sensor is defined to the I^3 system, the registers associated with that sensor type are specified. When a sensor manager creates a new sensor, a new set of registers are added to the database/registry.
4. Each registers in the database/registry is associated with a price determined by the sensor manager. Different sensor manager may price the same sensor's registers to different price points. Read and write actions on a register may be priced differently.
5. The database/registry logs how often an application utilizes a register in the database/registry along with the cost of using that register. If the sensor manager changes the price for a specific register, all subsequent access requests will show the new price point in the log file.
6. Beacons are not sensors that report information to the database/registry. They are active devices that trigger sensor to report their status to the database/registry. Beacon locations are tracked in the registry for informational purposes.
7. Register data could be subjective data. For instance, if a sensor detects a person is in a bad mood (this could also be detected by body temperature of a wearable), the data would be transmitted to the database/registry and properly reflected in a register. The I^3 system should be agnostic to the specific interpretation of any data in the database/registry.
8. The sensor manager can create pseudo-registers in the database/registry that are not linked to a physical sensor. An application can read multiple registers which are tied to physical sensors and then write a logical composite value to a pseudo-register. When a pseudo- register changes value, it can result in the generation of an update notice which will be delivered to any subscribing applications. Pseudo registers can also be used to allow an application to 'logically' correct data that it decides is in error without losing the original data. Pseudo-registers can also be used to logically extend an information value associated with a physical register (for example, by adding personal information to your location data one can make the location data more valuable to the advertising industry and increase interest in your data).
9. Registers values may reflect a sensor's location (latitude, longitude, description, comments), status information (last updated, on/off-line, error/operational flags), read/write indicators, polling configuration (async reporting, periodically polling, ..)
10. When the application manager sets up a relationship between the App-API and the Database/registry, the relationship is a publish/subscribe messaging relationship. A single register in the database/registry can support have many application side publish/subscribe relationships. On the sensor side, there is a 1:1 publish/subscribe relationship with a physical sensor.
11. The system will maintain directories of applications and their data needs as well as sensors and their data capabilities. New entries in either directory will trigger an update message to the sensor or application owners as appropriate if the destination owners have asked for notification and the source owners have not asked for anonymity. There will also be a recommendation engine that allows those seeking new marketplace connections to search based on description, time added, by trust level, or by more complex recommendation criteria.

5.5. Peering Link

Peering Link: Multiple I^3 systems can be tied together. When two I^3 systems are linked either system can create a register in its local cache that is not tied to a sensor but instead reflects a register in the remote system; the local database/registry is updated as the remote register changes state.

1. The sensor manager can declare whether their sensors are visible to other I^3 systems.
2. A sensor manager on a remote I^3 system can create a mirror register on a local I^3 systems and set local permission parameters for that virtual sensor register.
3. A local I^3 system will send register update notices to any remote I^3 systems supporting mirrored registers.
4. The remote mirror cell can be configured to periodically poll the remote I^3 system periodically for validation.
5. All peer link requests are recorded in the log by the remote and local I^3 systems so both sites can determine if they are a net requestor or a net transmitter of data.
6. I^3 system administrators have to explicitly configure the peer-link connection to other I^3 system. Once a peer system relationship is established between two systems, the I^3 system administrator will be able to see all the registers maintained at the other peered I^3 system.
7. If the system administrator wants to enable its local application community to see a register maintained on a distant peered system, a mirrored register is created on the local machine.
8. When two disconnected I^3 systems want to establish a peered relationship, one I^3 system must apply to be connected to the other.
9. When peered I^3 systems link, they exchange contact information and establish a technical connection used to resolve operational problems and a billing connections used to manage the peered relationship. The assumption in a peered relationships is that a balance is achieved between two interaction I^3 systems. If the relationship goes out of balance, the I^3 systems that supported the majority of the service requests will fund the entity that satisfied the requests.
10. On a monthly basis, two peered systems will attempt to rationalize the relationship with each of its connected peers.
11. If the peering relationship cannot be rationalized between the two I^3 system administrations, either side of the peering link has the option to either a) suspend the peering relationship, or b) terminate the peering relationship.
12. When one I^3 system establishes a peer relationship with another I^3 system that has other peered relationships, the peering window can effectively see through the intermediate I^3 system. Thus mirrored registers can be daisy chained through the I^3 system. Considering the following diagram where system #1 has a peering relationship with system #2, system #2 has a peering relationship with system #3, and system #4 has a peering relationship with system #2 and system #3. Note that there is no peering relationship between system #1 and system #3 meaning that system #3 and system #1 do not have a fiscal relationship that allows mutual support of their IOT market place. Under this scenario, register B and C on system #1 is mirrored in a register on system #2. System #3 is mirroring register C from system #2. System #4 is mirroring register D, E, and C from systems #2 and #3. System #4 is not directly mirroring register C from system #1 because there is no fiscal relationship between system #1 and #4 which is needed if system #4 is sharing revenue with system #1.



13. The peering link effectively sets up a publish/subscribe relationship between two distinct Database/registry entities. A register can support multiple publish/subscribe relationships if multiple I^3 system are mirroring a common local register.

5.6. Sensor Manager

Sensor Manager: A sensor manager is responsible for configuring all the sensors in the registry that are owned/managed by a single person or entity. A single sensor manager may be responsible for multiple sensors. The sensor manager module is used to create, delete, and to set the permissions flags that provide application specific visibility of a specific sensor register.

1. Sensor managers can add, delete, or suspend, or activate sensors listed under a sensor account
2. Sensors can be defined in the I^3 database/registry in advance of their physical deployment and put on line once the sensor is put in place and made operational.
3. Sensor managers can generate a report showing the history of application requests for a sensor they own. Reports can be sent to the screen or a CVS/Text file that will be emailed to the sensor manager.
4. Sensor owners can generate a report for all sensors they own and which applications have access to those sensors, which applications have pending requests, and which applications have been refused access. Reports can be sent to the screen or a CVS/Text file that will be emailed to the sensor manager.
5. Sensor owners can generate a report for all sensors they own and when the sensor was last accessed and by which applications. Reports can be sent to the screen or a CVS/Text file that will be emailed to the sensor manager.
6. When a sensor manager receives a sensor access request from an application manager, they can grant the request, refuse the request, or delay responding to the request.

7. Sensor managers can configure the sensors for complete invisibility which causes the I^3 system to hide the sensor from all application managers. Sensor managers can also declare sensors as public which allows all applications to see the sensor data without requesting permission. Sensor data that is declared normal requires that the application manager request permission to access the registers.
8. Sensor managers can configure sensors are read, write or read/write sensors. If a sensor physically supports read/write attributes but the owner declares the sensor to be read-only, applications will not be able to write to the sensor.
9. When a new sensor is defined in the I^3 system, a new entry is made in the database/registry. Sensor definitions can include latitude, longitude, street address, nearby landmarks, floor, and room number as descriptor data. These fields can be manually entered by the sensor manager or uploaded from physical sensor if that capability is supported. The system will suggest incremental information based on operator entry of partial descriptors.
10. Sensor managers can grant or remove application rights to sensor data as desired (a request is not necessary)
11. System managers can also suspend logical groups of sensors, for example, all sensors in a building, all sensors owned by a specific user, or all sensors associated with a specific application.
12. When a sensor is initially configured, the sensor manager will set conditional flags that define whether or not a message is sent to an application with explicit access rights when a) a sensor is deleted, suspended, or an error condition is detected. Public sensors do not receive such alerts.
13. Sensor manager accounts are tracked to determine when they have last updated their password. Managers that have not updated their passwords are given a gentle reminder and if the password remains updated for a significant period of time, the account and all sensors associated with that user are locked.
14. A sensor manger can give administrative rights to other users so a single sensor can have multiple managers associated with a single sensor account. These secondary managers may or may not have rights to delete, suspend, or activate sensors. The original manager is the primary owner and the primary owner cannot be deleted until they have assigned primary rights to another manager.
15. Sensor managers can send a message to application owners advertising and informing the application owners when new (or updated) sensors become available. References to the application and sensor owner is masked in case the owner prefers to remain anonymous.
16. For each sensor account, the system tracks whether that account has generated a current credit or balance due since the last billing cycle. It also tracks the last billing cycle and whether the balance or credit has been cleared or is still open.
17. There is an automated process for sensor managers to recover from lost passwords that is difficult to be hacked by those with malicious intent.
18. A sensor can have a variable number of status reports that can be dynamically adjusted over the life of the sensor. For example, a sensor that monitors the packages in a truck and the status of each package may show a growing number of packages as the truck is being loaded and a declining number of packages as the packages are delivered.
19. It is possible to move sensor register data from one senor to another. For example if a delivery service tracks a piano to a home, the location of the piano is no longer associated with the delivery truck after delivery and instead it is associated with the delivery location (or tracked as an independent sensor).
20. Application managers must acknowledge the I^3 privacy policy as part of the application process. The application manager must also assert that their application information is valid and that they understand that they can be removed or suspended if the I^3 administration believes they are not acting in the best interest of the I^3 community.

21. When a sensor manager blocks permission from an application to access specific sensors, the application manager continues to have the ability to send messages to the blocked sensor manager.
22. Sensor managers can edit their account information if that information changes. For example, it their external email, billing information, phone number, or social network account references change, the sensor manager can self-edit this information.
23. Messages sent from/to the sensor manager are logged so they can later be tallied for billing purposes.
24. If the sensor manager changes the price of a sensor (a price increase or decrease), the sensor manager software should send a price change notice to any application managers who have visibility to that sensor. This includes sending a message to the peer link system if the register is mirrored on a remote system. There should also be an option that allows the same message to sensor managers that have potential to use the sensor allowing the sensor manager to 'advertise' a price reduction that might drive increased usage of the sensor.
25. Sensor managers should have the potential to rate and comment on the application managers and applications that use their data. A high rating of an application manager may encourage other sensor managers to approve them for use of their data. Application managers can comment on the ratings their receive (comment on but not change)
26. Device owners will be able to review an application developers data privacy policy and their ratings as other device owners rate them as a quality developers.
27. Device owners can set heuristics so the I^3 system auto-approves any request for data from pre-approved application owners. Pre approval may be based on a specific named developer, a developer from a specific domain, or a developer who's data policy and use meets certain criteria.
28. The device owners will be able to specify different price points for the same data based on different developer criteria. For example, an application developer from a nonprofit may be given a lower price point than a for-profit organization or developers may need to have a quality score > 90% to qualify for the lower price point.
29. Device owners can browse the application developers that have used their data in the past and can rate the quality of the experience with that developer. When rating the quality of the data, the I^3 system will provide a report card that allows the device owner to rate the data in different categories. The I^3 system will compile the rating factors into a single score which will be averaged with other device owner scores. The algorithm for converting the report card into a single numeric score will weigh the different criteria. The algorithm will not be made public and it can be modified as part of a system update if, over the course of the I^3 operational life, the weightings are changed.
30. Device owners will not be able to comment on specific data beyond the numeric ratings. The fact that there are no comments will prevent device owners from flaming IOT device owners and avoid fake comments. Device owners cannot rate developers unless they have had experience with that developer and there are time limits that prevent device owners from rating very old experiences.
31. The I^3 system will initially support a fixed number of price levels (e.g. 5). Device owners will specify a specific price point for a specific data set. Device owners may specify that the stated price point is auto-approved for developers that meet specific criteria and they may specify that requests for access will be auto-refused for developers with other characterizations. When the situation is unclear, the system will deliver the developers credentials with each request for access and the user can pick which price point they wish to sell their data for. In future releases, the I$_3$ system may allow the specification of an 'Other' price which allows the user to specific an ad-hoc and unique price point per developer.

32. Dynamic pricing where developers bid on access to data will not be supported but may be added at a later date

5.7. Sensor Driver

Sensor Driver: The sensor drivers interact with the remote sensors clients to determine (or set) register values. The sensor driver can accept autonomous indicators from the sensor client and changes the cache registers to reflect the status of the physical sensor. The sensor driver can also poll sensor clients if the client cannot create autonomous alerts or when a validation action is being performed,

1. Sensor driver modules should be monitored so that if the driver detects that the sensor has gone silent for an excessive period of time or if the sensor exceeds expected activity levels in a period of time, the application is flagged as potentially having gone rogue. (Silent sensors may be sent a wake-up packet before declaring an error condition).
2. All sensors will have a primary communications path to the central registry. That path may consist of a mix of cellular, WiFi, Bluetooth, dedicated connections and IP multiplexed connections. Critical sensors will also support a backup communications path allowing a fire alarm to automatically connect to the registry via a cellular link if the hardwire link should fail. The sensor drivers can be configured to automatically reconnect via the primary link when that link returns or it can be configured to require a manual operator reset.
3. Sensors can have dual paths to a single I^3 system or the dual paths can report to different I^3 systems to increase system reliability. When the two paths report to the same I^3 system, when one communications link fails, the other link is activated. Operator intervention is needed to flip between paths to avoid ping-ponging when the communications environment is unreliable. When the two paths report to different I^3 systems, one system is considered the primary I^3 system and the other operates as though it supports a mirrored set of device registers. When the communications path fails, the mirrored system becomes primary and the other system becomes the mirrored image.
4. After a communications link to the sensor recovers from a failure, once a link is restored after being suspended by the sensor manager or system administration, or after initialization, the sensor driver will query the sensor for its current state and update the I^3 system cache so its content reflects the current state of the sensor.
5. It is possible that the communications network that ties the registry to the sensors will become blocked during time of peak traffic or if the system is facing a DOS attack. Such conditions should be detected and properly reported. The sensor drivers should queue and commands in the hope the storm is temporary and passes. An administrator reset ultimately be needed; a soft reset maintains a queued commends whereas a hard reset would flush any queued commands. Any reset request may or may not require the logical communications link to be re-established.
6. If the system detects multiple devices reporting to the system with the same IP address, the system should assume there is a malicious attack being attempted and all impacted devices/applications should be suspended until the administrator can identify the source of the problem.
7. The system should support different levels of security between the devices and the I^3 system. At the highest level of security, the connection between the device and the I^3 system should be encrypted and dedicated. Dedicated communications may be achieved via a VPN or by using fixed IP addresses at the device and I^3 system. When the communications channel is first established a checksum is passed to ensure the far end of the link is running expected software. This may mean that the sensor driver has to allow an alternate checksum so the device software can be upgraded as long as the new checksum is manually configured on the I^3 system before the upgrade process is started.

The goal should be to make sure that one device cannot spoof the system and falsely impersonate another device.

8. Different kinds of sensors are supported by the I^3 system. Some Sensors will report status updates autonomously. Some sensors will require that the I^3 sensor driver periodically poll the sensor the latest status.
9. To simplify systems maintenance, configuration files/parameters should be used to allow a single sensor driver module to support a number of sensor types. Configuration files might allow specification of initialization commands, polling commands, structure of response messages, polling frequency, and alarm responses.
10. The sensor driver can be configured to support driver reaction to many detected error conditions. For example, if communications with the sensor is lost, the sensor driver may be configured to a) wait for communications to be reestablished, b) to zero the register associated with the sensor, c) to flag the register associated with the sensor as being in an error state
11. The sensor driver can accept autonomous indicators from intelligent sensor clients. A validation process is also supported to ensure the database/register data reflects the sensor's current state.
12. Sensor data from different manufacturers will be stored in a common cache system for easy reference by applications from different application developers. There is no common sensor format so the sensor driver will need to equalize the data to common format. To minimize the effort with the equalization process, the I^3 system designers will need to make a judgment call and structure the data within the I^3 system around a fledgling standard with potential for industry adoption.
13. While numeric/binary sensors create controllable data that can be understood, some sensors such as cameras produce large binary files and video devices produce streaming data that is not appropriate for storage in a structured cache. Unstructured data is treated differently from structured data in that the data is stored outside the I^3 system core (e.g. network attached storage or a cloud server) and the structured cache only contains a pointer to that data file/stream. It will be important to offload these types of files so that I^3 system process power is focused on providing near-real time support of the other sensor data with minimal processing impact from unstructured sensors.
14. The unstructured data structure that is used to support picture and video uploads from IOT devices can also be used to distribute unstructured data back to the IOT devices. For example, the unstructured data structure can be used to disseminate a new software release to remote IOT devices. Additionally, the unstructured data structure can be used to disseminate video or illustrations to electronic signs.
15. The sensor owner determines the level of I^3 connectivity and the level of connectivity may be less than full connectivity if the sensor so desires. For example, it the sensor device is a smartphone, the smartphone is capable of reporting location (lat, long), battery status, signal strength, and may other parameters (some being physical parameters and others being more amorphous such as mood of the user which might be determined by a screen slider bar). The sensor driver might support a reporting of all these parameters to the I^3 database/registry. When the sensor owner defines a new sensor device and specifies the driver associated with that device (protocol used between the device and the I^3 system), they can specific that certain parameters are not collected in the I^3 system. The sensor owner has the option, if the device supports it, to define a secondary path from the device to some other centralized intelligence (e.g. a cloud application) and this secondary path may support a different set of parameters. This puts the sensor owner in complete control as to what is reported to the I^3 system and what information is reported to other applications around the I^3 system.
16. The sensor driver managed encryption of commands going to the IOT device and decryption of messages coming back from the IOT device.

17. Different sensor drivers are used with different sensor protocols, to manage the sensor/controllers. The data is written to the database/registry. Because different sensor drivers are reading from different protocols and writing to a common database/registry, an application does not need to know device specific protocols and can read from multiple sensors that use different protocols.
18. The base I^3 system should support high usage sensor drivers when it is released and should allow system administrators to add new-custom sensor drivers to their local system. Some candidate sensor drivers for inclusion in the base package include CoAP (Constrained Application Protocol), Alljoyn, etc (see section on Leveraging Standards).
19. The system should support a function that can test any newly defined IOT system to see if access can be gained via known default admin/password or known back doors (Telenet, etc). Any such detected system represent a potential security threat and should ask for validation before the IOT device is accepted and the potential security issue should be flagged as a system alarm.
20. In the case where the sensors are generating data faster than I3 can process data and a queue begins to build in the driver, it is permissible for the driver to downshift message flows to better manage the incoming traffic. If this happens, the driver should generate an I3 message volume alert in order to alert operations staff.

5.8. App-User Admin

App-User Administration: The I^3 systems administrator uses this module to configure the application owners so they can use the application administration module to manage their applications.

1. Application developers/suppliers must 'apply' for access to the I^3 systems . The application process requires a potential developer submits contact name, billing information, address, phone numbers, email, etc in order to create an application account on the I^3 system.
2. The application process should have a nominal fee requirement to cover admin costs and to make sure the user is both serious and able to pay bills that I^3 might send to them
3. A single application account may be associated with multiple applications.
4. The system administrator can create a report showing the applications associated with all accounts and usage for each application within that account.
5. The system administrator can generate a report showing accounts by state (pending approval, approved and in good standing, suspended, etc).
6. It is possible for there to be more than one application manager associated with a single user account; the application manager or the system administrator can create a report showing the managers associated with a specific account.
7. The application manager approval process is normally expected to be a totally automated process. This implies an automated means to validate external email and billing information. The I^3 systems administrator is manually alerted if the automated process cannot self-validate the request so that the system administrator can step in and take manual action.
8. Sensors can be placed in a box and that box can be placed in a crate. The crates can then be placed on a truck. Thus sensors can be related to other sensors. Each sensor record will be tagged so it is clear whether the sensor has child relationships or whether the sensor record is a child record. These fields to not need to be filled but they are there for the convenience of the applications. The I^3 system will not manage these fields directly but will assume an application layer process will update these fields when the relationships are made or broken.
9. Before a new device owner is accepted by the system, it will send a validation message to the user's email address to validate the user can indeed be contracted via that email address. When a user

adds a new device to the I3 system, it will send a validation email to the device owner to make sure that user did indeed intend to add the device.

5.9. Database/System Admin

Database/System Admin: The I^3 systems administrator can use this module to manually manipulate the cache registers to adjust the configuration of the registry and to manage I^3 upgrades.

1. I^3 system administrators are tracked to determine when they have last updated their password. Administrators that have not updated their passwords are given a reminder and if the password remains updated for a significant period of time, the other I^{3}system administrator are notified and asked to resolve the situation.
2. The I^3 system supports multiple system administrators. System administrators can add, edit, or delete parameters associated with the other system administrators.
3. Root system administrators are not normally used for any system administration. Multiple root administrators can be supported by the system allowing a different root administrator for each operational shift to be defined.
4. System administration log in and log out actions (including attempted actions) are broadcast to all other system administrators.
5. System administrators can shut down any application. They can also shut down logical groups of applications, for example, all applications associated with a specific user.
6. System administrators can backup and recover the database/registry, log files, or the complete I^3 system.
7. Individual software modules can be upgraded and restarted to manage bug fixes.
8. There is a defined process for system administrators, sensor managers, and application managers to log bug reports and feature enhancement requests. Such requests can be annotated as they are worked through closure. The system administrator can create a report of open requests, closed requests, and inactive requests (requests still open but showing no activity).
9. Every system administration action is logged by the I^3 system.
10. A billing process can be scheduled to run on a periodic basis. The billing process uses the log files to create application usage reports and sensor usage reports. Application and sensor usage reports are reconciled to create invoices which can be physically or electronically mailed to account holders. Invoice amounts can also be electronically submitted to banks (credit cards) or create other charges. As outstanding charges are cleared, credits can then be electronically issues to sensor manager accounts as appropriate.
11. The billing process should be manageable through Paypal or some other operational entity service. Use of an external service to manage distribution of incentive payments reduces the complexity of native I^3 system software AND provides an externally auditable means of validating the status of payments. Paypal software validated banking information on behalf of the I^3 system.
12. There are permission levels for each application desiring to access a sensor register. At the record level, different applications can be given different permissions. For example, one application may have access to the IOT device's temperature register but another application many not have access to this register. Similar devices can have different permissions for the same application.
13. If the server owner wants to preload a large number of sensors as a batch file, they would create a file off-line and send the file to the I^3 system administrator. The system administrator would validate the file before importing it to the cache.
14. The I^3 system administrator should be able to configure the node to reflect local communications conditions. For example, if the administrator is connected to the network via an unlimited data

plan, perhaps frequent hear-beat checkins are appropriate to ensure operational integrity. If the network service is priced based on bandwidth or number of messages, perhaps other parameters might be more appropriate.

15. The I^3 system configuration should specify if there are requirements for battery backup for continued operation in the face of a power failure. The system should also be able to cache some level of traffic to ensure graceful operation in spite of operational faults. Also the system should be able to restore itself automatically (without human intervention) if a shutdown is unavoidable.

5.10. Report Admin

Report Admin: The I^3 systems administrator can use this module to create, schedule, print, or save system reports.

1. Law enforcement officials may present the I^3 systems administrator with a CLEA request. CLEA requests are legal requests, signed off by a judge, asking for histories of application and/or sensor logs. The administrator should be able to generate and provide these logs to law enforcement officials. The creation of such reports should be logged with annotations that allow the operator to comment on the source and validity of the request.
2. The I^3 system administrator can generate multiple reports that are printed, displayed on a screen, or saved to a TXT or CVS file
3. System administrator reports are dated and can include user configured page headers and footers (text, page numbers, date, …)
4. System administrator reports can be generated that show who owns a specific sensor, a specified list of sensors, or of all sensors
5. Sensors may be independent sensors or linked sensors. For example a flow censor may be put on each commode, another one on the floor feeder pipe, another one on the building feeder. Administrators should be able to step-wise query each sensor along a linked path.
6. The system should allow many different kinds of linked sensors. For example, plumbing might be one linked sensor system, air conditioning another, electricity yet another.
7. System administrator reports can generated that show the permissions of one or more sensors
8. System administrator reports can generated that show what sensors are owned by a specific sensor manager
9. System administrator reports can generated that show what sensors are near a specific location.
10. System administrator reports can generated that show what sensors can be accessed by a specific application or a set of applications owned by an application manager
11. System administrator reports can generated that list sensors by type or by sensor installation date
12. System administrator reports can generated that list sensors by sensor state (active, suspended, error, etc)
13. System administrator reports can generated that list applications that have access to a specific sensor.
14. On a scheduled basis, the I^3 system should report (via email) to a central organization a count of the total number of sensors it is managing locally, the number of sensors it has visibility to via the peering link, the number of applications it supports, the number of sensor managers, the number of application managers, the number of system administration accounts, and the latest release levels of major software modules. These reports should not identify specific uses but should be used to aid in identification of potential security flaws and to track operational statistics.
15. When there is reason to suspect a malicious users is attempting to breach the I^3 security systems, a detailed report should be auto-generated and sent to the central management authority so it can

watch for threat trends. On a scheduled basis (e.g. monthly) the I^3 system should report (via email) to a central organization summary information about any attempts to breach the system security mechanism.

16. On a scheduled basis, the I^3 system administration system will scan the usage log to collect aggregate usage statistics that can be used to create a billing report. The billing report will create invoice records for all prepaid and postpaid accounts. Sequential systems will then deduct usage fees from prepaid accounts and lock any accounts that have gone into arrears. Invoices/bills will be created and issued for any postpaid accounts and any postpaid accounts that have outstanding balances at that point in time will be administratively locked. It is important that the system support both prepaid and postpaid user accounts. Similar processes will be run for linked peer systems; if a peered I^3 system is in arrears, all mirrored registers and any transactions with the peered account will be administratively locked until the matter is resolved or the lock is administratively removed.
17. Administrative locking of an account is considered a high priority alarm for the local I^3 administration as these locks will interfere with ongoing processing of messages.
18. Message traffic (volumes) from each IOT device and from applications are tracked and if volumes exceed expectations, alerts are raised as this may be an indicator of an active cyber-attack against one or more of the IOT Devices (or applications)

5.11. Sensor-User Admin

Sensor-User Admin: The I^3 systems administrator uses this module to add sensor owners to the system so that they can begin defining new sensors in the registry.

1. Sensor managers must 'apply' for access to the I^3 system. The application process requires a potential sensor manager submits contact name, billing information, address, phone numbers, email, etc in order to create an account on the I^3 system. The application process also allows users to specify their LinkedIn, Facebook Snapchat, Instragram, and Twitter accounts where they are willing to accept messages from applications expressing interest in their sensor data.
2. The application process should allow for a nominal fee requirement to cover admin costs and to make sure the user is both serious and able to pay bills that I^3 might send to them. Currently the sensor manager fee is envisioned to be zero (free)
3. Once a sensor manger is approved, they may add other sensor mangers to the same account.
4. A single sensor manager may be associated with multiple sensors.
5. The system administrator can create a report showing all sensor managers associated with a sensor accounts.
6. A sensor manager can create a report showing the applications that have been using the sensors they manage and the usage level of each application.
7. The system administrator can generate a report showing sensor manager accounts by state (pending approval, approved and in good standing, suspended, etc).
8. The sensor manager approval process is normally expected to be a totally automated process. This implies an automated means to validate external email and billing information. The I^3 systems administrator is manually alerted if the automated process cannot self-validate the request so that the system administrator can step in and take manual action.
9. When a sensor manager gives an application visibility to a particular register in the database/registry, they do NOT have permission to see the historical information for that register (this data could be seen from the system's log). If the system manager wants to provide an application a view of the sensor history, they can run a history report for the sensor(s) they own and can send that report to the application developers. (Sending the report through the I^3 system will

mask the sensor manager's id so the application developer has no direct access to the sensor manager).

10. Sensor managers must acknowledge the I^3 privacy policy as part of the application process. The sensor manager must also assert that their application information is valid and that they understand that they can be removed or suspended if the I^3 administration believes they are not acting in the best interest of the I^3 community.
11. When a sensor manager blocks permission from an application to access specific sensors, the sensor manager continues to have the ability to send messages to the blocked application manager.
12. As a part of the registration process, users will be asked their birth data. Device owners under the age of 18 will not be able to directly register their devices with the system. A device owner can specify subaccounts under their account which they accept responsibility for. Sub account owners can register IOT devices on the system as though they were main account owner. This allows device owners to allow their children to manage their own devices with parental approval.
13. Device owners should have a manual means to validate that the data collected from the sensors they own is being properly recorded by the I^3 system.
14. Beacons detect devices near the beacon and beacons can cause identification messages to go the I^3 system. The I^3 system should be able to respond to such inquiries whether the device is a recognized device or a new and unregistered device.
15. A user may be associated with multiple devices (e.g. a tablet, a smartphone). Each device may also be associated with other devices such as a fitbit or a watch via Bluetooth. Thus many devices might be associated with a single user and the devices may connect to the I^3 system through different devices and connection processes throughout the life of the device. If the user changes permission for one device, the permissions should change for all devices owned by that user.
16. A single device could be shared by multiple users. For example, one family might share a single cellphone among multiple family members. The I^3 system will allow a device owner's ability to 'give' their device to another owner. When a device is controlled by a secondary user, the permissions of the secondary user apply to the device and any incentives flow to the secondary user. The device owner's I^3 permissions will be maintained so that when the device is given back, the original permissions are restored. When a user gives control to another there should be an option that requires an explicit release back to the original owner OR there could be a timed option that allows temporary control to revert automatically after a defined period of time.
17. Applications can write permission settings to user records. Application specific settings will be deleted from user's record when the user deletes information flow from their devices to the application. Application specific settings will be deleted from all user records when the application is deleted. Users cannot see or manipulate any application specific settings associated with their I^3 account.

18.

5.12. Sensor Client.

Sensor Client: Sensors clients are device specific software modules that must be loaded to the sensor device. These modules report sensor specific information to the sensor driver in the I^3 system.

1. Devices that support multiple sensors should report all sensor conditions to the sensor driver when polled in order to minimize the number of communications messages.

2. Communications messages from the sensor client to the sensor driver should attempt to follow a preferred industry standard whenever possible.
3. The I^3 systems should support multiple IOT messaging protocols (google, apple, etc)
4. If the sensor client detects that it has lost communications with the I^3 system, it should create a local alert. When communications is restored, it should notify the I^3 system of the temporary outage and transmit a complete sensor update report.
5. Sensor clients should transmit a sensor update when there is a detected change in sensor data.
6. Sensor clients should respond to a polling request from the sensor driver with a complete sensor update report.
7. When a new sensor is defined, the I^3 system will suggest default sensor operational parameters for all applications seeking to access that sensor. The device owner can change the suggested defaults so the default application permission reflects the owners desired permissions. When an application requests access to a sensor these adjusted defaults are suggested as a part of the permission request. Administrators can accept the suggested default or can change the permission on an application by application basis. The default permission parameters will allow a sensor to reject all application requests for access from characterized applications, will allow a sensor to accept requests from characterized applications, and will allow the device owner to hide a sensor from applications. If a sensor is masked from external visibility, the device owner can still grant manual application to their IOT devices.