IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF ILLINOIS
EASTERN DIVISION

ROKIOT USA, LLC,	§	
	§	
Plaintiff	§	
	§	
v.	§	**CIVIL ACTION NO. _____**
	§	
BOSCH SOFTWARE INNOVATIONS CORP.; ROBERT BOSCH GmbH; ROBERT BOSCH LLC; BOSCH CONNECTED DEVICES AND SOLUTIONS GmbH; and BOSCH SOFTWARE INNOVATIONS GmbH	§	**JURY TRIAL REQUESTED**
	§	
Defendants	§	

ROKIOT USA, LLC'S ORIGINAL COMPLAINT
FOR PATENT INFRINGEMENT

Plaintiff Rokiot USA, LLC files this suit against the Bosch Defendants for infringement of U.S. Patent Nos. 7,895,257 and 8,631,063.

The Bosch Defendants infringe the asserted patents by providing the Bosch Internet-of-Things ("IoT") platform used to connect, monitor, and command devices in smart homes and buildings and industrial applications including manufacturing, energy, and mobility.

THE PARTIES

1. Plaintiff and patent owner Rokiot USA, LLC ("Rokiot USA") is a Delaware limited liability company headquartered in Winters, Texas.

2. Bosch Software Innovations Corp. is an Illinois corporation with an established place of business at 161 N. Clark Street #3550, Chicago, IL 60601. Bosch Software Innovations Corp. may be served through its Illinois registered agent, Illinois Corporation Service Company, 801 Adlai Stevenson Drive, Springfield, IL 62703.

3. Robert Bosch LLC is a Delaware limited liability company headquartered in

Chicago, Illinois. Robert Bosch LLC may be served through its Illinois registered agent, Illinois Corporation Service Company, 801 Adlai Stevenson Drive, Springfield, IL 62703.

4. Bosch Software Innovations GmbH is a foreign entity based in Germany and located at Schöneberger Ufer 89-91, 10785 Berlin. Bosch Software Innovations GmbH may be served through its agents, Robert Bosch LLC and Bosch Software Innovations Corporation, and their registered agent for service of process, Illinois Corporation Service Company, 801 Adlai Stevenson Drive, Springfield, IL 62703.

5. Bosch Software Innovations (www.bosch-si.com) is focused on IoT software and system design and development and responsible for the Bosch IoT platform and IoT Suite.

6. Bosch Connected Devices and Solutions GmbH is a German company with a domestic location at 161 North Clark Street, Suite 3550, Chicago, IL 60601. Bosch Connected Devices and Solutions (www.bosch-connectivity.com) provides IoT solutions including sensors, actuators, data transfer and integration with cloud platforms such as the Bosch IoT cloud. Bosch Connected Devices and Solutions GmbH may be served at its Chicago location or through its agent, Robert Bosch LLC, and its registered agent for service of process, Illinois Corporation Service Company, 801 Adlai Stevenson Drive, Springfield, IL 62703.

7. Robert Bosch GmbH is a German company with a domestic location at 161 North Clark Street, Suite 3550, Chicago, IL 60601. Robert Bosch GmbH may be served through its agent, Robert Bosch LLC, via its registered agent for service of process, Illinois Corporation Service Company, 801 Adlai Stevenson Drive, Springfield, IL 62703.

8. The Bosch defendants are related, wholly owned subsidiaries of Robert Bosch LLC or the ultimate corporate parent, Robert Bosch GmbH.

9. Defendants make, use, sell, advertise, distribute, promote, lease, license, and/or

import the Bosch IoT software suite, hardware platform, IoT cloud services, Bosch IoT sensors and actuators, and IoT solutions.

10. Defendants provide the following description of their IoT solutions:

> *As a leading IoT company, Bosch offers innovative solutions for smart homes, smart cities, connected mobility, and connected manufacturing. It uses its expertise in sensor technology, software, and services, as well as its own IoT cloud, to offer its customers connected, cross-domain solutions from a single source.*

JURISDICTION AND VENUE

11. This is a patent suit brought under the United States Patent Act, namely 35 U.S.C. §§ 271, 281, 283, 284, and 285, among other laws. This Court has subject-matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and 1338.

12. Defendants each are subject to this Court's specific and general personal jurisdiction due at least to their regular and substantial business in Illinois and this district, Defendants' purposefully availing themselves to the privileges of conducting business in this state and the Northern District of Illinois; and Defendants' activities in Illinois and this district from which Rokiot USA's causes of action directly arise. Defendants make, use, advertise and sell, and contract for the provision of the accused products and services in Illinois and this district.

13. Venue is proper in this judicial district pursuant to 28 U.S.C. § 1400(b). Defendants reside in this district, have a regular and established place of business in Chicago, and have committed acts of infringement in this judicial district.

BACKGROUND

A. ROKIOT USA

14. Rokiot USA is controlled by Abdelsalam ("Sumi") Helal, Ph.D. Dr. Helal is a recognized pioneer of the Internet of Things. A professor at the Computer and Information Science and Engineering Department at the University of Florida, Dr. Helal is a recognized leader in mobile

and pervasive computing. He has made significant contributions to the advancement of smart spaces, assistive technology, and development of IoT-based enabling technologies for the elderly and hearing/visually impaired.

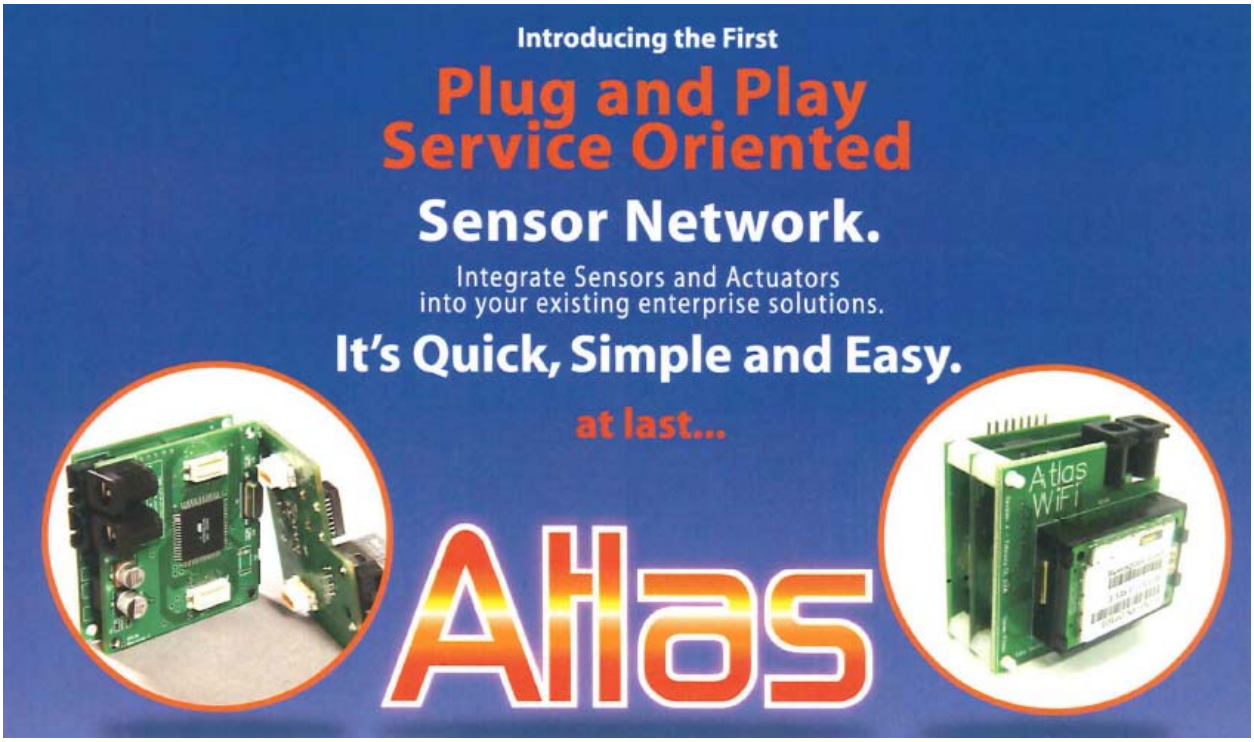
15. A prolific scholar, Dr. Helal currently serves as Editor-in-Chief of IEEE Computer, the flagship magazine of the IEEE Computer Society.

16. At the University of Florida, Dr. Helal co-founded and directed the Gator Tech Smart House, an experimental facility for applied research, development, and validation of IoT technology in the domains of elder care and digital health.

17. The Gator Tech Smart House garnered widespread attention as a showcase of assistive technology developed by Dr. Helal and his co-inventors that was enabled by the inventions in the asserted patents, including a smart mailbox that sensed when mail arrived, window blinds that automatically controlled light and privacy, a smart bed that monitored sleep patterns, and a floor that tracked occupants level of activity and that could detect falls.

18. In 2006, Dr. Helal co-founded Pervasa, Inc., a University of Florida technology startup focused on commercializing the modular architecture successfully proven in the Gator Tech Smart House. Pervasa's Atlas platform and middleware enabled a plug-and-play framework to

sensor/actuator networks. Atlas devices appeared in the framework automatically as they were added and powered on. Atlas applications included smart homes, healthcare, and asset tracking.



19. In 2007, Pervasa won the Silver "Best of Sensor Expo Award" beating University of California Berkeley's Crossbow, which took the Bronze.

20. The Patent Office recognized the pioneering efforts by co-inventors King, Bose, Pickles, Russo, Ploeg, Zabadani, Kouche and Helal by awarding patents covering their novel IoT platform technology.

21. The University of Florida assigned the patents-in-suit to the original inventors in 2013, and they assigned their interests to the Plaintiff in 2017.

B. THE ROKIOT USA PATENTS

22. Rokiot USA is the owner, by assignment, of U.S. Patent Nos. 7,895,257 (the "'257 Patent") and 8,631,063 (the "'063 Patent") both titled "Modular Platform Enabling Heterogeneous Devices, Sensors, and Actuators to Integrate Automatically into Heterogeneous Networks."

23. A true and correct copy of the ’257 Patent is attached as Exhibit 1.

24. A true and correct copy of the ’063 Patent is attached as Exhibit 2.

25. As the sole owner of the ’257 and ’063 Patents (the “asserted patents”), Rokiot USA holds all substantial rights in and under the patents, including the right to grant sublicenses, exclude others, and to enforce, sue, and recover damages for past, present, and future infringement.

26. The United States Patent Office granted the ’257 Patent on February 22, 2011, and granted the ’063 Patent on January 14, 2014.

27. The ’257 and ’063 Patents are valid, enforceable, and were duly issued in full compliance with Title 35 of the United States Code.

28. The asserted patents generally are directed to a platform for integrating heterogeneous devices (i.e., sensors and actuators) and applications. In pervasive computing spaces such as a smart home, industrial, or manufacturing facility, actuators and sensors are deployed to monitor an environment, condition or state, send and receive data, and respond to control signals typically sent remotely over a wireless network.

29. The specification describes applicability of the patented technology to remotely controlled appliances, lights, doors, coffee machines, temperature controls, home theater systems, communication systems, security cameras, surveillance equipment, and the like. ’063 at 1:21-27.

30. Home automation systems, or “smart homes,” provide the convenience of controlling systems in a home or business from a central or remote location. ’063 at 1:41-45.

31. The ’257 and ’063 asserted patents address problems related to deploying such systems and integrating new devices.

32. Previously, when a new component was added, developers had to undertake the tedious task of learning and accounting for characteristics related to operating, interfacing,

communicating, and configuring the device. '063 at 2:30-33. New devices (i.e., sensor or actuators) were physically integrated, configured, and tested within an overall system. Computer applications for the new device had to be written with knowledge of the resources assigned to connect the device, signals required to query the control device, and the meaning of any signals return to a centralized system from the device. '063 at 2:35-38. Any changes in the deployment device required repeating the configuration process. Moreover, once application software was developed, changes may require modification along the entire communication path from device to data repository to ensure interoperability.

33. In short, previous systems lacked a refined way of achieving "modularity" where the devices could be added to a system without configuration overhead. In addition, previous systems that attempted to provide scalable solutions focused heavily on sensors (e.g., temperature sensors, pressure sensors) without sufficient regard for scaled integration of actuators.

34. The claimed subject matter of the asserted patents describes scalable, reliable, and secure data ingestion (e.g., from sensors) and command and control messaging (e.g., with actuators).

35. Accordingly, the asserted patents relate to platforms that provide a uniform interface to any type of sensor, actuator, or connected device. '063 at 5:20-23.

36. By providing the capability to represent connected devices automatically as software services to programmers and users, a larger number of devices may be supported. '063 at 4:8-30.

37. The inventors recognized a "need for a modular, service-oriented sensor and actuator platform specifically designed to support the development of scalable pervasive computing spaces." '063 at 4:28-30.

38. They further recognized that "development of smart spaces is very different in goals and requirements from the typical sensor network application." '063 at 5:21-22.

39. Describing generally an embodiment of the claimed subject matter, the inventors noted that "manual integration of sensors and actuators is preferably replaced by a scalable, plug-and-play mechanism . . . [such that] the smart space is preferably assembled programmatically by software developers instead of hard-wired by engineers and system integrators . . . allow[ing] for cost-effective development, enable[ing] extensibility, and simplif[ying] change management." '063 at 5:21-29.

40. In a preferred embodiment described in the specification, "a pervasive space exists as both a runtime environment and a software library." '063 at 5:32-3.

41. Benefits provided by the claimed inventions include: (i) interchangeability of various sensors and actuators without the need for cumbersome reworking of the platform and/or associated software; (ii) enabling users of the platform to control, and interact with, the sensors and actuators in a higher level language without the need to program at the hardware level of the devices; and (iii) interchangeability of the hardware modules (e.g., one communication module can be interchanged with another to allow for the use of different networking technologies without reworking of other modules). '063 at 5:52-63.

42. Components of a disclosed embodiment include a hardware platform, a middleware module, and one or more "software services" that represents an active object. In normal operation, the hardware platform communicates with at least two active objects, where at least one active object is an actuator and one active object is a sensor.

43. Those skilled in the art understand that the Internet-of-Things generally categorizes "Things" as sensors or actuators.

44. Sensors provide information about a particular domain, supplying data to the system about the current state of the space. In general, sensors create data, usually by providing measurements or telemetry.

45. Actuators are active objects that alter a space. Typically, actuators accept commands to perform certain functions.

46. The specification describes sensors and actuators as "the foundations of a pervasive space, as they provide the means for gathering information about the state of the space and for controlling devices that can modify the state of the space." '063 at 6:47-53.

47. In one embodiment, the platform connects numerous and heterogeneous sensors and actuators to the services and applications that monitor and control the space. As shown in Figure 7 (below), information flows through nodes 54 and middleware 10. Each node is given a unique identifier, and when a node comes online it sends its identification or other data 96 to the middleware 10. When this is acknowledged, it sends the driver bundle 98 for the attached devices 16, 18, and, after this, the application function loops, handling any incoming network packets, periodically sampling the sensors 16, sending signals to actuators 18, transmitting sensor data, and sleeping.

48. The figure below is a reproduction of Figure 7 from the asserted patents.

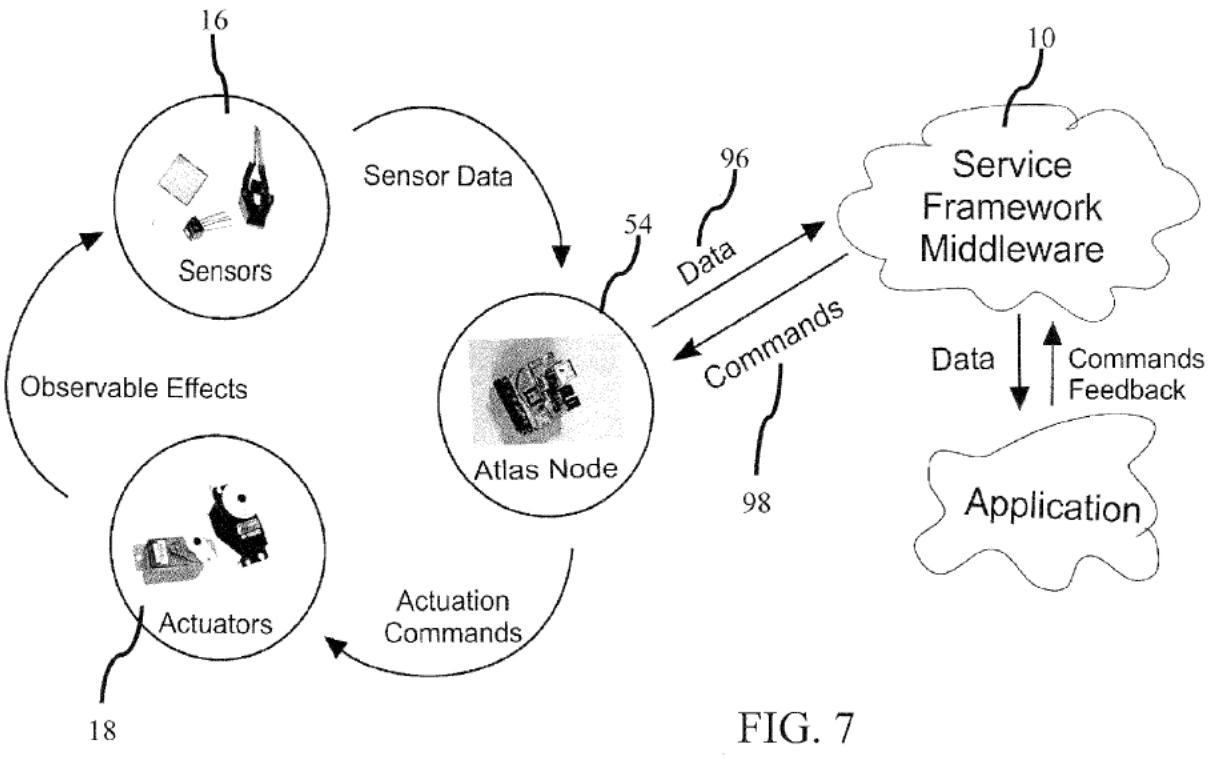


FIG. 7

49. As illustrated in Figure 7, commands flowing to actuators and data flowing from sensors.

50. As expressed in claim 1 of the ’063 Patent, an embodiment of the claimed subject matter includes a middleware module that generates one or more software services for each of at least two active objects, receives commands from one or more applications written in a high-level language via the software services, converts commands into low-level commands (capable of controlling the active objects that can be understood by the active objects), and transmits low-level commands to active objects via a hardware platform.

51. Further according to ’063 claim 1, a hardware platform receives raw data from at least one active object and passes it to a middleware module. In turn, the middleware module converts raw data into useable data and passes it to the software service for that active object.

52. The specification describes higher-level application software receiving usable data

from a particular active object's software service.

53. According to the specification, interchangeability of sensors and actuators without the need for cumbersome reworking of the platform or associated software is a benefit of the claimed subject matter. '063 at 5:52-63. In an exemplary embodiment, hardware platforms, connected devices, and associated software services appear as a single, homogeneous environment even if the actual environment comprises heterogeneous networks or devices. '063 at 7:36-40.

54. This functionality enables users of the platform to control and interact with "Things" in a higher level language, obviating the need to program each Thing at the device's hardware level.

55. Another described advantage lies in the interchangeability of hardware modules. For example, one communication module can be interchanged with another to allow for the use of one networking technology or another without reworking other modules. '063 at 5:52-63.

56. Programming an intelligent space involves three activities according to an embodiment of the asserted patents: (i) context engineering involves interpreting sensory data; (ii) software engineering includes describing various software component's behavior; and (iii) associating behavior with context includes defining which pieces of software can execute in a particular context and which pieces of the system should invoke upon a contextual change. '063 at 6:14-21. As discussed below, Defendants' IoT system adheres to this same development framework.

57. In the asserted patents, Figure 1 is a schematic view of one embodiment of a middleware architecture for programmable pervasive spaces built using the platform of the claimed subject matter:

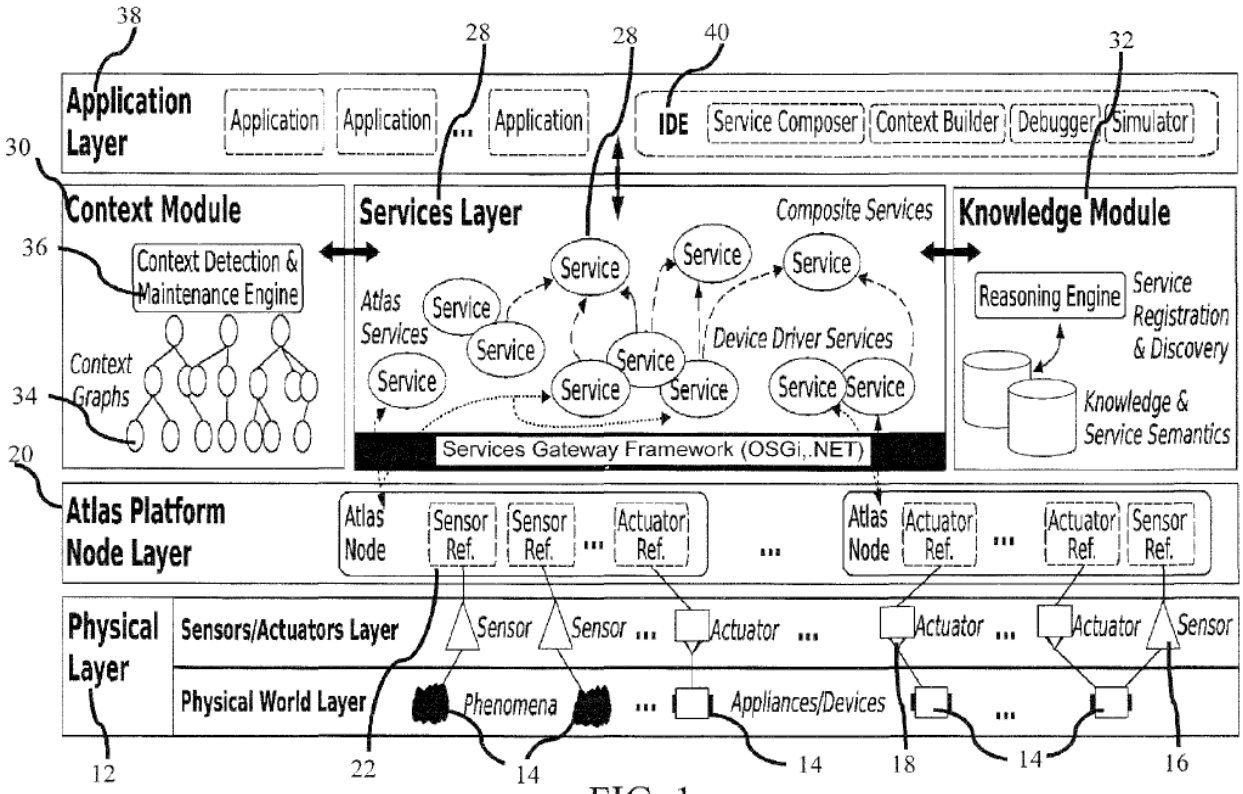


FIG. 1

58. Referring to Figure 1, the specification describes an embodiment of a middleware architecture 10. '063 at 7:48-8:38.

59. In the physical layer, various phenomena, appliances, and devices are active objects captured through actuators and sensors into the smart space for observation and control.

60. Above the physical layer in the representation of Figure 1, the Platform Node Layer contains sensor and actuator platform notes that automatically integrate the sensors and actuators and hence their respective active objects and export their service representations to the layers above.

61. The platform layer represents sensors/actuators in the physical environment as one or more software services that can be programmed or composed into other services. Thus, the physical world is represented as a set of software services to programmers.

62. Above the platform layer in Figure 1 is the service layer, which holds the registry

of software service representations of the sensors and actuators. In one embodiment, the service layer runs on a centralized server and contains a context management module and knowledge representation and storage module. These provide remote management functionality including registration, context creation, and device management generally.

63. The specification describes an exemplary embodiment comprising an application layer sitting atop the platform layer. The application layer includes a runtime environment that provides access to a software library of sensors, actuators, and other services.

64. In a disclosed embodiment, the application layer also includes actual IoT applications and composed services that monitor and control elements of the pervasive space.

65. In another disclosed embodiment, the platform represents any attached object in an IoT space as a Java program and the object is represented as an OSGi service bundle. The middleware framework in such an embodiment is shown in Figure 8.

66. With reference to the embodiment of Figure 8, the specification describes registering and hosting software services in an industry-standard service framework such as the Open Services Gateway initiative (OSGi) standard specifications that are governed by the OSGi Alliance.

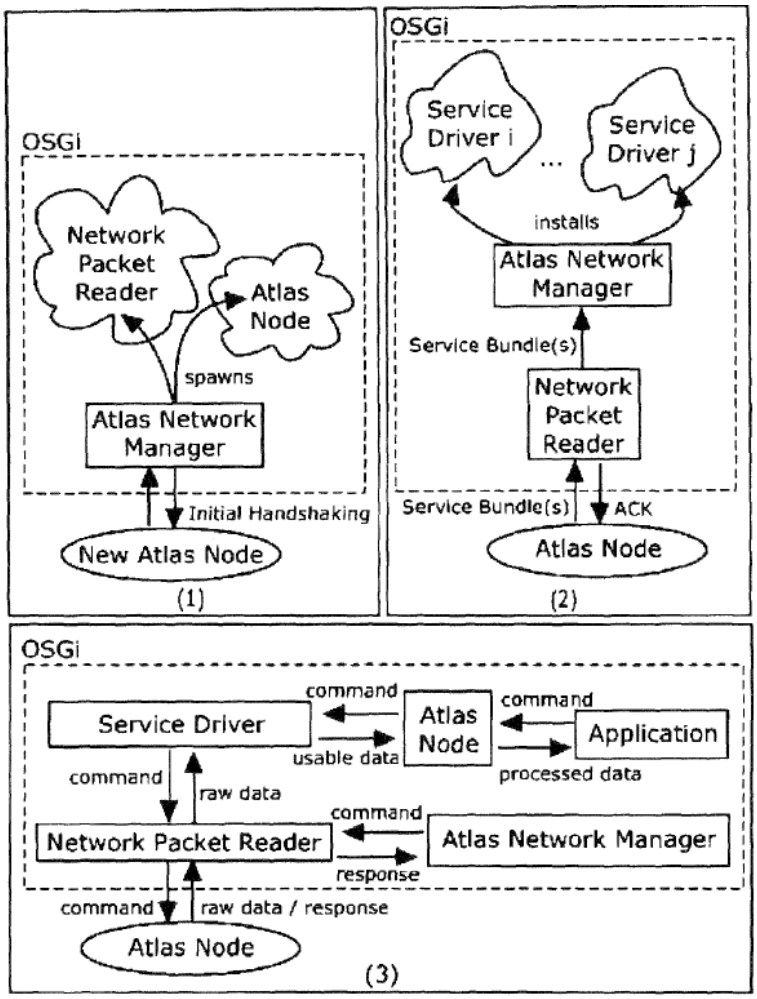


FIG. 8

67. The OSGi Alliance develops and promotes open specifications that enable the modular assembly of software built with Java technology.

68. In an exemplary embodiment, a driver represents each of the hardware sensors, actuators, or other devices connected to the platform as one or more software services on a software interface such as the middleware. These services are then made available to client programs or users through the middleware, for example, by applications or other services. '063 at 13:53-58.

69. Thus, in such an embodiment, each software service, regardless of the type of associated device, complies with a standard interface such as the middleware and can be

discovered and accessed through this interface by applications and other services using standard mechanisms such as those provided by the standards-based service framework.

70. Defendants have taken this approach. According to a representative from Bosch, OSGi and IoT fit together because "it's really all about interoperability" and with many different devices and technologies, OSGi provides a common layer that enables application development apart from the actual devices. Bosch uses OSGi in smart home, mobility and transportation, and industrial IoT applications. *See* Bosch Software Innovations Video *available at* https://www.osgi.org/business/success-stories/.

C. BOSCH IOT PLATFORM

71. Bosch imports, makes, uses, sells, offers for sale, licenses, tests, develops, and/or distributes Bosch IoT platforms and systems that embody the asserted claims of the '063 and '257 patents.

72. Bosch's IoT platform and systems include the Bosch IoT Suite, servers and cloud-server infrastructure, IoT gateways, middleware, software, hardware, drivers, interfaces, actuators and sensors. Defendants provide these components to facilitate communication between field devices and software applications in various infringing combinations for uses such as those described herein. For convenience, these accused instrumentalities may be referred to herein as the Bosch IoT System or "BIOTS."

73. At a high level, Bosch describes the accused instrumentalities as a triad: sensors, software, and services. (Bosch ConnectedWorld Blog, September 3, 2016, Exhibit 3)

74. Defendants provide sensors and actuators and integrate Bosch and third-party sensors and actuators in the accused instrumentalities.

75. The following excerpt from a recent press release describes the importance of

BCDS's IoT sensor solutions (Exhibit 4):

Press release

BOSCH

CES 2017, Booth No. 14128

Bosch Connected Devices and Solutions showcases sensor solutions for connected mobility, Industry 4.0 and logistics

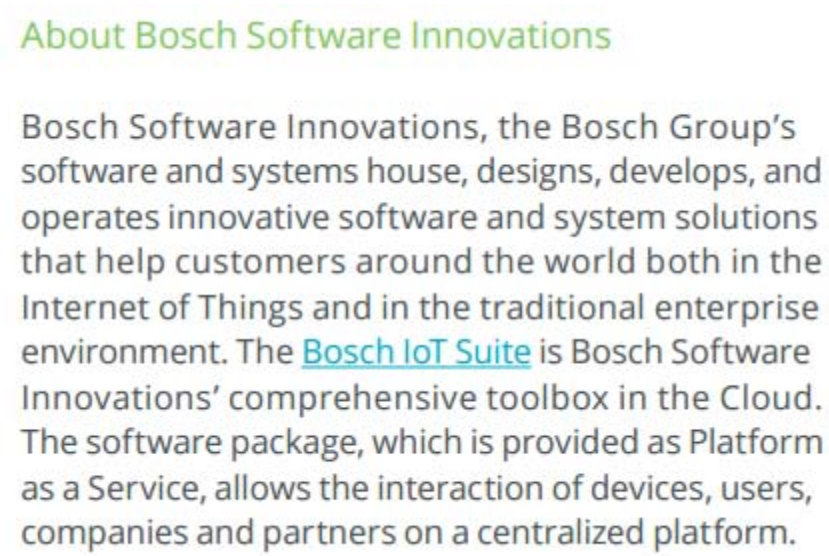
Assisting drivers, improving accountability, and reducing IoT development time

January 5th, 2017
PI 9498 DF/CH

"Sensor-based connected devices and solutions lie at the heart of many applications today, including connected mobility, Industry 4.0 and logistics – and we offer some of the most innovative products in these dynamic sectors", said Markus Lang, General Manager of Bosch Connected Devices and Solutions

76. According to Defendants, "[o]ne of the first critical IoT infrastructure elements" Bosch built was the Bosch IoT Suite. (Exhibit 3)

77. The Bosch IoT Suite is a cloud-enabled software package for developing IoT applications. It provides a comprehensive toolbox on Bosch's IoT hardware platform:



About Bosch Software Innovations

Bosch Software Innovations, the Bosch Group's software and systems house, designs, develops, and operates innovative software and system solutions that help customers around the world both in the Internet of Things and in the traditional enterprise environment. The Bosch IoT Suite is Bosch Software Innovations' comprehensive toolbox in the Cloud. The software package, which is provided as Platform as a Service, allows the interaction of devices, users, companies and partners on a centralized platform.

78. Bosch provides the following description of its IoT Suite in a publicly available white paper titled "Addressing the Real-World Challenges of IoT Solution Development" (Exhibit 5):[1]

> The Bosch IoT Suite provides the foundation for service enablement, both in terms of connecting things to the internet - reliably, securely, cost effectively, and a scale- and in terms of delivering the backing application logic for value-added services. The Suite consists of a set of software services that provide all the necessary key middleware capabilities for building a sophisticated IoT application from top to bottom. Customers can use any combination of the IoT services as needed to rapidly implement the desired solution.

79. Defendants publish the following description on the bosch.com[2] website:

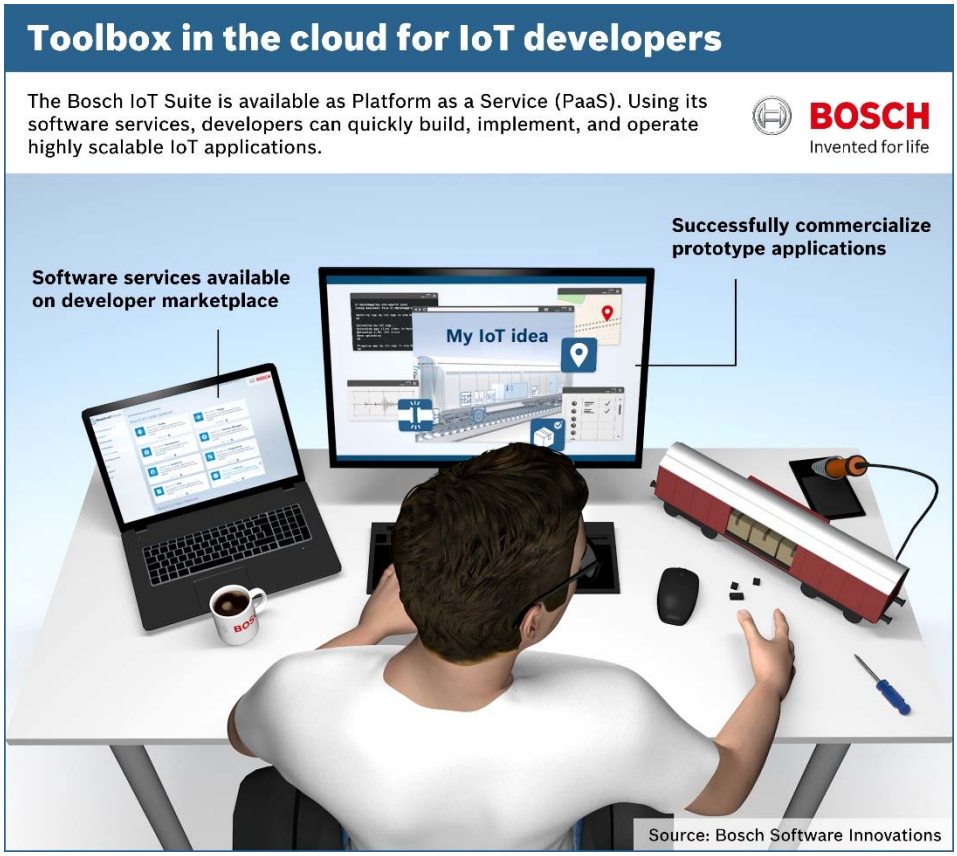
> The Bosch IoT Suite allows developers to quickly build, implement, and operate cloud-based and highly scalable IoT applications. Its cloud services address all common IoT requirements, such as reliably managing devices, machines, and gateways; providing secure access management; executing software rollout processes; connecting third-party systems and services; and analyzing data.

80. With the Bosch IoT Suite, Defendants provide developers a "Toolbox in the Cloud":

---

[1] https://www.boschsi.com/media/bosch_si/iot_platform/20160601_iot_solution_development_white_paper.pdf.

[2] https://www.bosch.com/products-and-services/connected-products-and-services/software-solutions/



81. Defendants identify the core functions of the Bosch IoT Suite as including: (i) device management; (ii) business logic; (iii) process and integration logic; and (iv) data mining and analytics.

82. Bosch IoT Suite device management provides device abstraction so devices are available to users as logical things or systems within the context of IoT applications.

83. Defendants identify the following benefits of the Bosch IoT Suite:

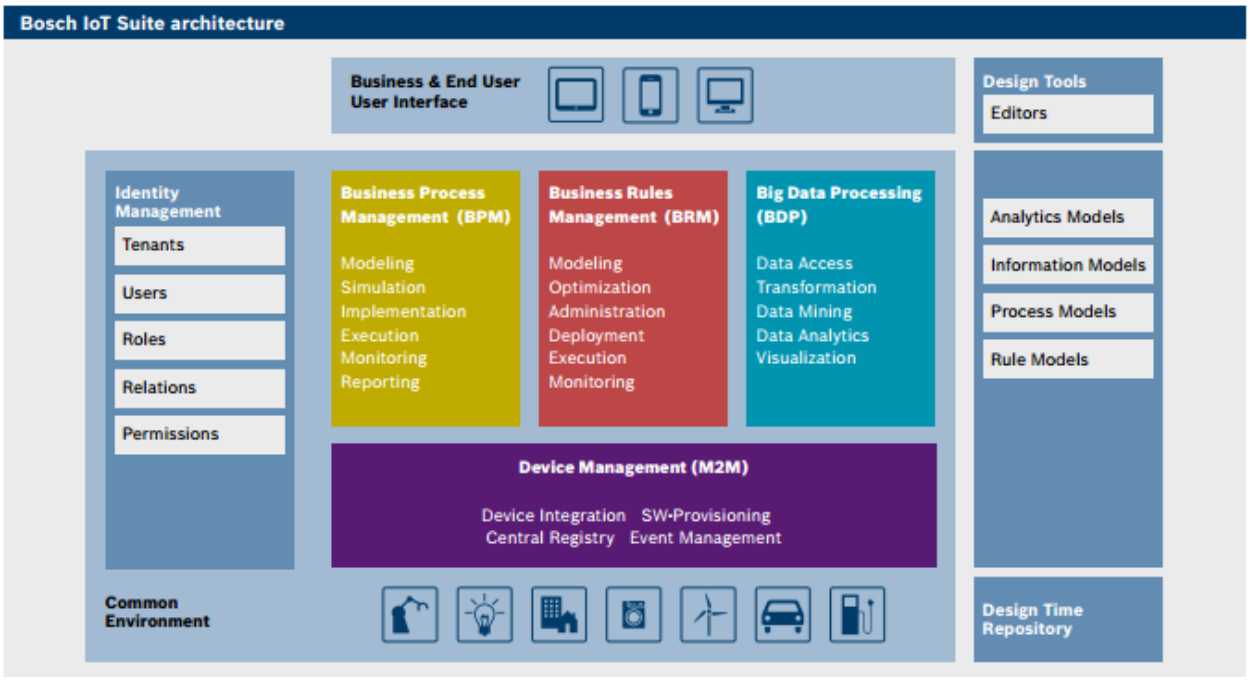
- Fast application development thanks to integrated IoT middleware that covers rules, process and device management, orchestration, and operation.
- State-of-the-art security technology that can be used directly in application development.
- Availability of a trusted and secure IoT cloud.
- Consistent and systematic access to Bosch devices and third-party devices based on open standards using information models derived from business

logic.

- Open platform due to the incorporation of numerous open-source software products and consistent standards support.

- Flexible and adaptable pricing model (depending on the number of connected devices, users, transactions) that also allows for partner business models with revenue sharing.

84. Defendants provide the following graphic to describe the Bosch IoT Suite architecture (Exhibit 6):



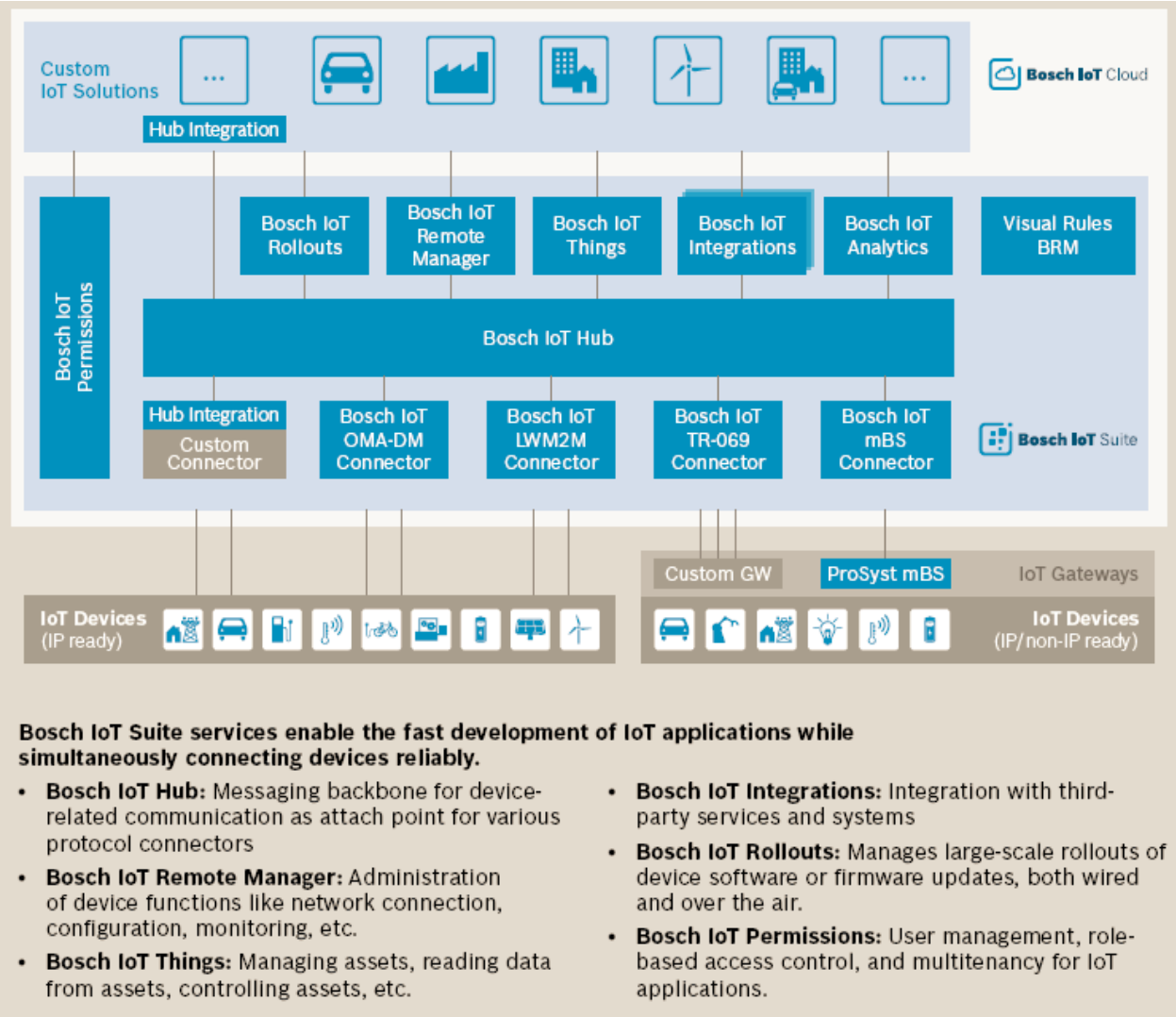
85. Elements of the Bosch IoT Suite include: IoT Hub; IoT Remote Manager; IoT Things; IoT Integrations; IoT Rollouts; and IoT Permissions.

86. Bosch IoT Things is the brand for Defendants' service offered in the cloud. It provides central management of assets and data (reading data from assets and controlling assets, for example) for the Internet of Things based on Device Abstraction and previously was known as the Central Registry. (Exhibit 7)

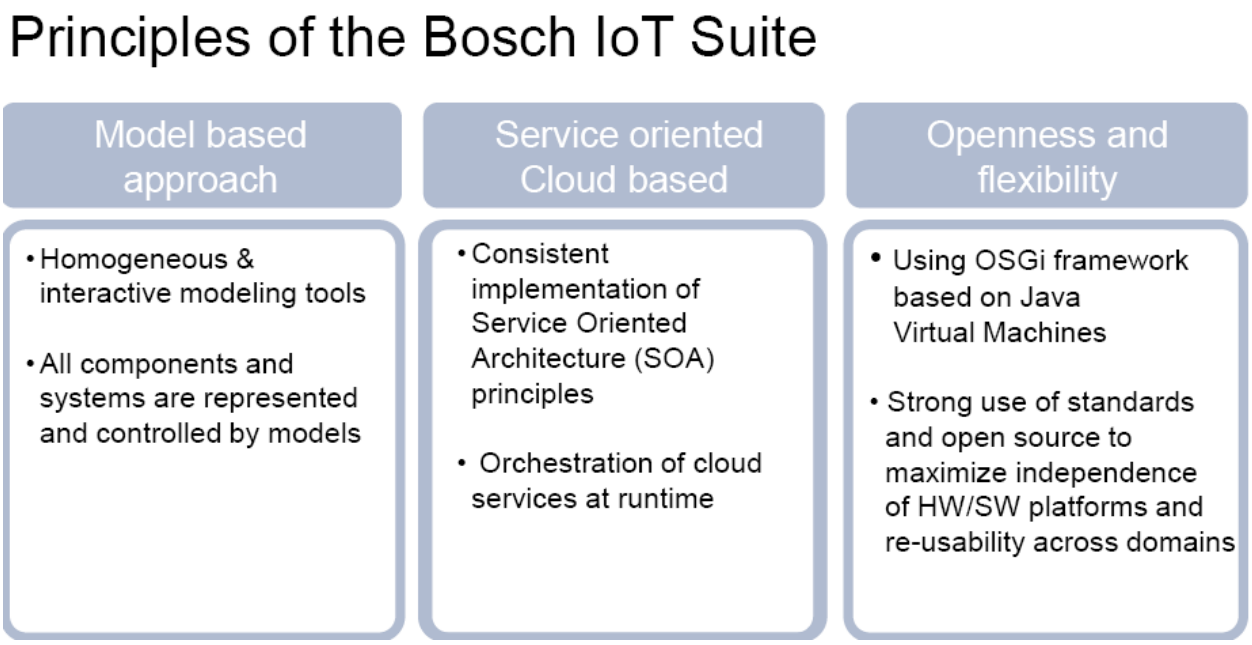
87. IoT Rollouts manages device software and over-the-air firmware upgrades.

88. Defendants provide the Bosch IoT Suite services to enable developers to quickly build, implement, and operate cloud-based and scalable IoT applications. (Exhibit 8)

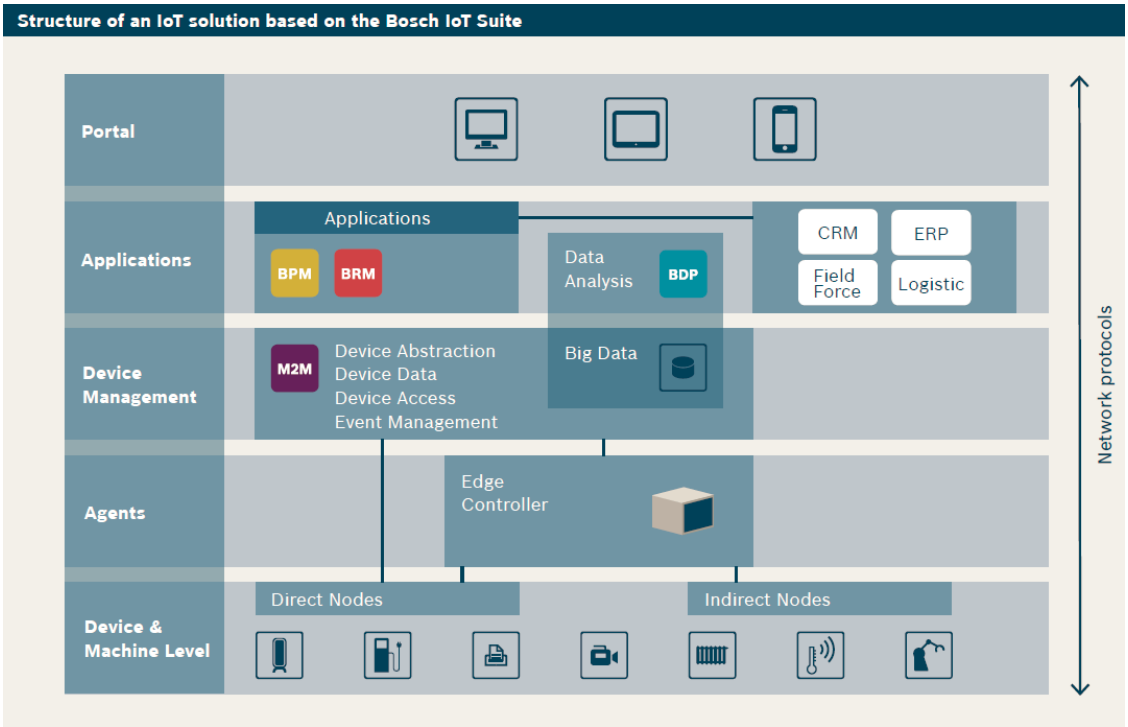
89. The following graphic shows various components of the accused instrumentalities and describes their respective primary functionalities:



90. The Bosch IoT Suite takes a model-based approach meaning that all components and systems are represented and controlled by models. (Exhibit 9)



91. The structure of an IoT solution based upon the Bosch IoT Suite is shown below. (Exhibit 10)



92. In the accused instrumentalities, a Thing is a generic entity and is mostly used to cluster multiple Features and manage access to the data and functionality the Thing represents.

(Exhibit 7)

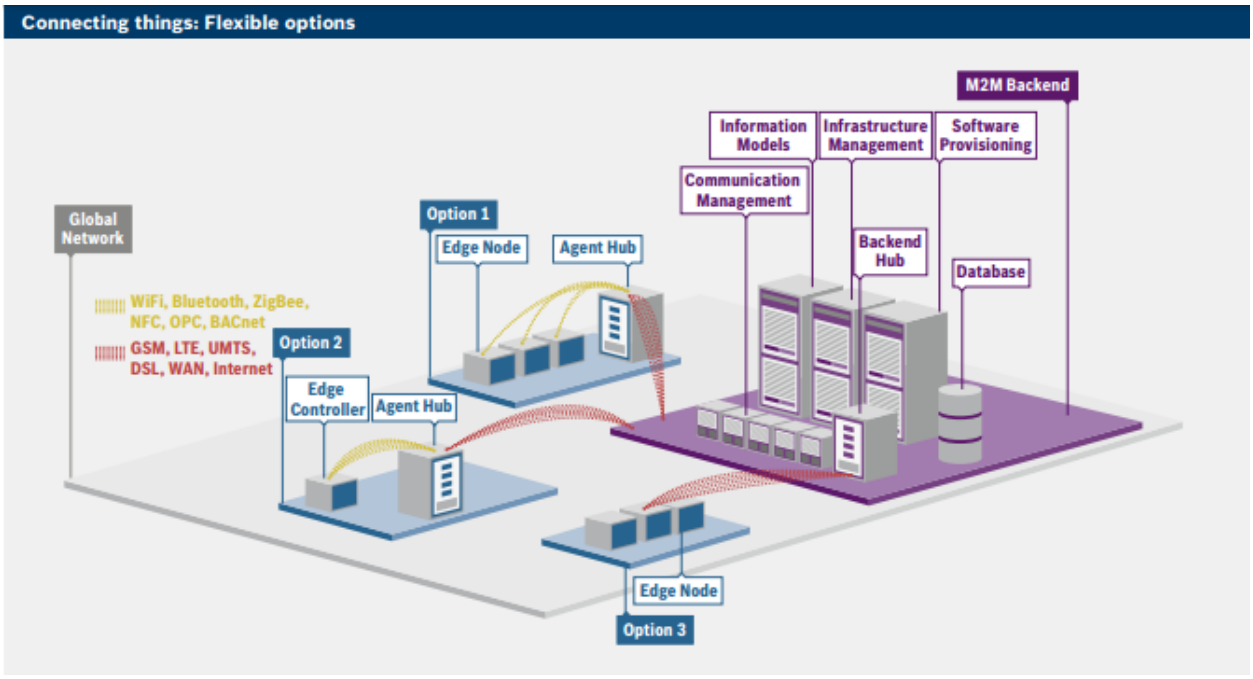
93. BIOTS comprises a hardware platform that typically includes servers, IoT Gateways, and storage, but may be implemented in other infringing configurations.

94. Deployed accused instrumentalities may include Bosch ProSyst Gateways and Gateway Software (e.g., ProSyst mBS) that facilitates interaction between connected devices. In such systems, ProSyst software may perform a role of translator if necessary to establish communication:[3]

> In combination with the Bosch IoT Suite. . .the ProSyst software will enable our customers to launch new applications on the Internet of Things more quickly and be one of the first to tap into new areas of business, . . . The ProSyst software is highly compatible with the Bosch IoT Suite, our platform for the Internet of Things. Above all, it complements our device management component by supporting a large number of different device protocols.

95. Defendants' hardware platform provides communication management, software provisioning, device management and all necessary functionality and may be accessed through various standardized comunication systems (e.g., WiFi, Bluetooth, LTE, etc.) (Exhibit 10):

[3] https://www.bosch-si.com/corporate/insights/features/middleware-specialist-prosyst.html.



96. Deployed BIOTS are adapted to be communicably connected to active objects including sensors and actuators.

97. In 2013, Bosch launched a new division, Defendant Bosch Connected Devices and Solutions (referred to herein as "BCDS").

98. BCDS specializes in the development of networked sensors and actuators, and in a February 2014 Press Release, Defendants described BCDS's focus (Exhibit 11):

Press release

BOSCH

"Bosch ConnectedWorld" conference
Bosch connects the virtual and the real worlds
Bosch CEO Denner: "Bosch offers solutions for connected life"

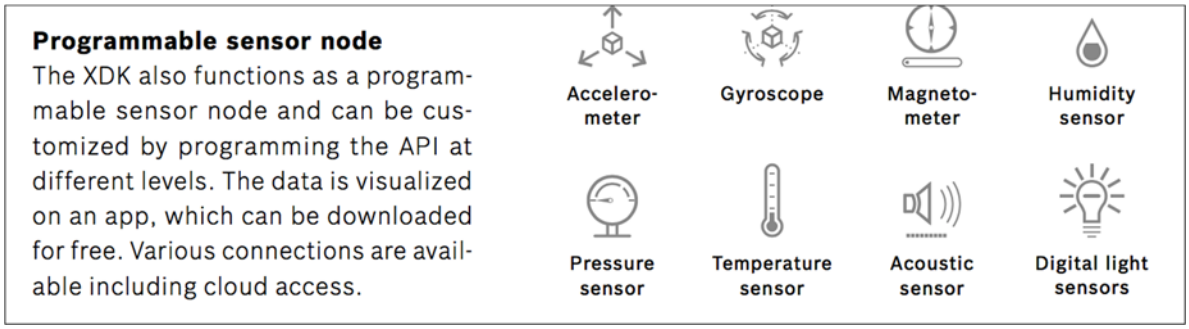
February 5, 2014
PI 8444 RB Res/SL

- New business models with high customer benefit
- Bosch Software Innovations offers software and systems solutions

In addition, Bosch recently founded a separate company dedicated entirely to the internet of things. **Bosch Connected Devices and Solutions GmbH** offers the compact electronic products and the software know-how needed to make devices and objects internet-capable in a wide range of application areas. The company will initially concentrate on sensor-based applications for households ("smart home") as well as for the transportation, traffic, and logistics segments. **Bosch Sensortec** is a world leader in micromechanical sensors for the internet of things, with some three million of these tiny components being produced every day at the company's state-of-the-art wafer fab in Reutlingen, Germany.

99. Defendants' XDK is a fully integrated Cross Domain Development Kit consisting of both hardware and software and includes the following MEMS devices: accelerometer, magnetometer and gyroscope in addition to humidity, pressure, temperature, acoustic and digital light sensors. (Exhibit 12)

100. The XDK also includes a microcontroller, integrated antennas, a micro SD card slot and a rechargeable battery. Multiple wireless technologies ensure connectivity and access to the Internet, and the included software development environment and operating system offer access to different API layers allowing the user to program at their preferred depth.



101. Defendants license the Bosch XDK Software Development Kit (SDK) under an End User License Agreement ("EULA"). (Exhibit 13)

102. The Bosch XDK SKD End User License Agreement restricts customers from

adjusting, modifying, or otherwise amending the XDK SDK. Under the EULA, Defendants' customers are prohibited from connecting the XDK SDK to other programs in a manner other than via the interfaces provided by Bosch, may not decompile it to another display format, or remove, circumvent, or modify security codes, if any, or features serving the identification of the XDK SDK, or to remove information contained in the SDK and program documentation about authorship, copyright, or other property rights of Bosch. Defendants' XDK SDK and associated documentation is considered as being confidential and may not be disclosed to any third party. (Exhibit 13)

103. An example of a commercially available Bosch sensor is the Retrofit eCall, a device installed in a vehicle for detecting a collision. If that happens, a call is generated by the user's mobile phone to a call center where a live operator can speak with the driver and dispatch emergency services if necessary.

104. In 2015, Bosch established a new subsidiary to operate the company's smart-home activities, including related software and sensor-system expertise. Robert Bosch Smart Home GmbH aims to support Defendants' goal of being a single supplier of products and services for connected homes in the United States.

105. Defendants describe the Bosch Connected Industrial Sensor Solution (CISS) in a published brochure (Exhibit 14) as providing a sensor device for harsh industrial environments:

Bosch Connected Devices and Solutions GmbH | One Pager

BOSCH
Invented for life

Connected Industrial Sensor Solution | CISS

Improving your Utilization and Processes

Increase your manufacturing efficiency by monitoring your machines, processes and environmental conditions. The Connected Industrial Sensor Solution (CISS) is a compact multi-sensor device for harsh environments. Machine condition tracking enables predictive and remote maintenance to save costs. With CISS production yields can be optimized via live process monitoring. Due to its motion and environmental sensing abilities the CISS is ideally suited for I4.0 applications.

Application Example: Condition Monitoring at an Injection Molding Machine in harsh environment

106. For transportation logistics applications, Defendants provide a Transport Data Logger (TDL) that, once attached to a shipment, measures and records temperature, humidity, tilt, and shock, with the data visualized through a mobile application. The limits of each parameter can be individually configured, and any violation is traceable and clearly assignable throughout the entire supply chain. (Exhibit 15)

107. Bosch IoT Things are used in the accused instrumentalities as a "handle" for multiple features belonging to a Thing. Things may be a sensor or actuator. (Exhibit 16)

108. Things are identified and described by attributes and Thing IDs. Attributes are typically used to model static properties at the Thing level, and Thing IDs conform to IETF RFC 2396 governing characters that may be used for uniform resource identifiers. (Exhibit 16)

109. For example, the Bosch IoT Things Protocol provides commands to initiate an operation at the Things Service. (Exhibit 17)

Command

Each example always starts with a command message that initiates an operation at The Bosch IoT Things service (e.g. create a thing, retrieve a thing).

```
{
  "topic": "com.acme/xdk_58/things/twin/commands/modify",
  "headers": {
    "correlation-id": "a780b7b5-fdd2-4864-91fc-80df6bb0a636"
  },
  "path": "/"
  ...
}
```

110. BIOTS Things are represented by a value in the thingIds array and may be retrieved using the appropriate command in the Bosch IoT Things Protocol. (Exhibits 18 and 19)

111. The BIOTS Things Service provides two APIs for sending messages to Things and receiving messages from Things: REST API and Java API. (Exhibits 20 and 21)

REST API and Java API

Bosch IoT Things service provides two interfaces:

- A (blocking) RESTful HTTP API (Application Programming Interface) with a sophisticated resource layout, that allows to create, read, update and delete Things and the Thing's data.
- The other interface is WebSockets. Our Java client implements a non-blocking API (Application Programming Interface) which uses this WebSockets interface to communicate with the Things service.
  The communication is implemented by sending and receiving messages with the Simple Text Oriented Messaging Protocol (https://stomp.github.io/ (https://stomp.github.io/)).

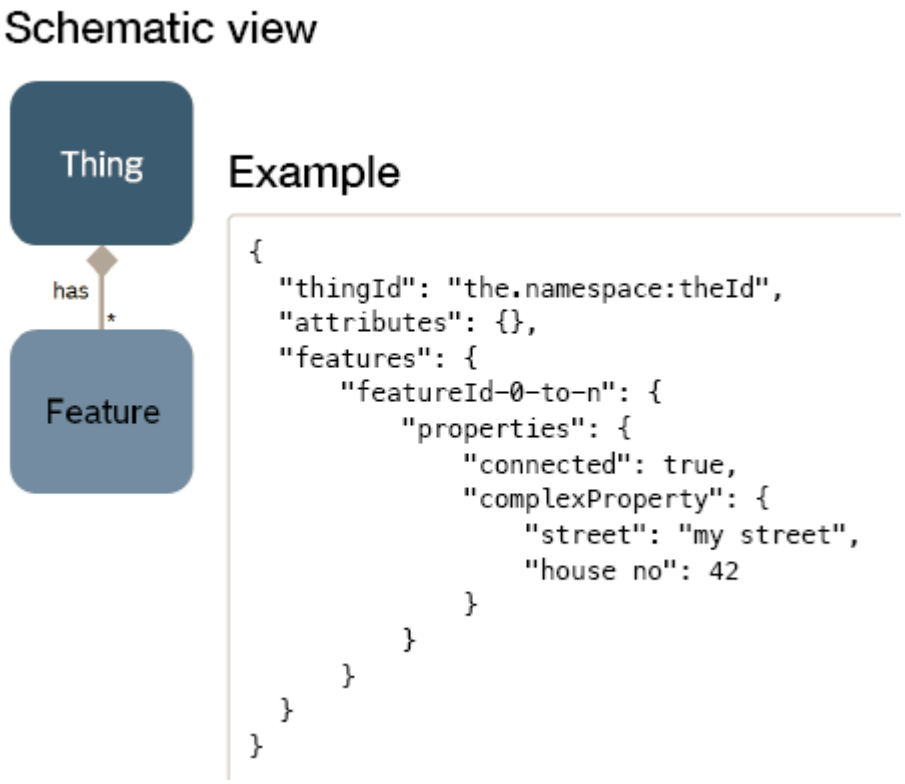
Both APIs are **equally powerful** and allow the same operations to work with the Thing's data, send messages to Things and receive messages from Things.

- The Java client is mainly targeted to environments which support a Java runtime to implement e.g.
  a device integration layer on a hub/gateway,
  or to implement a rich client
  or to implement a cloud service of a custom solution.
- The lightweight REST API (Application Programming Interface) can be used on less powerful devices lacking a Java runtime or supporting other (scripting) languages like JavaScript, Python, C/C++.
  The REST API (Application Programming Interface) also supports convenient development of Web-based UIs.

112. The BIOTS hardware platform includes communication modules configured to connect with network servers. A protocol binding defines how Things protocol messages are transported (e.g., over websocket or AMQP 1.0). (Exhibit 22)

113. In the accused instrumentalities, Things have "Features" that are Java Objects representing active objects by properties that can be categorized to manage the status and configuration of the Thing as well as any fault information. Defendants provide the following

example and schematic depicting the logical relationship between a Thing and its Feature (Exhibit 16):



114. Features may be modified or deleted by command. (Exhibits 23 and 24)

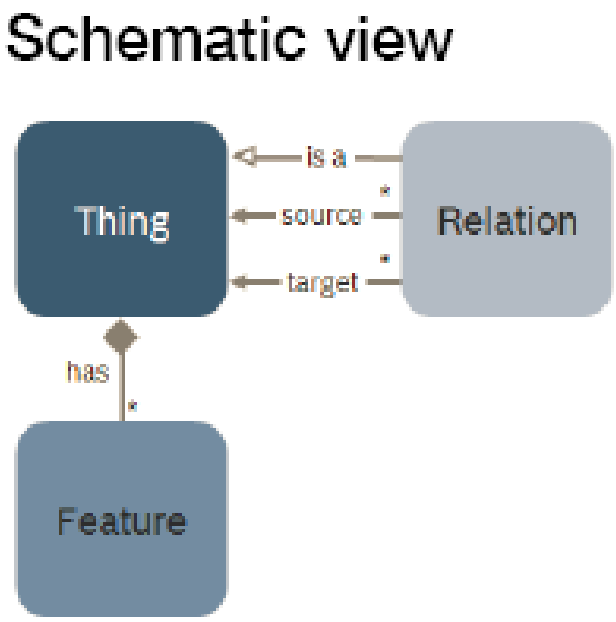
Modify single Feature of a Thing

Create or modify a specific Feature (identified by the Feature ID in the `path`) of the Thing (identified by the `<namespace>` and the `<thingId>` in the `topic`).

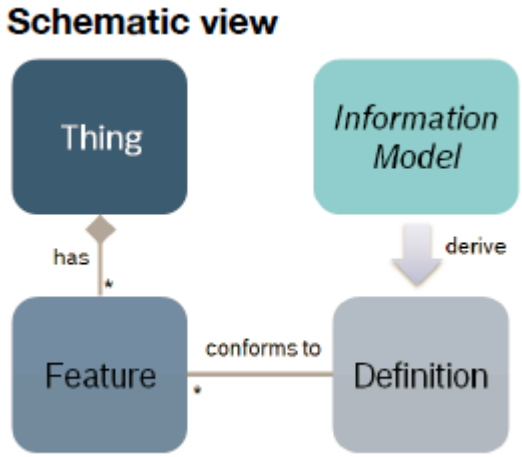
Command

Field	Value
topic	`<namespace>/<thingId>/things/twin/commands/modify`
path	`/features/<featureId>`
value	The specific Feature of the Thing as JSON, see property `features` of Things JSON schema. see Things protocol payload (JSON)

115. Relations describe the assignment of a source-Thing to a target-Thing. A Relation is a special Thing using the Feature mechanism of Things and is shown schematically below. (Exhibit 25)



116. Bosch IoT Things uses Definitions derived from device abstractions (i.e., Information Models) to enable large-scale management. (Exhibit 26)



117. Defendants publish protocol examples of a command to create a Bosch IoT Thing and instructions how to associate applications and Things. (Exhibits 27, 28 and 29)

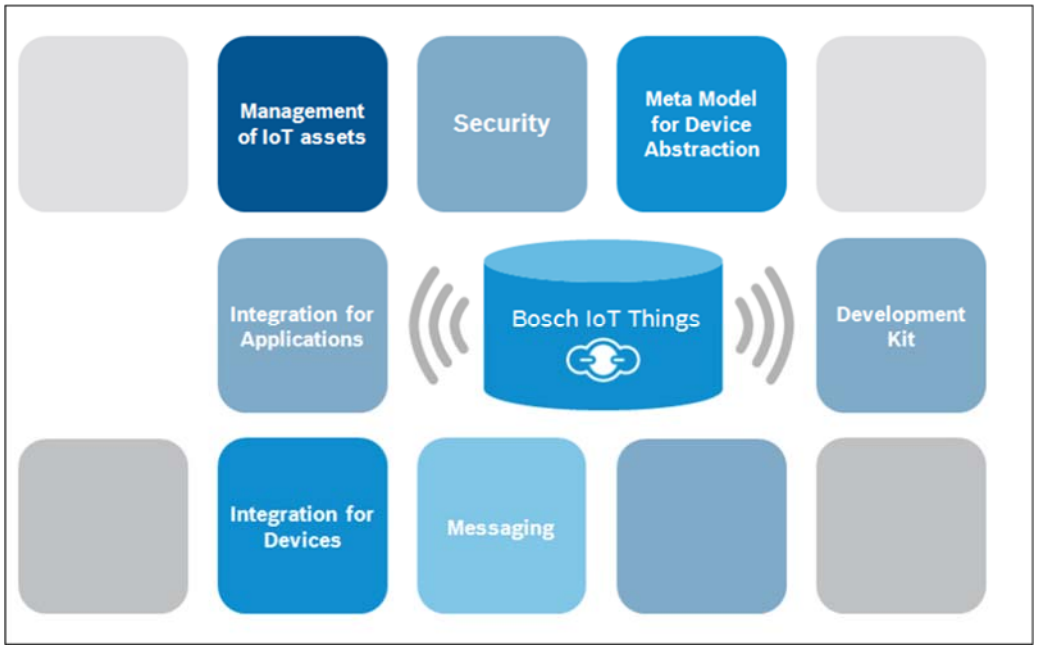
Create a Thing

Command

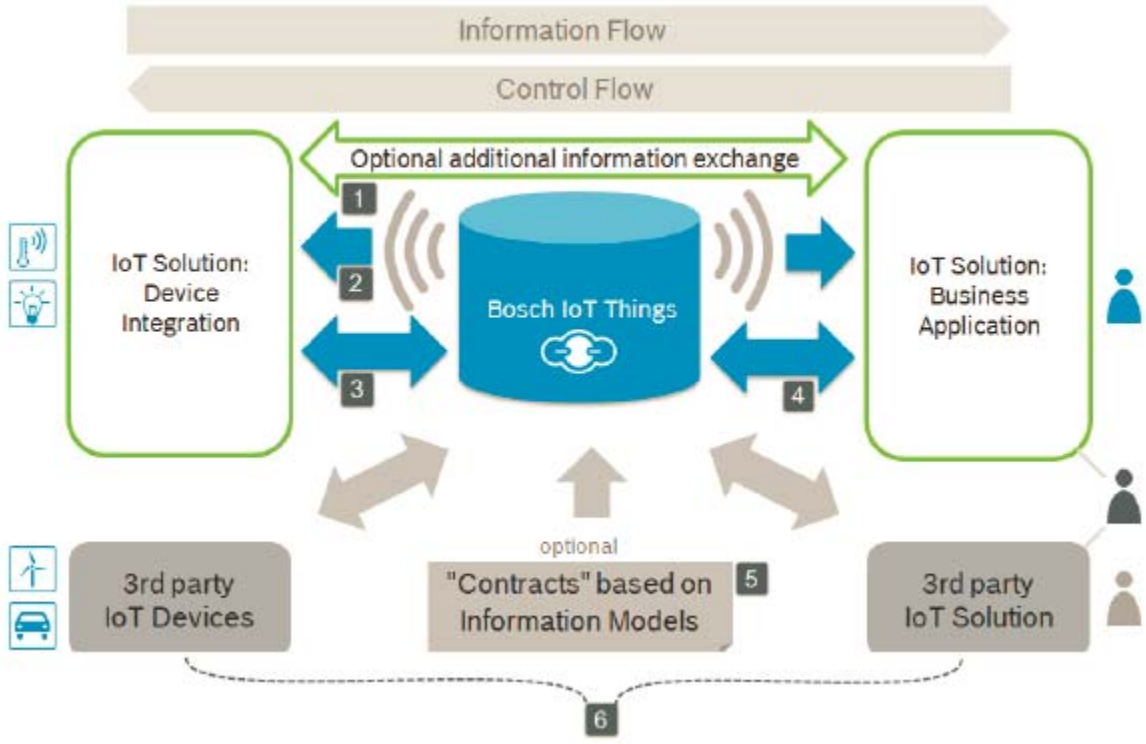
```
{
  "topic": "com.acme/xdk_53/things/twin/commands/create",
  "headers": {
    "correlation-id": "a780b7b5-fdd2-4864-91fc-80df6bb0a636"
  },
  "path": "/",
  "value": {
    "thingId": "com.acme:xdk_53",
    "attributes": {
      "location": {
        "latitude": 44.673856,
        "longitude": 8.261719
      }
    },
    "features": {
      "accelerometer": {
        "properties": {
          "x": 3.141,
          "y": 2.718,
          "z": 1,
          "unit": "g"
        }
      }
    }
  }
}
```

118. Bosch IoT Things enables applications, cloud services, and devices to manage asset data and provide secure messaging between IoT solutions and their devices. Bosch IoT solutions can store and update the asset and device data, properties, and relationships of IoT assets. (Exhibit 30) Solutions are "registered" on the BIOTS hardware platform. (Exhibit 31)

119. Defendants' IoT Things service (aka Central Registry) enables management of IoT assets. (Exhibits 32 and 33)

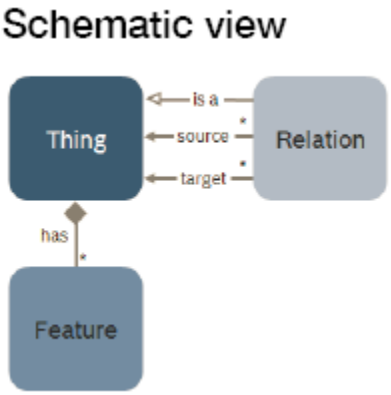


120. In the following picture, Defendants illustrate the role of Bosch IoT Things within an IoT solution (Exhibit 34):



121. In addition to Features, Things may have "Relations" that describe the assignment

of a source-thing to a target-thing. In the accused instrumentalities, a Relation is a special Thing using the Feature mechanism of Things. These associates are shown schematically below. (Exhibit 25)



122. These capabilities of the Bosch IoT Suite enable configuration, deployment, management, and communication through standard interfaces with "handles" representing actuators and sensors.

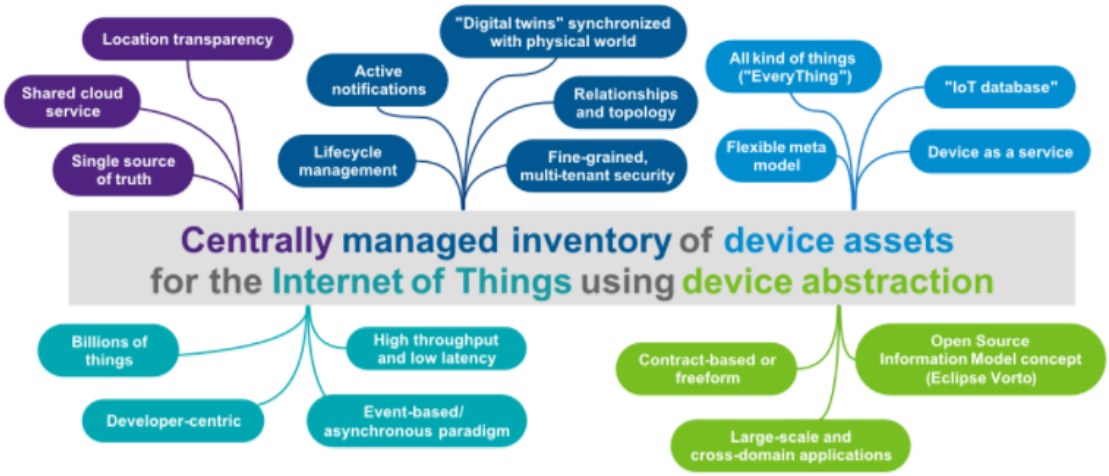
123. A Feature is used to manage all data and functionality of a Thing. A Thing can handle any number of Features, which are identified by Feature ID. Data related to Features is managed in a list of properties:

```
{
  "thingId": "the.namespace:theId",
  "attributes": {},
  "features": {
      "featureId-0-to-n": {
          "properties": {
              "connected": true,
              "complexProperty": {
                  "street": "my street",
                  "house no": 42
              }
          }
      }
  }
}
```

124. To achieve this functionality, BIOTS relies on device abstraction:

Bosch IoT Things

Bosch IoT Things enables applications, cloud services, and devices to manage the data of their IoT assets in a simple, convenient, robust, and secure way. Solutions can store and update the data, properties, and relationships of your domain's assets and get notified of all relevant changes.



Bosch IoT Things allows all connected devices — such as tools, cars, sensors, and other web-enabled things — to be integrated into cloud services or other applications and to interact with each other. Specifically, that means that applications, other cloud services, and devices can manage their asset data and share it across IoT solutions. What is more, IoT solutions can store the data for further analysis, remotely request the change of properties for physical devices, and describe relationships of your domain's IoT assets in a simple, convenient, robust, and secure manner.

(Exhibit 35)

125. Defendants provide a Java client and HTTP API to enable registration, communication, and updating of BIOTS Things (Exhibit 36):

Manage and organize your things

Register, read, and update your *Things* using our lean yet powerful Java client or HTTP API . The Java client supports the asynchronous and non-blocking programming paradigm. The REST-like HTTP API uses a sophisticated fine-grained resource layout that allows you to address every single part of your thing. Bosch IoT Things also allows you to easily organize your things by relating them to each other. For instance, via *Relations*, you can create hierarchies or meshes of things and enrich these relationships with solution-specific semantics.

126. Defendants' implementation of device abstraction provides a basis for implementation of device drivers and device-specific protocols:

Management of IoT assets

- Manage data of Things by properties (e.g. for configuration, status and fault)
- Manage life-cycle of Things
- Manage relations between Things (e.g. topology)
- Represent and use the functionality of Things and events triggered by Things
- Complex search
- Dashboard to browse all IoT assets and analyze statistics

Integration for Applications

- Easy to use REST-like API for access and interaction with data and functionality of Things
- Easy to use Java client for access and interaction with data and functionality of Things

Integration for Devices

- Integration of "indirect" devices (non-IP ready) and "direct" devices (IP ready) via Java client or REST-like protocol including support for custom Hubs/Gateways

Meta Model for Device Abstraction

- Unified language to describe devices based on Eclipse Vorto (see http://www.eclipse.org/vorto/)
- Describes structures and functionality of devices
- Leverage device description for contract-based development of large-scale or cross-domain solutions
- Basis for implementation of device drivers and device-specific protocols as well as code templates for solution development

127. In operation, Defendants' Things Service is used to keep track of device state and allows applications to interact with sensors and actuators in different domains. (Exhibit 37)

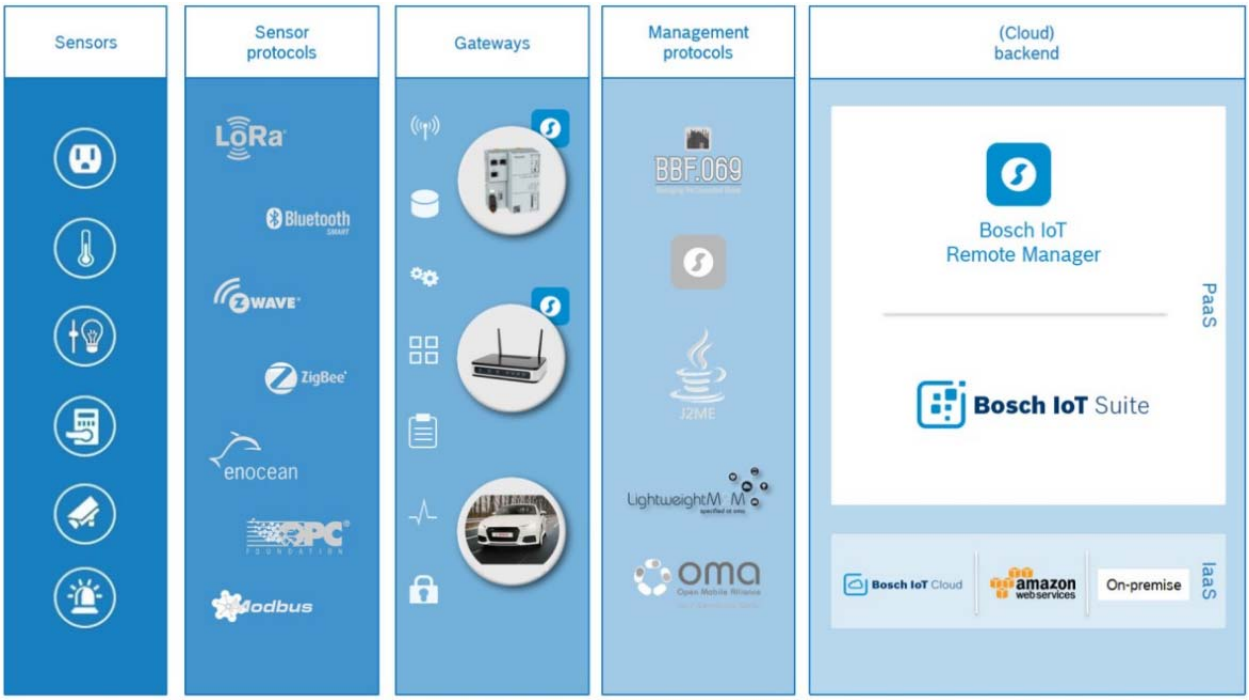
128. The Things protocol payload contains application data (e.g., an updated sensor value). A command elicits a Things protocol response (e.g., a status code indicating success or failure of the command).

129. To manage distributed devices and devices with different software requirements, Defendants integrated the ProSyst OSGi-based framework in IoT gateways and adopted certain international standards and tools to define device and asset interfaces and support scalable data ingestion, command and control messaging, and provide interfaces for provisioning and managing device identity and access control rules. Bosch participates in the Eclipse IoT and Automotive Working Groups to develop such standards. (Exhibit 8) Defendants recently acknowledged the problems associated with scaling IoT deployments that Dr. Helal and his co-inventors identified

and addressed in the asserted patents (Exhibit 38):

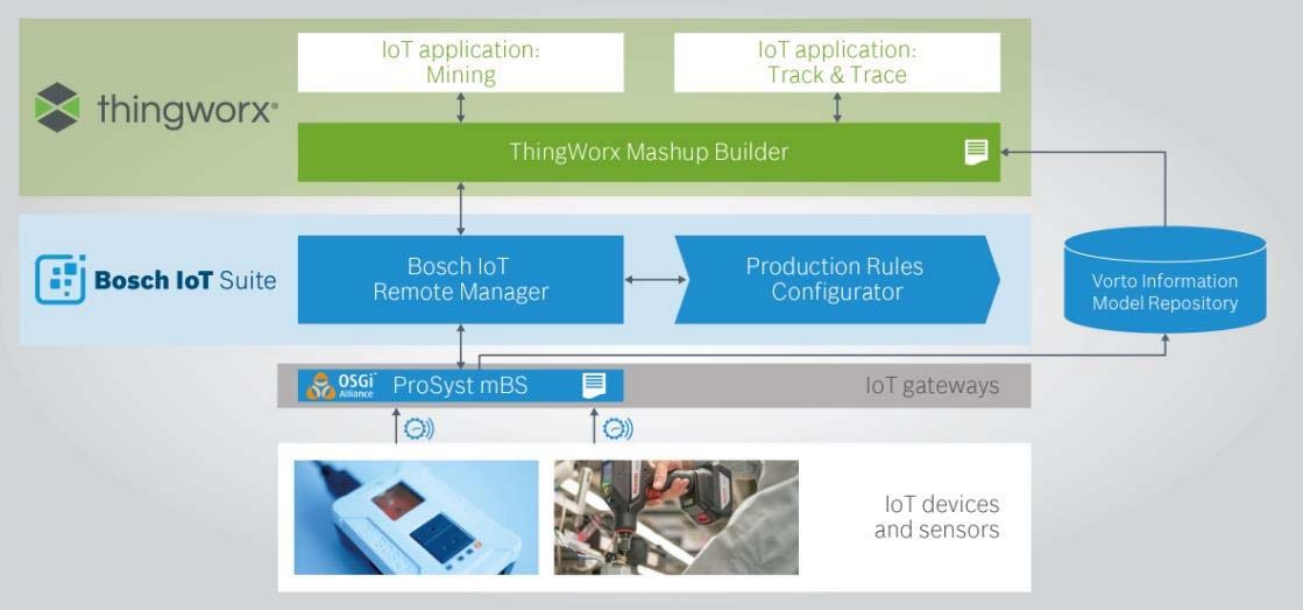
> In our experience, it is not the sheer number of connected devices that poses the major challenge to making the loT a reality. Instead, it is the heterogeneity and diversity of constantly evolving things that must be reliably addressed. Our gateway and device management software therefore facilitates interaction among connected devices.
>
> True to our open loT strategy, we provide our users with an open platform for the loT market, leveraging open source technologies and open standards such as Linux, Java, Eclipse IDE, OSGi, and Cloud Foundry. OSGi in particular is a global standard as well as the key to our gateway technology and remote technical management. It is also ideal for managing distributed devices, and is thus used by multi-device vendors for complex software requirements. Optimized for use in embedded and resource-constrained platforms and products, our gateway software supports all major loT protocols, thus facilitating the integration of devices and sensors.

130. The BIOTS Remote Manager is available as a cloud service or may be deployed as a standalone, on-premise product. (Exhibit 38)



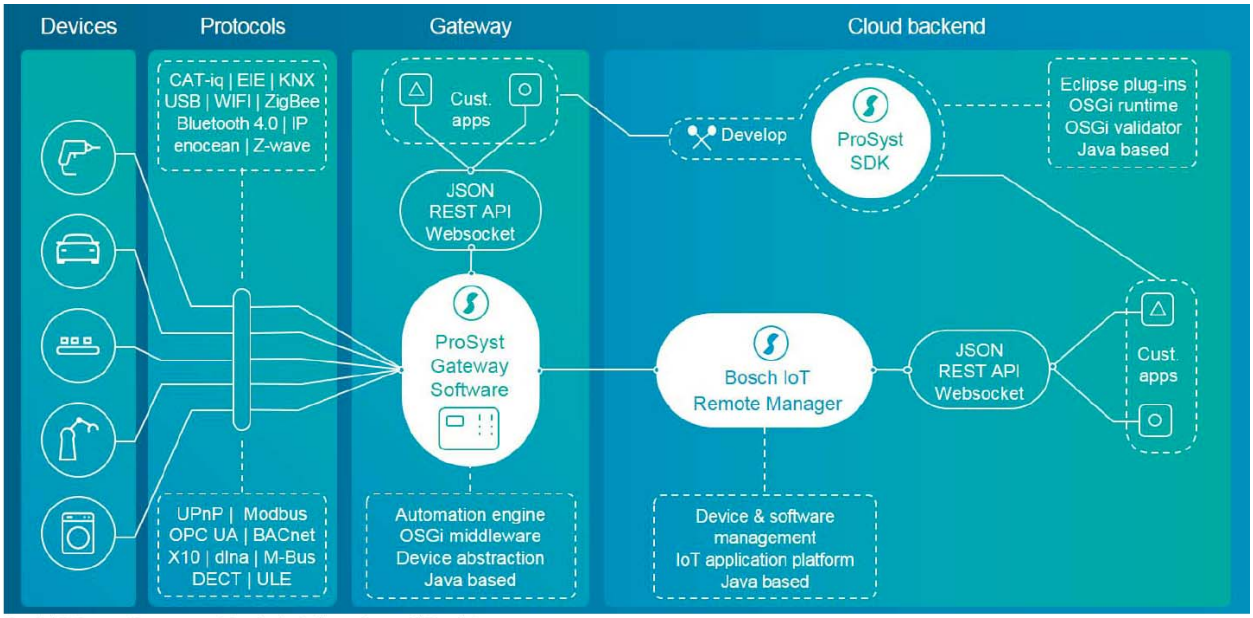
131. BIOTS Remote Manager provides: remote access for apps and application servers to gateways and devices; advanced gateway, device, and software management for mass-management operations (bulk device configuration and provisioning), software repository

(inventory of device software and configuration settings); stand-alone and web-based administration interface; end-to-end security (user authentication and authorization, network communication, certificate management); advanced load balancing; scalability; and enables platform, application & service lifecycle management (remote installation, updating, uninstallation, configuration – on the fly), firmware & file update, remote configuration & provisioning, remote monitoring & diagnostics (status checks, logging, monitoring, etc.), and remote security administration in commercial product deployments such as smart homes, connected cars, and the Industrial Internet. (Exhibit 38)



132. Defendants provide security within the BIOTS to ensure secure connection of sensors, actuators, and embedded systems, using encryption and a public key infrastructure (PKI) as well as access control for Things. (Exhibit 39)

133. BIOTS Remote Manager includes a software repository providing an inventory of device software and configuration settings. (Exhibit 40)



Bosch IoT Remote Manager provides the basis for end-to-end IoT solutions.

134. BIOTS devices may comprise a sensor (e.g., a thermometer providing temperature data) or an actuator (e.g., a switch to control supplied electrical power) or both (e.g., an electric motor incorporting the sensor and actuator as a safety to reduce power before overheating).

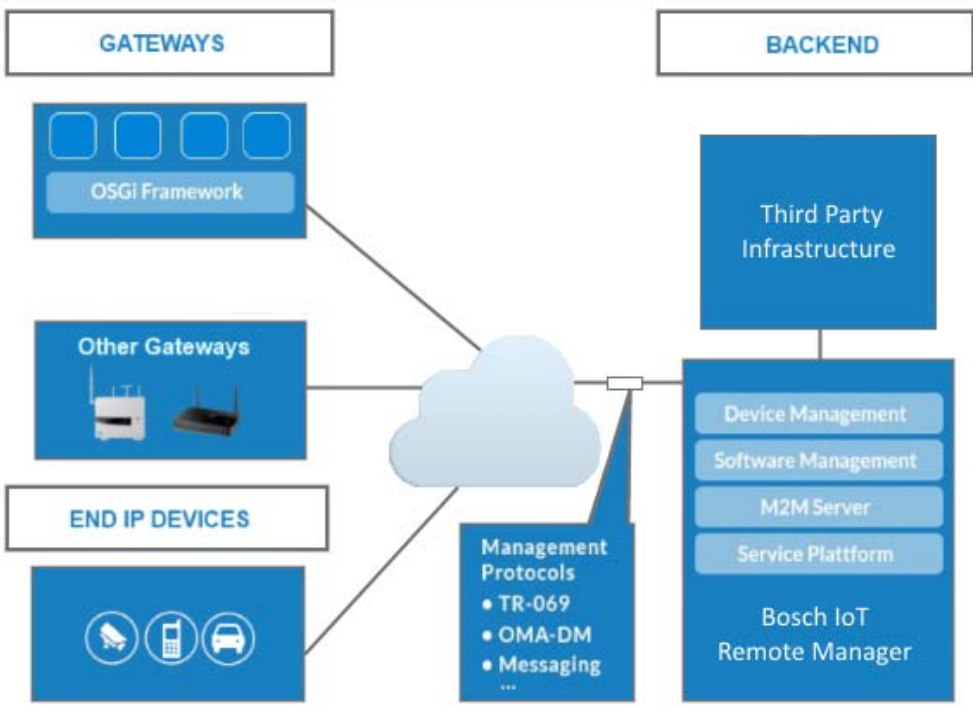
135. BIOTS middleware modules residing and/or executed on the hardware platform include ProSyst Gateway Software, Bosch IoT Remote Manager, Bosch IoT Hub, and BIOTS Things service and their respective components and associated software modules.

136. Middleware for BIOTS gateways faciliate communication between the platform and connected devices. For example, ProSyst software assumes a translator role. If technologies such as central heating systems, household appliances, and security cameras are to be interlinked in a smart home, they must all "speak the same language." This is especially difficult when the products are made by different manufacturers, use different communication protocols, or are not web-enabled.[4]

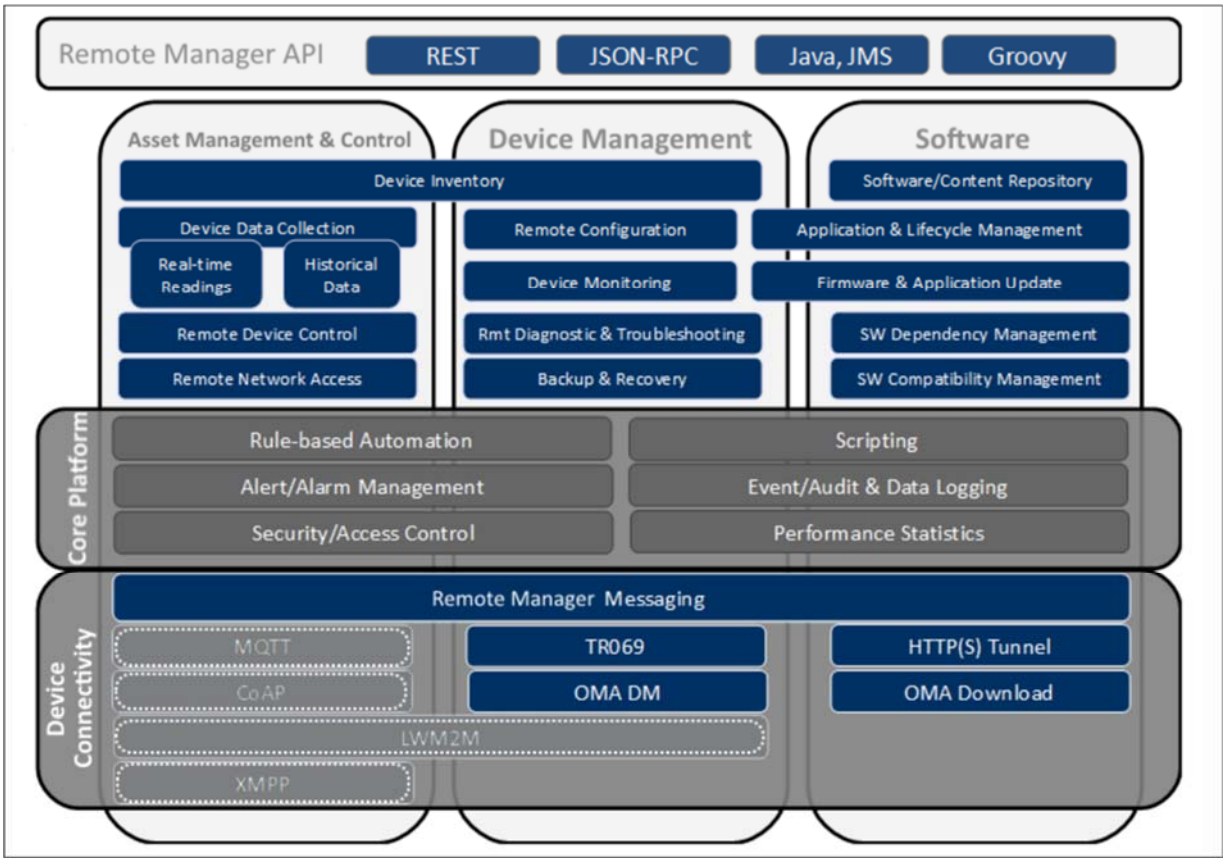
[4] https://www.bosch-si.com/corporate/insights/features/middleware-specialist-prosyst.html

> ProSyst's mBS is an IoT middleware stack for gateways that facilitates the interaction between connected devices in the mobility, smart home, and connected industry segments. It is based on open standards with OSGi at its core. OSGi is a globally accepted standard and the key to our gateway technology and remote technical management, used to manage distributed devices and by multi-device vendors with complex software requirements.

137. Remote Manager is in production use in connected homes, fleet management, healthcare, and M2M scenarios. An exemplary IoT/M2M End-End Solution is provided by Defendants in their published product documentation. (Exhibit 41)



138. Functional aspects of BIOTS Remote Manager are shown diagrammatically below (Exhibit 41):



139. According to Defendants, "Java Script API can be used when developing an end user application. This API masks and hides the communication complexity to the backend (or gateway). It is developed on top of REST/JSON for the requests and Web sockets/Long polling for eventing. The state of the device is directly represented in the browser so the application programmer can concentrate over the UI and application logic." (Exhibit 41)

140. The Remote Manager system keeps device specific properties (i.e., the BIOTS Software Repository) which can be used for matching the requirements of a client bundle to the capabilities of a client device when the bundle is to be deployed on the device. (Exhibit 42)

141. The capabilities of a device are specified when it is added to the Remote Manager system. When a client bundle is installed on the device, device manager module contacts the Basic Principles of the Software Repository to get the most suitable client bundle. On its behalf, the

Software Repository contacts the profiles manager to match the requirements of the client bundle against the device capabilities according to predefined property and profile relations and rules. As a result, the Software Repository determines the best client bundle for the device. (Exhibit 42)

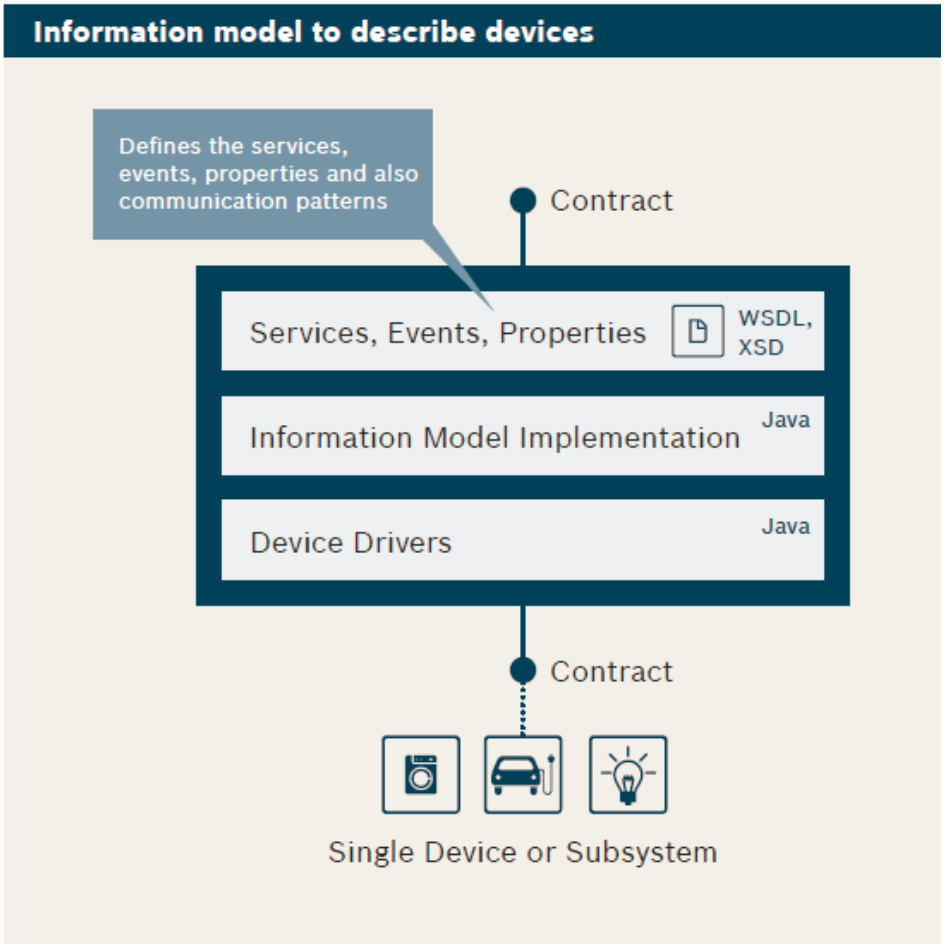
142. The Remote Manager retrieves a set of device capabilities when activated on a client device or when the device is synchronized. Remote Manager offers several ready platform properties with all necessary relations set. They are used in the process of client bundle requirement matching and deployment on devices. All retrieved capabilities are merged and sent to the backend where they can be used in the client bundle deployment process. (Exhibit 42)

143. The client bundle is defined to work on one or more devices whose capabilities match specific bundle requirements. When a client bundle is to be deployed on a device with special system characteristics (retrieved when the device is registered in the system), then the repository selects the most suitable client bundle for that device platform and Remote Manager deploys it. In this case, the Software Repository contacts the Profiles Manager. The Profiles Manager matches the requirements of the client bundles against the properties of the platform specifics. (Exhibit 42)

144. BIOTS middleware modules generate software services representing active objects deployed in the system.

145. Bosch IoT Suite uses information models to create generic descriptions devices and systems of devices. (Exhibit 10)

146. The following graphic describes the Bosch IoT Suite information model:



147. To streamline development, BIOTS enables generation of a digital twin of the physical asset based on readouts from machine components as well as additional sensors. This digital twin in the cloud provides additional functions and solutions including predictive analytics in a secure sandbox in the cloud which controls data access for each application. (Exhibit 3)

148. BIOTS employs control unit abstraction to unify the control of miscellaneous device units and introduce a common interface to manage different types of device controllable modules (e.g., software components, configurations, user settings, etc.).

149. A control unit is a collection of state variables and actions. State variables represent control unit's specific parameters, for example an X10 lamp may have a state variable for indicating on and off, and have unique ID within the control unit scope and value of certain type.

(Exhibit 43)

150. The actions of a control unit are the commands that the control unit can execute, such as turning a lamp on and off. An action has ID, unique within the control unit scope, and may have input and/or output arguments. The collection of state variable IDs and types and of action IDs form the interface of the control unit. The current values of control unit's state variables determine the present state of the control unit. (Exhibit 44)

151. Defendants' control unit abstraction provides a unified, well-known way to monitor the state of and execute an action on a control unit resulting in a state change. Therefore, although there is a great flexibility in what tasks can be invoked on units and the state variables they provide, applications can manage control units in a uniform way, without the need to have specific knowledge about the underlying resources. (Exhibit 44)

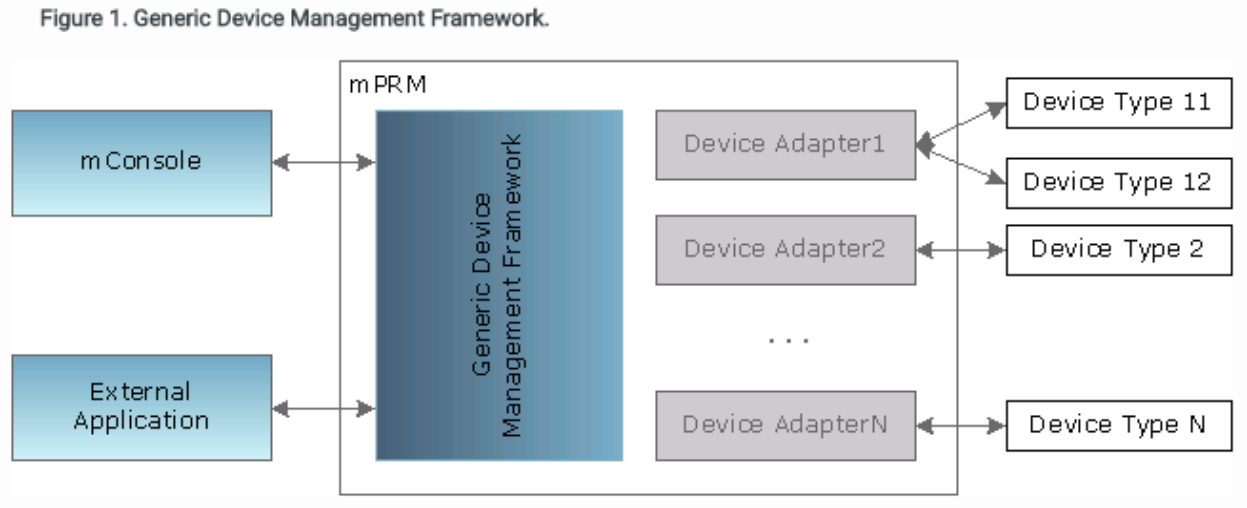
152. Control Unit Providers are pluggable units which represent specific devices and their elements as control units. Control Unit Providers are responsible for performing a range of functions (Exhibit 43):

- Communicating with the device
- Providing metadata of subordinate control units
- Providing correct values of control unit state variables
- Interpreting action invocations
- Creating and destroying control units
- Notification of changes in a device's (control unit's) state
- Maintenance of persistent information about its devices

153. The Remote Manager's Generic Device Manager may distribute devices among management servers on the hardware platform and among separate hosts of a single management server. (Exhibit 43)

154. Defendants' Generic Device Management Framework introduces a general paradigm for management of miscellaneous devices and/or manageable resources. The framework

allows for extending the types of Remote Manager-managed devices and resources with minimum efforts—an advantage of the claimed subject matter—and provides the device operator with a central point for administration of the manageable devices and resources. The figure below represents the Generic Device Management framework (Exhibit 45):



155. As shown in the figure above, an integrator of a new device type in Remote Manager need only to implement a Device Adapter, running on a Remote Manager management server for the device type. Higher level applications will see the new device in the same way as the devices of the already existing types without the need to develop a new GUI to access the functionality and control the features of the new device.

156. According to Defendants, a Device Adapter is based on the Bosch SI control unit model and can provide representation of the very device (implemented in a Device Control Unit Provider) as well as of specific resources available on the device (implemented in a number of Component Control Unit Providers). (Exhibit 45)

157. Scripting provides automatic generation of classes, properties, and methods. The GDM Script Service gives the opportunity to manage devices via scripts in the Remote Manager framework. It follows the idea that devices and their components (software components,

configurations, user settings, etc.) are presented with control units (CU), where the control unit has state variables (properties) and actions (methods), that can be executed. The conception of the GDM script service is to ease the use of control units so that bulk tasks on devices can be performed. This is achieved by dynamically generated classes that represent the objects of a concrete control unit type. (Exhibit 46)

158. In normal operation of accused instrumentalities using the Bosch Software Innovations and PTC combined IoT technology stack, distributed devices are analyzed and information models derived (e.g., according to the Eclipse Vorto standard in the integrated architecture of the Bosch IoT Suite and the ThingWorx IoT application enablement platform) to provide an abstract definition of the devices to be integrated. (Exhibit 5)

159. The OSGi-based gateway software, ProSyst mBS, leverages these interface definitions to enable communication to and from the devices and integrate them with the central backend via the Bosch IoT Remote Manager. The Production Rules Configurator provides a rules engine and can be used to help automate complex decisions. The ThingWorx application enablement platform is used to build dynamic API and UI-centric applications, which in turn leverage the ThingWorx core solution model. The Vorto-based code generator can generate a ThingWorx model, creating a fully realized API that is ready for application construction. (Exhibits 5 and 47)

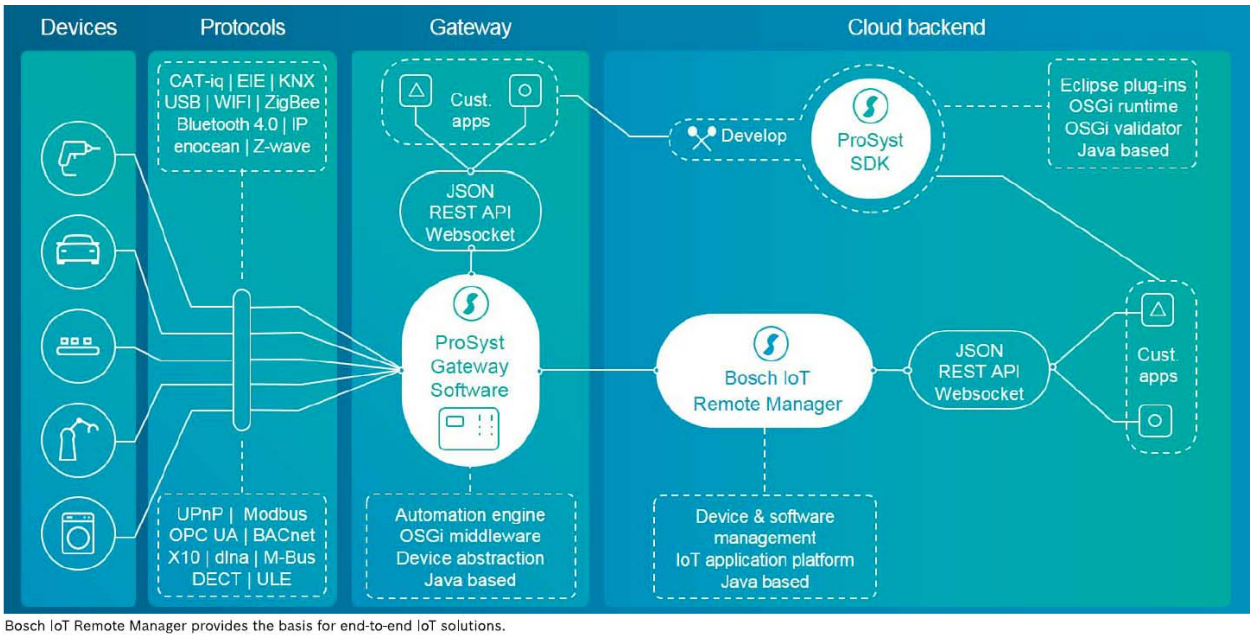
160. In a white paper titled “Addressing the real-world challenges of IoT solution development,” Defendants describe how the BIOTS Remote Manager generate software services to represent active objects (Exhibit 5):

The Bosch IoT Suite provides strong support for remote management of heterogeneous devices and assets. As just one of the cloud services offered in the Suite, the Bosch IoT Remote Manager allows developers to wrap the often very low-level and technical asset and device interfaces in a common language. The Suite additionally provides capabilities for software lifecycle management operations, firmware over-the-air updates, remote configuration and provisioning, as well as diagnostics. It also serves as the foundation for the implementation of rich IoT applications with sophisticated user interfaces, for which the ThingWorx rapid application enablement platform can be used.

161. BIOTS active objects include devices comprising an actuator for switching, controlling, and automating an IoT system. Actuators receive low-level commands from BIOTS middleware (e.g., Remote Manager, Things Service, ProSyst gateway software, etc.) that are converted from application commands (see, e.g., Exhibit 40).

Meta Model for Device Abstraction

- Unified language to describe devices based on Eclipse Vorto (see http://www.eclipse.org/vorto/)
- Describes structures and functionality of devices
- Leverage device description for contract-based development of large-scale or cross-domain solutions
- Basis for implementation of device drivers and device-specific protocols as well as code templates for solution development



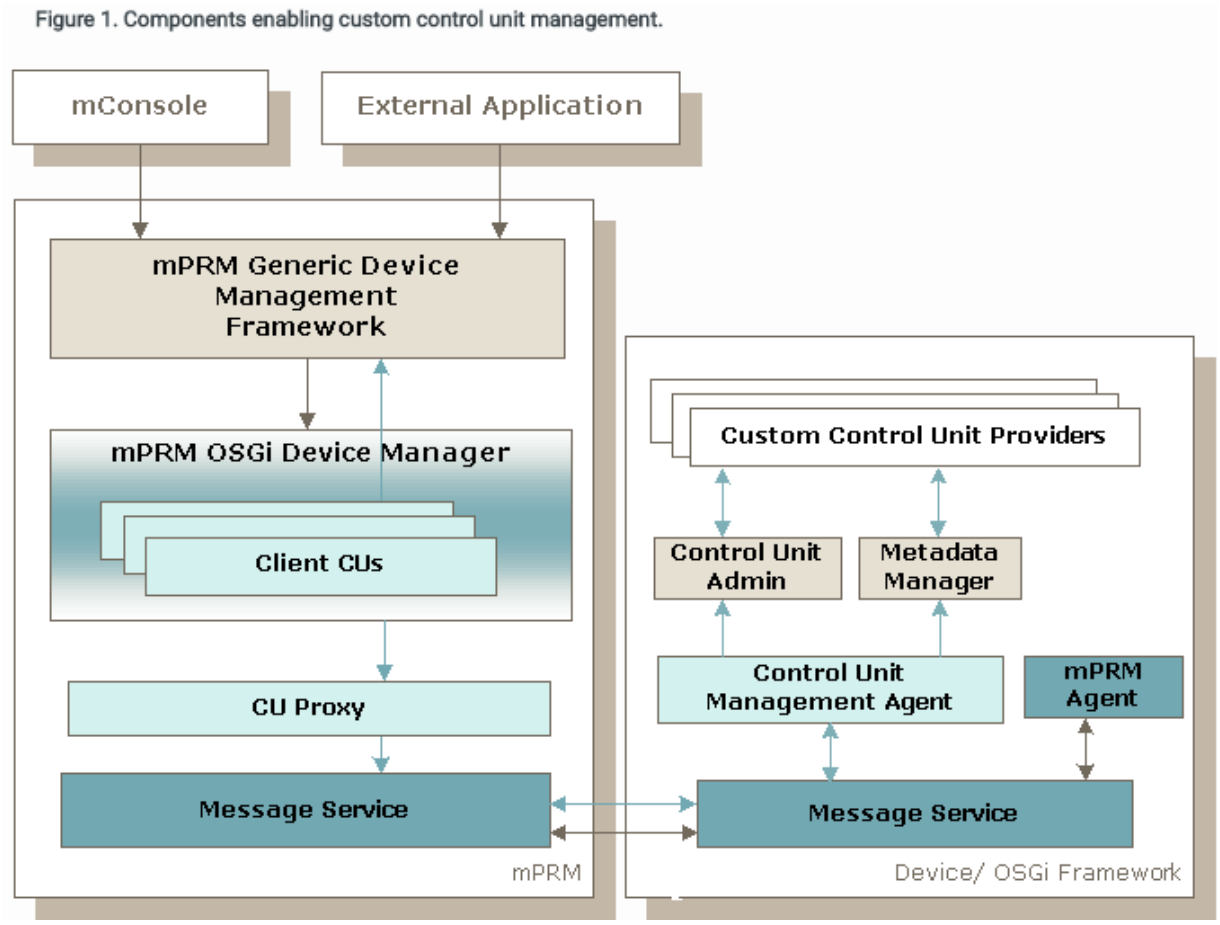
Bosch IoT Remote Manager provides the basis for end-to-end IoT solutions.

162. Through BIOTS Remote Manager middleware, for example, asset management and

control functions include device data collection and remote device control. (Exhibit 41)

163. Remote Manager provides OSGi and generic device management. The Control Unit Abstraction is the heart of the Remote Manager device management mechanism. The Remote Manager Generic Device Management Package provides utilities for representing each manageable unit (device or any other resources) as a set of control units, therefore enabling application to access them through the generic APIs of the Remote Manager. (Exhibit 48)

164. The following picture shows the components enabling custom control unit management:



165. BIOTS Remote Manager Driver Locator allows suitable driver bundles to be located in the Remote Manager software repository through the Remote Manager Driver Locator,

to be downloaded, and installed on demand, when new devices are detected.

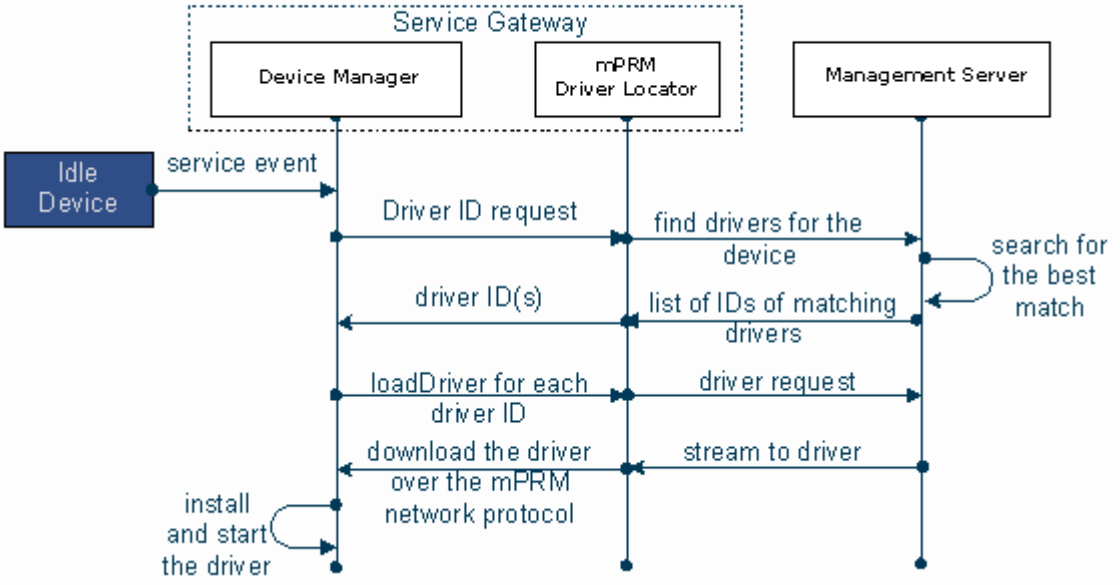
166. The Remote Manager Driver Locator is compliant with the OSGi Device Access Specification.

167. In normal operation, BIOTS driver locator middleware service is responsible for attaching a suitable driver to a newly plugged device under the control of the device manager. When a new device is detected, its properties are compared with the driver properties in the software repository. The driver whose properties correspond to the service registration properties of the particular device will be instructed by the device manager to attach to the device object. When the matching driver bundle is started on the device framework, it enables device registration. (Exhibit 49)

168. A Driver Locator encapsulates the knowledge of how to fetch the Driver bundles needed for a specific Device service through a matching filter. The Remote Manager Driver Locator is a driver locator implementation which uses the Remote Manager management server for driver location and download. (Exhibit 49)

169. Defendants provide the following ladder diagram showing the Remote Manager driver locator service in operation (Exhibit 49):

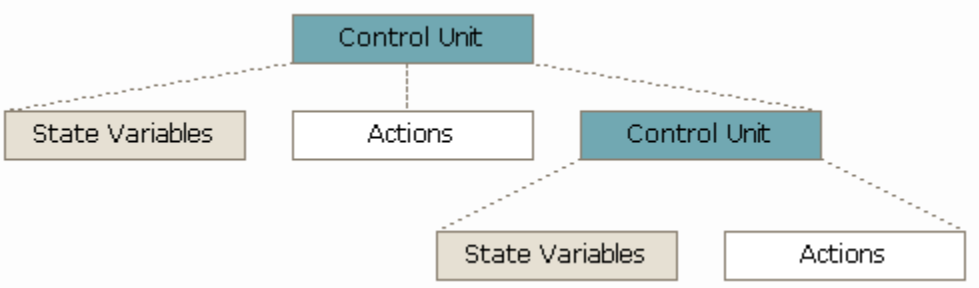
Figure 1. Remote Manager matching algorithm.



170. BIOTS Remote Manager uses control unit abstraction to provide a common interface to manage different types of devices. (Exhibit 44)

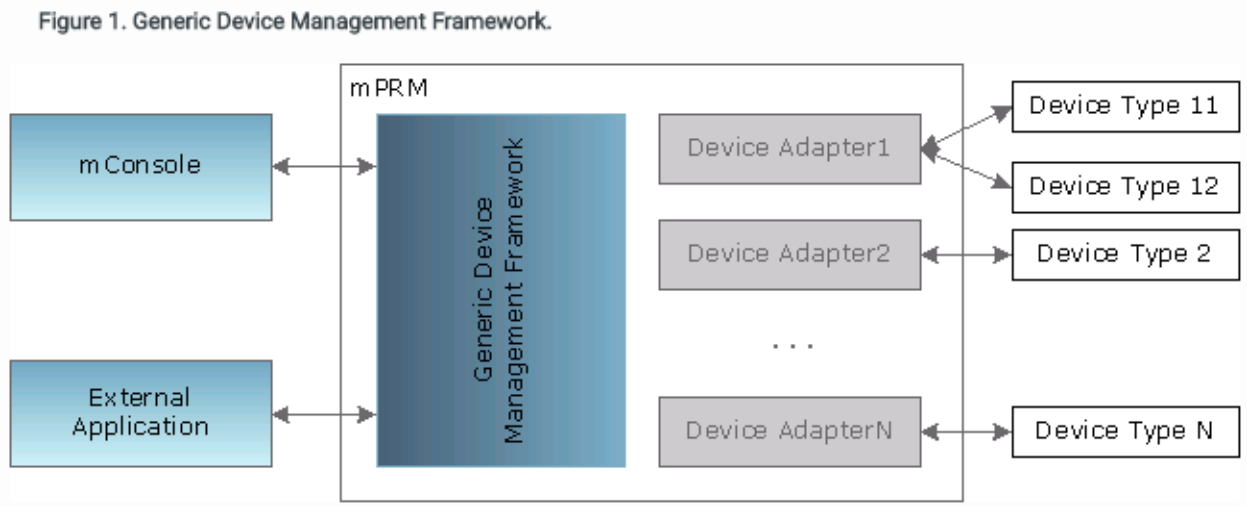
171. A control unit is a collection of state variables and actions. State variables represent control unit's specific parameters, for example an X10 lamp may have a state variable for indicating on and off, and have unique ID within the control unit scope and value of certain type. The actions of a control unit are the commands that the control unit can execute, such as turning a lamp on and off. An action has ID, unique within the control unit scope, and may have input and/or output arguments. (Exhibit 44)

172. Control units may be structured as shown below (Exhibit 44):



173. According to Defendants, "control unit abstraction provides a unified, well-known way to monitor the state of and execute an action on a control unit resulting in a state change . . . applications can manage control units in a uniform way, without the need to have specific knowledge about the underlying resources." (Exhibit 44)

174. Thus, in normal operation, BIOTS middleware provides an application-device interface for commands to active objects (actuators) and data from active objects (sensors) via the hardware platform as shown in an exemplary embodiment below:



175. OSGi is one example of a standard, uniform interface with which BIOTS software services comply to achieve interoperability. (Exhibit 50)

176. A stated goal of the OSGi Device Abstraction Layer (RFC 196, initial draft pub. Jan. 22, 2013) is to make every possible device accessible within the OSGi environment via one single standardized interface without regard to the communication protocol employed. The Device Abstraction Layer ("DAL") also supplies notification mechanisms that can be used to monitor the status of devices, the data model, and operations.

177. The OSGi DAL addresses the same problems that Dr. Helal and his co-inventors identified and addressed in the asserted patents: developers need only program with a view to a

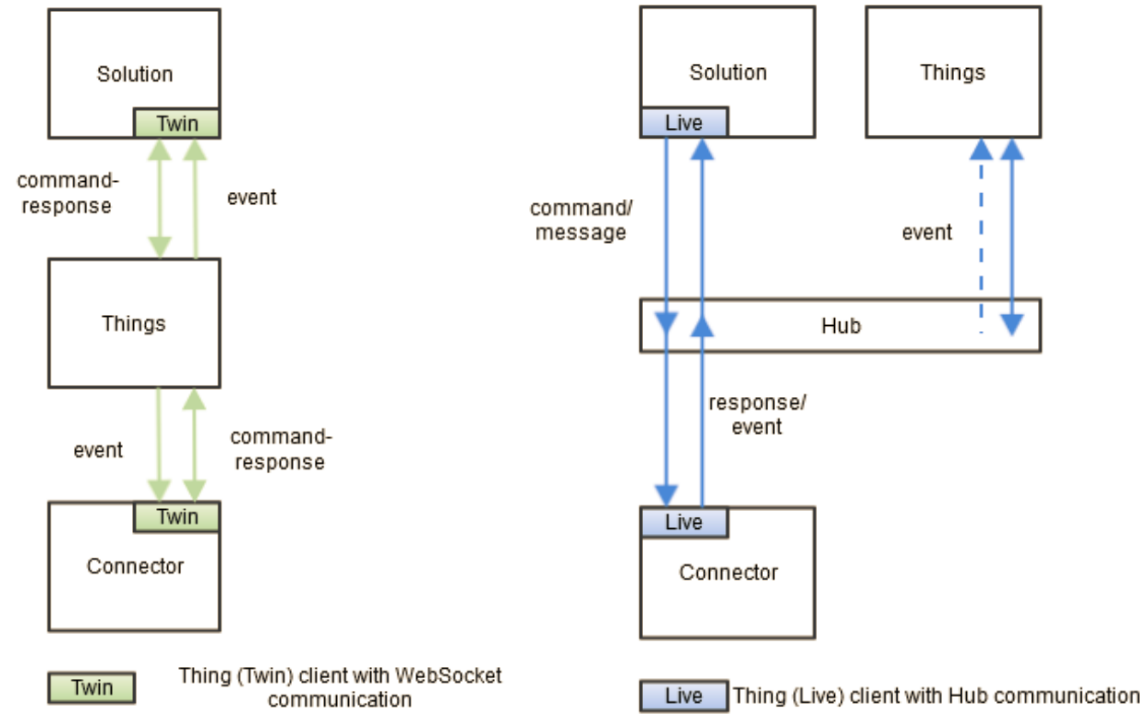
single interface and do not have to deal with protocol-specific problems and details. The DAL obviates the need for developers to build their own proprietary abstraction layers.

178. BIOTS middleware receives application commands written in high-level language (e.g., Bosch IoT Things HTTP API, Groovy, etc.) while the digital representation (twin) managed with the Things Service receives application commands at the HTTP API and the WebSocket interface via BIOTS Things Protocol. (Exhibits 51 and 52)

179. Bosch IoT Things Protocol covers two communication channels to address different aspects of devices and their digital representation. (Exhibit 53)

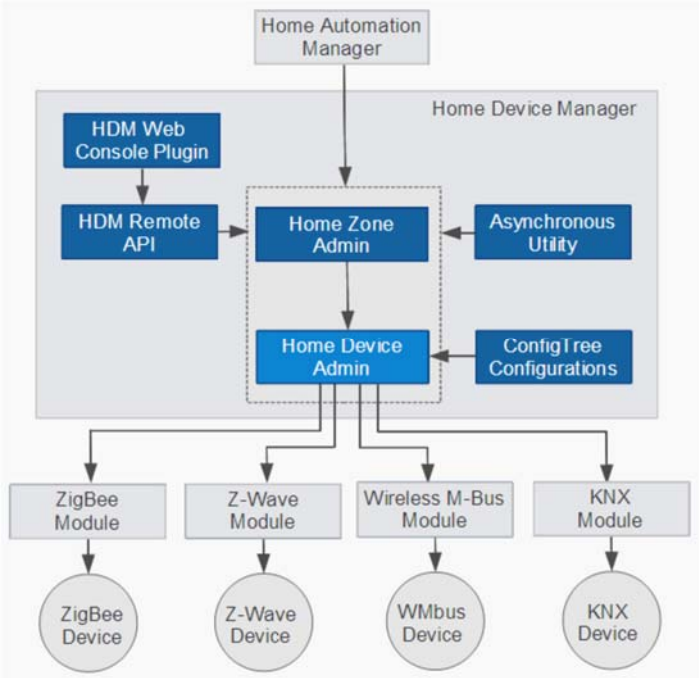
180. The first aspect handles the digital representation of an IoT asset, like a device. This asset, or Thing, is managed with the Things service and its state and properties can be read and updated. The channel to work with the digital representation is called twin. This channel is available both at the Bosch IoT Things HTTP API and the WebSocket interface which talks the Bosch IoT Things Protocol.

181. A command can be sent to the Things service to request a modification of a thing. When the Things service handled the command successfully, i.e. the updated thing is persisted, the service publishes an event. An event is the unit that describes a modification of a thing, e.g. a property change, or an attribute change. When sending a command, a response can be requested. The Things service asynchronously replies to such commands as soon as the change has been applied. Commands, events and messages are directly exchanged between live clients. A message carries a custom payload and can be answered by another, correlated message. Commands are defined to be used to change properties of e.g. connected device. In case a response to a command is requested, the receiver must fulfill this request. It is also always required that a live thing event is published after a command has been applied (to the device) successfully. (Exhibit 53)

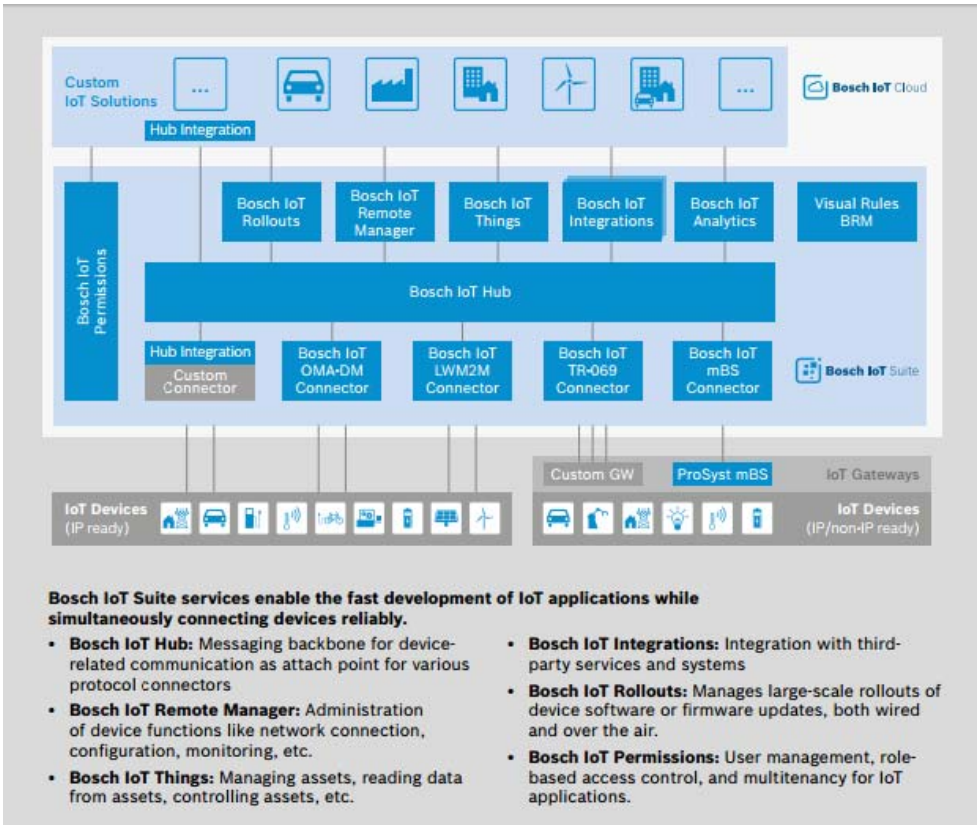


182. BIOTS middleware converts commands into low-level commands that can be executed by active objects. For example, for smart homes, the Home Device Manager transmits Z-wave commands to devices.

> The Home Device Manager is an abstraction layer that sits between the application that is used for controlling devices and the various protocol drivers (such as the ZigBee, Z-Wave, Video Cameras, etc) that are capable of communicating with these devices.



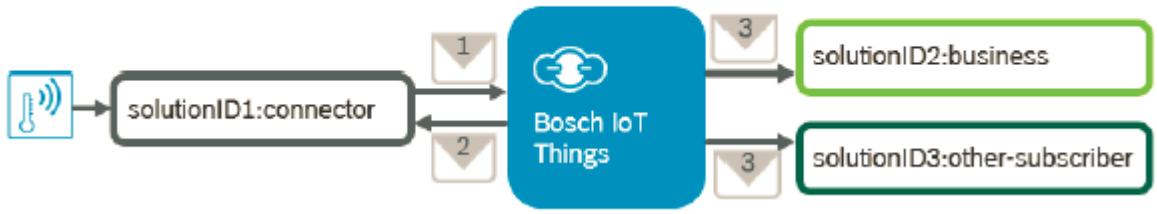
183. Bosch IoT Hub middleware communicates application commands to IoT devices via connectors (Exhibit 8):



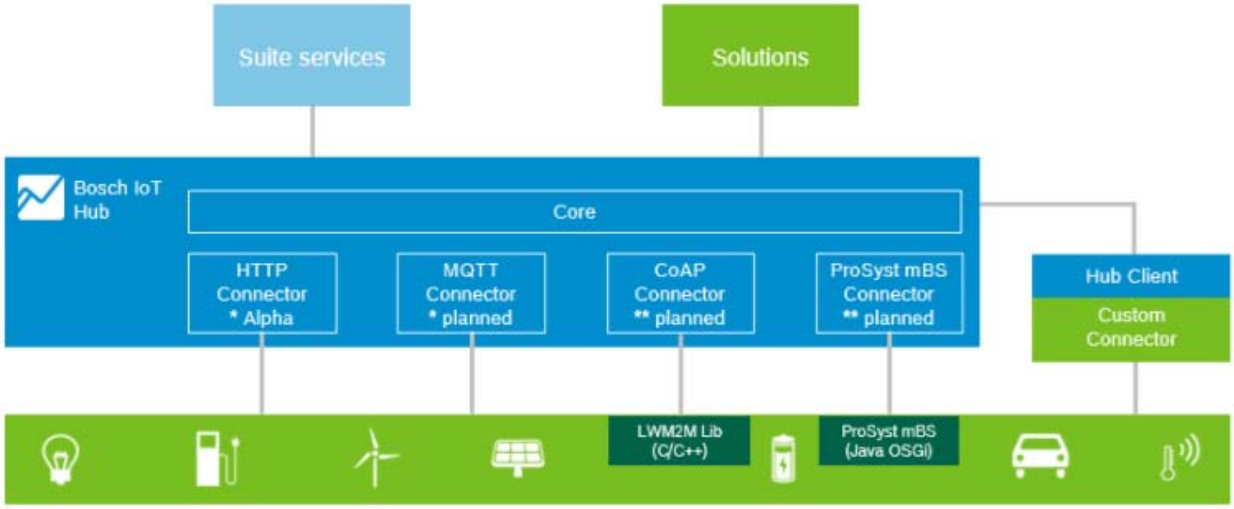
184. The Bosch IoT Hub allows reliable and secure messaging for device-related communication between services in an IoT platform. (Exhibit 54)

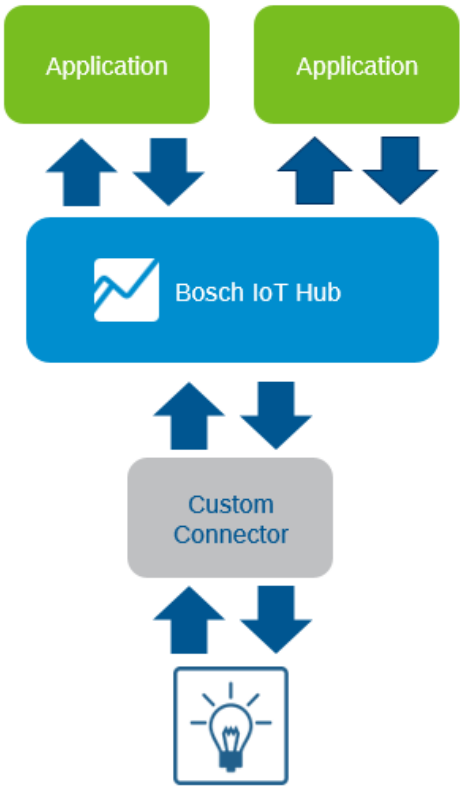
185. In general, communication flow between an application and active object through BIOTS middleware is shown schematically below. (Exhibit 55)

A schematic view for the communication flow:



186. Middleware connectors in the IoT Hub are shown schematically in the figures below. They exchange messages between applications and active objects. The connector of a sensor for example can communicate its measurements to interested cloud applications and send messages in order to remotely control devices. (Exhibit 54)





187. Messages may be used to retrieve a sensor value, for example, using the Thing Protocol Envelope. (Exhibits 56 and 57)

General protocol structure and envelopes

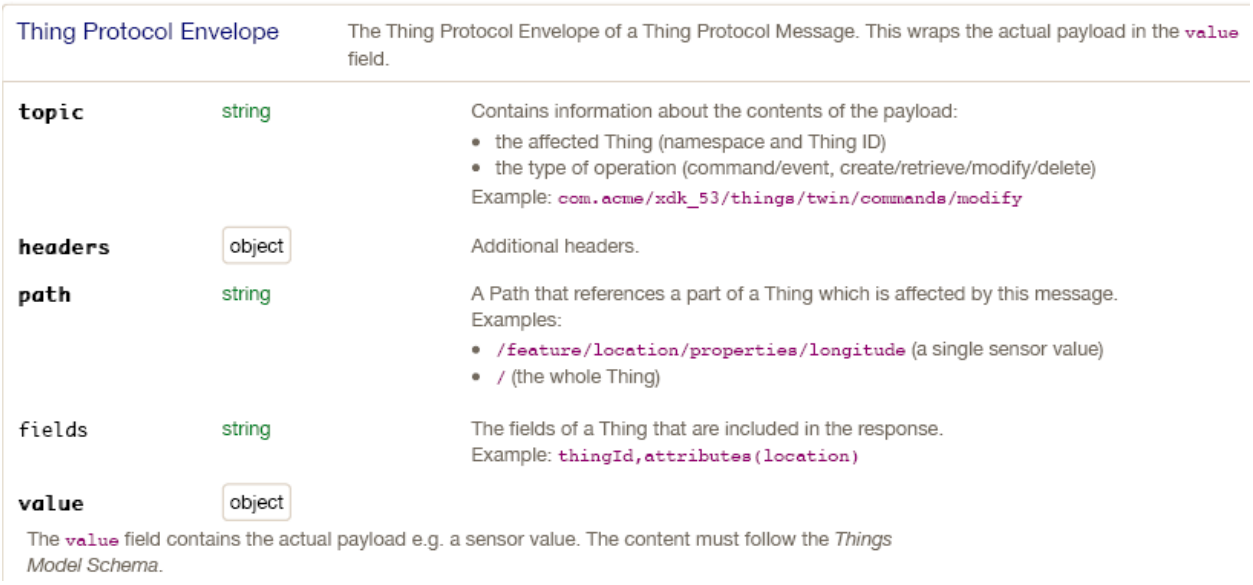
In order to comply with our Bosch IoT Things protocol, a message must consist of the following three parts:

- A communication protocol envelope (e.g. AMQP, WebSocket)
- A Things protocol envelope (JSON)
- A Things protocol payload (JSON)

Things protocol envelope

The Things protocol envelope describes the content of the message (the affected thing entity, a message type, protocol version etc.) and allows the message to be routed by intermediary nodes to its final destination without parsing the actual payload.

The protocol envelope is formatted as JSON (http://www.json.org/) (`content-type=application/json`) and must correspond to the following JSON schema:

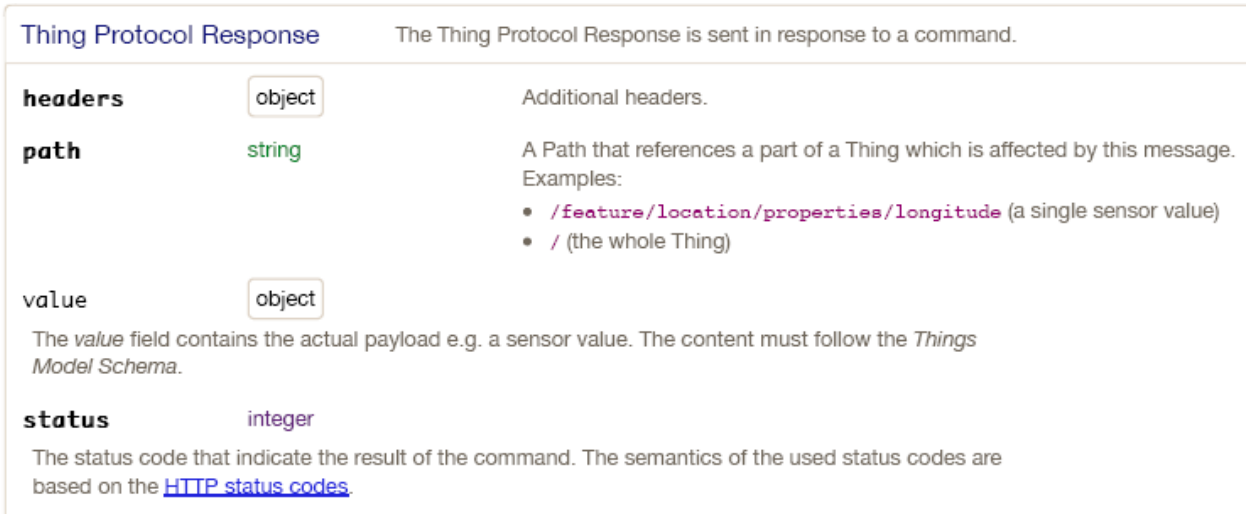


Thing Protocol Envelope — The Thing Protocol Envelope of a Thing Protocol Message. This wraps the actual payload in the `value` field.

topic	string	Contains information about the contents of the payload: • the affected Thing (namespace and Thing ID) • the type of operation (command/event, create/retrieve/modify/delete) Example: `com.acme/xdk_53/things/twin/commands/modify`
headers	object	Additional headers.
path	string	A Path that references a part of a Thing which is affected by this message. Examples: • `/feature/location/properties/longitude` (a single sensor value) • `/` (the whole Thing)
fields	string	The fields of a Thing that are included in the response. Example: `thingId,attributes(location)`
value	object	

The `value` field contains the actual payload e.g. a sensor value. The content must follow the *Things Model Schema*.

188. The Things model payload may contain application data (e.g., an updated sensor value):



Thing Protocol Response — The Thing Protocol Response is sent in response to a command.

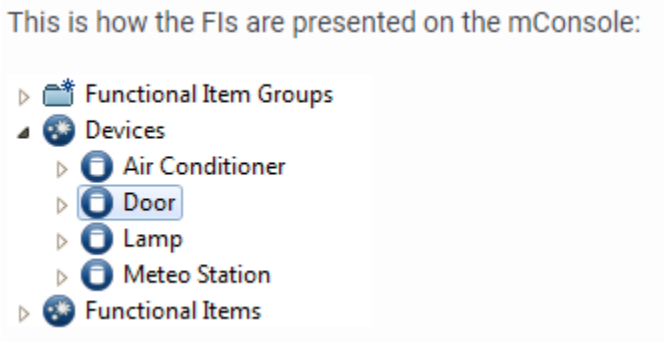
headers	object	Additional headers.
path	string	A Path that references a part of a Thing which is affected by this message. Examples: • `/feature/location/properties/longitude` (a single sensor value) • `/` (the whole Thing)
value	object	

The *value* field contains the actual payload e.g. a sensor value. The content must follow the *Things Model Schema*.

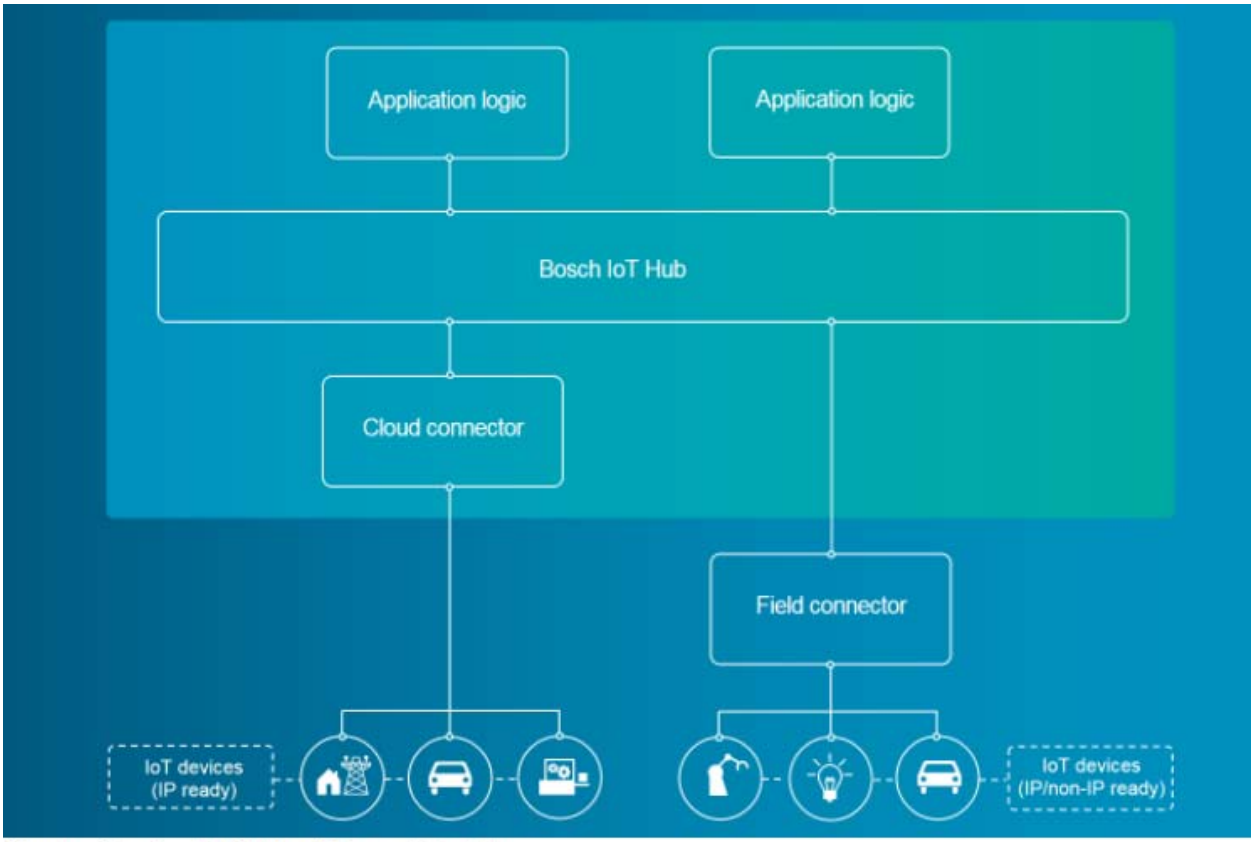
status	integer	

The status code that indicate the result of the command. The semantics of the used status codes are based on the HTTP status codes.

189. BIOTS actuators may be represented as client control units in the Remote Manager. These functional items are presented on the mConsole and managed through the Functional Item Management feature. (Exhibit 58)



190. Bosch IoT Hub enables IoT solutions to send and receive device related messages between services in the cloud, supports flexible message exchange patterns, data ingestion, and command and control messages. (Exhibit 59)



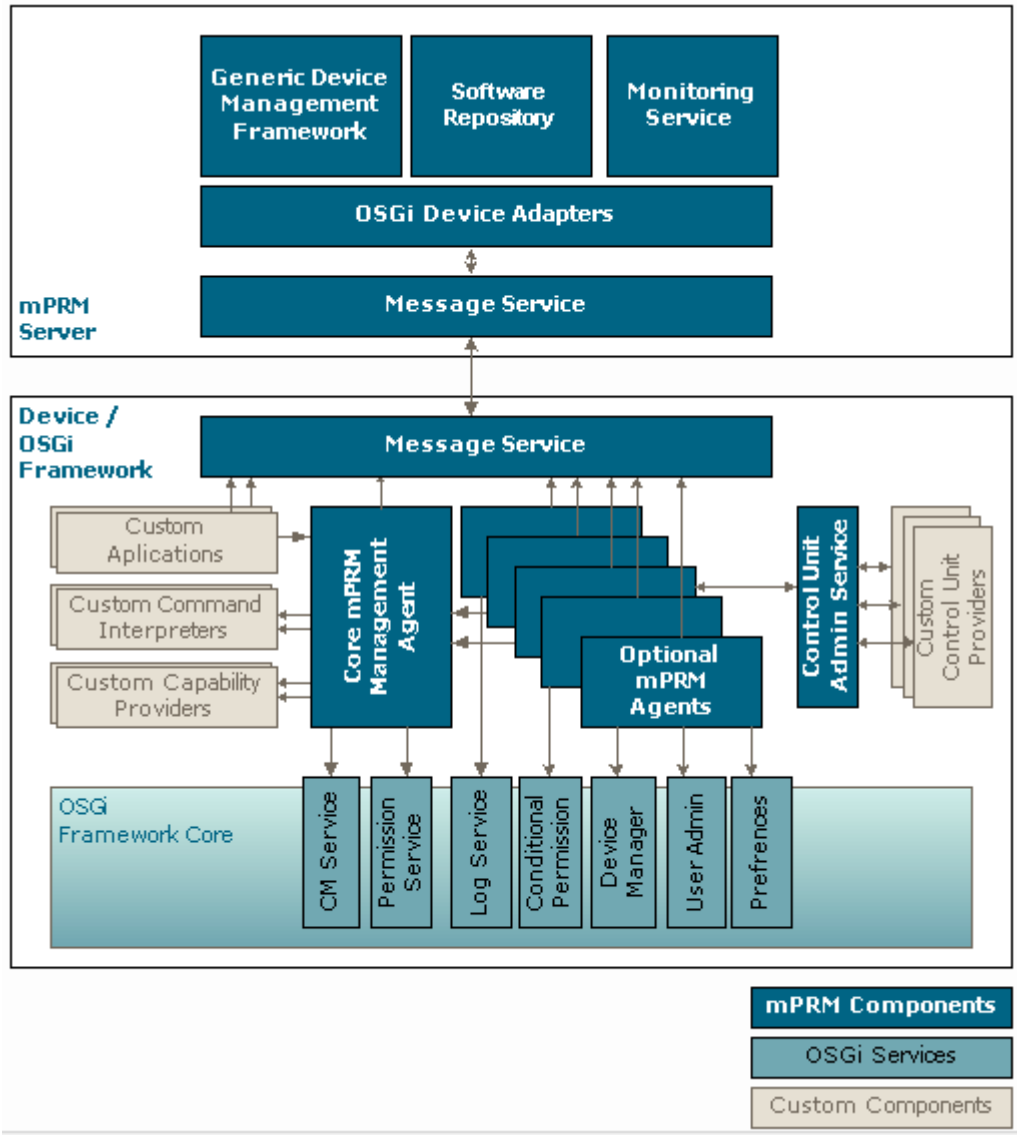
The role of the Bosch IoT Hub within your IoT solution.

191. Defendants' IoT Hub communicates with active objects via their abstract representatives referred to as "Connectors." (Exhibit 59)

192. The Bosch IoT Hub service contributes to relieving the strain on the integration

layer, by decoupling your business solutions from the technical device integration. (Exhibit 59) Connectors implement various transport protocols (i.e., lower level language) to enable message exchange with active objects, and responses or telemetry data received from devices are, in turn, dispatched by the Hub service to the application (i.e., a higher level language understood by the application).

193. Management of remotely managed devices is enabled through a set of components called Remote Manager agents that are activated as bundles on the target devices. These components are delivered and installed on the device during the initial provisioning of OSGi devices. The figure below illustrates the various components enabling OSGi device management. (Exhibit 60)



194. The Bosch documents referenced in this complaint accurately describe Defendants' products, services, and accused instrumentalities.

D. THE BOSCH DEFENDANTS PRACTICE THE ASSERTED CLAIMS

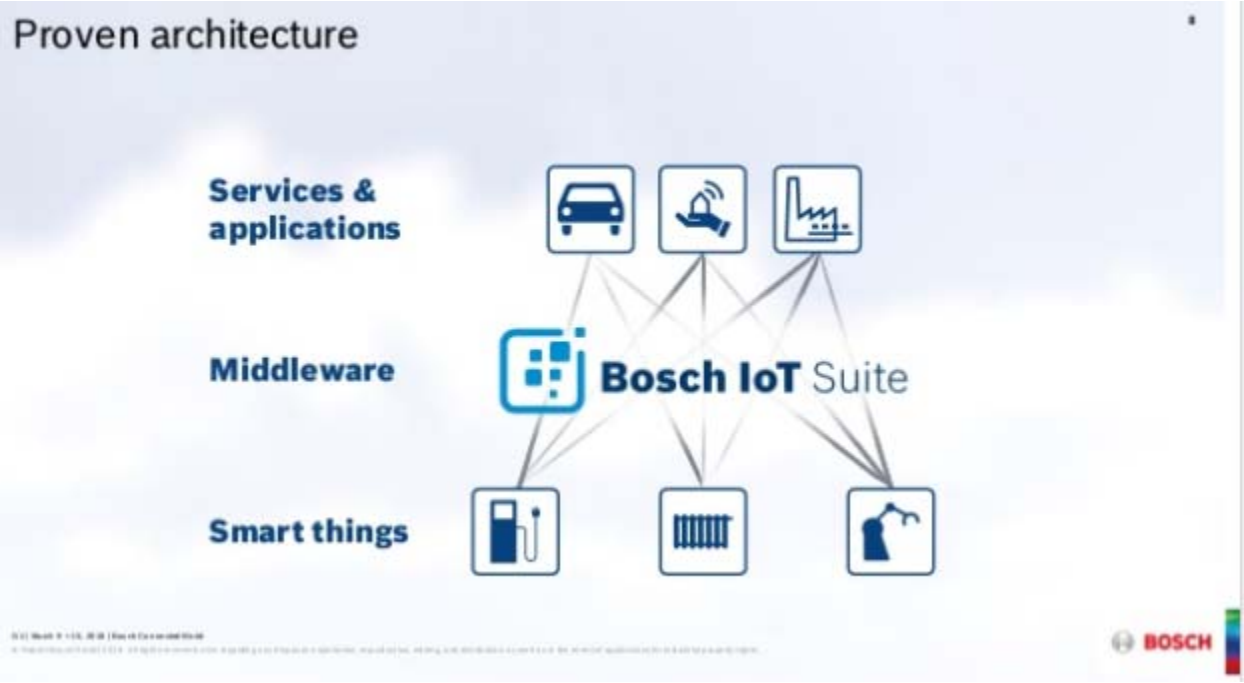
195. BIOTS meets each and every limitation of the asserted claims.

1. Hardware, Middleware and Software Services

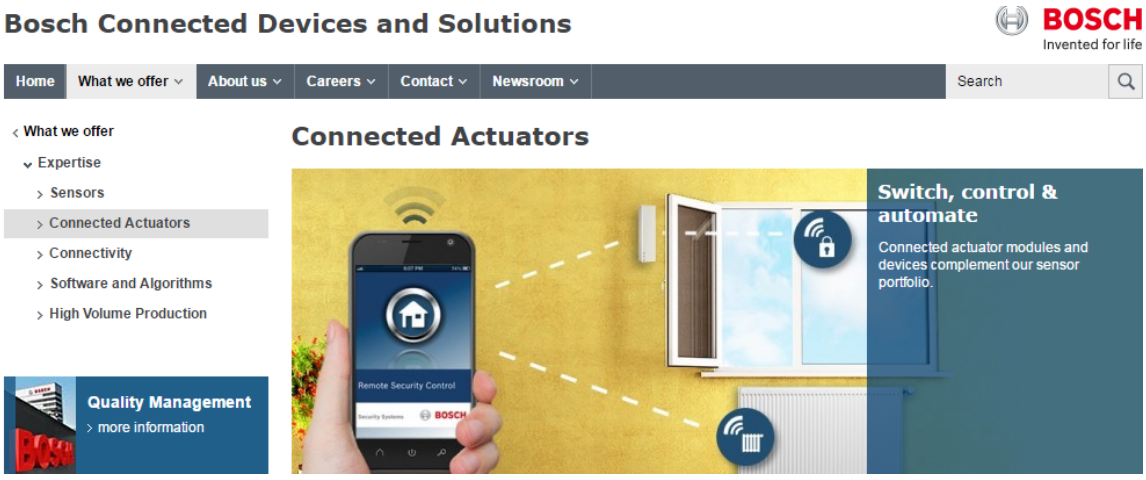
196. Hardware Platform with Middleware Module. Claims 1, 16 and 20 of the '063 Patent and claims 1 and 17 of the '257 Patent cover systems with a middleware module (e.g., software module) executing on a hardware platform. The hardware platforms covered by these

claims, including those covered by claim 7 of the ’257 Patent, communicate with active objects (e.g., devices that have sensors or actuators).

197. BIOTS IoT Suite is middleware:



198. Systems embodying claim 1 of the ’063 Patent interact with active objects that that have actuators. The accused systems include devices comprising an actuator such as those provided by Defendants.

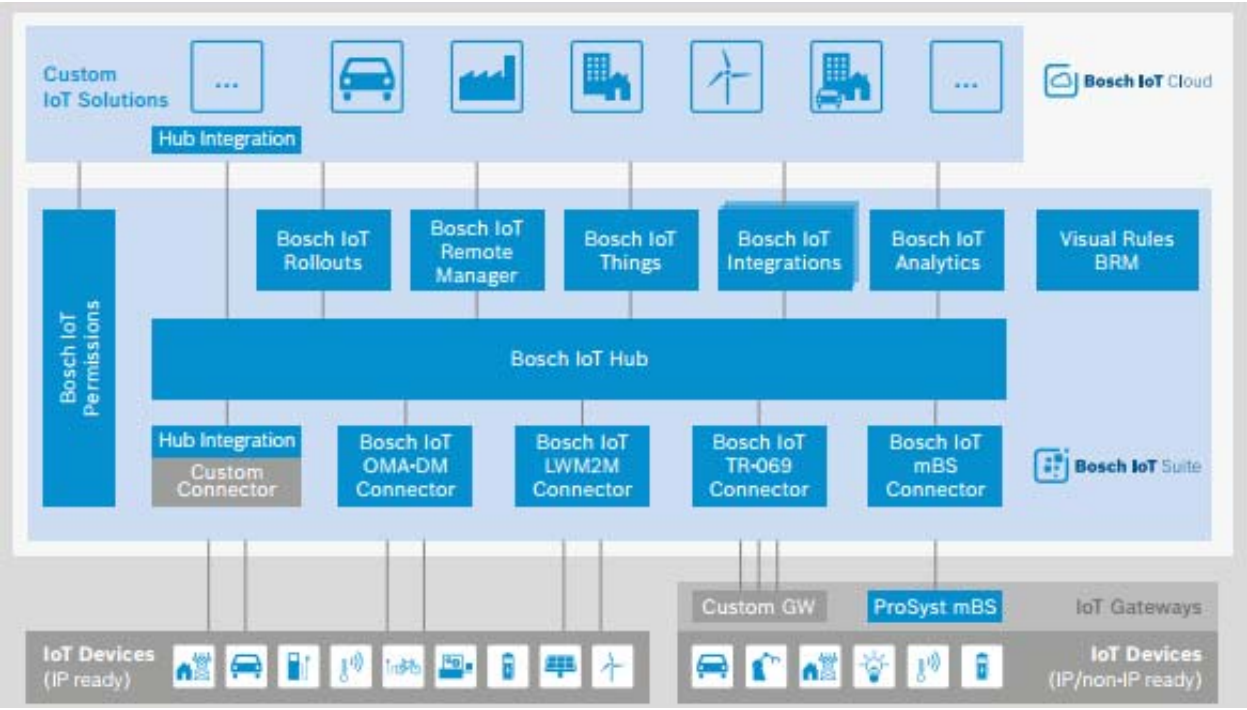


199. IoT systems embodying claim 1 of the ’257 Patent are adapted to communicate

with or interact with an active object with a sensor and another active object with an actuator, and the systems of claims 16, 37, and 38 of the ’063 Patent and claim 3 the ’257 Patent interact with active objects having both sensors and actuators.

200. The accused BIOTS systems comprise a hardware platform (e.g., server, gateway, and computing infrastructure) on which at least one middleware module executes to enable communication with active objects having sensors and/or actuators. Specific instances and examples of these structures are described above and throughout this complaint and include the Bosch IoT Suite executed on the BIOTS hardware platform, which includes “cloud” server infrastructure, Bosch IoT Gatways (e.g., branded “ProSyst mBS”) and Bosch IoT Hub middleware.

201. BIOTS communicates with active objects having sensors and/or actuators such as measurement sensors to provide telemetry data (e.g., speed, temperature or pressure sensors) and actuator switches or relays (i.e,. controllers or machines) that perform a particular function (e.g., closing a circuit).



202. Examples of sensors and actuators in BIOTS active objects include machinery, connected automobiles, consumer goods and tools, and appliances:

Bosch Sensors

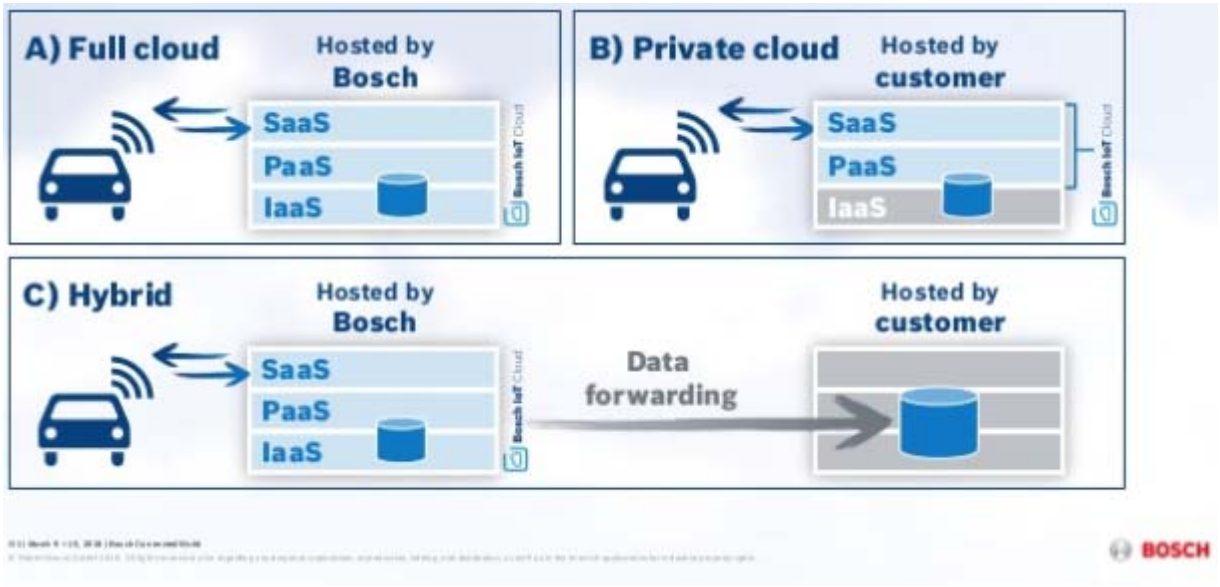
203. Middleware Module Location and Communication Module. Claim 12 of the '063

Patent and claims 13 and 18 of the ’257 Patent require that at least a portion of the middleware module resides on or is executed on a server connected to the hardware platform via a network.

204. Claim 13 of the ’063 Patent and claim 14 of the ’257 Patent cover systems that include a communication module as part of the hardware platform. A communication module is configured to connect the hardware platform with a server via a network.

205. Claim 14 of the ’063 Patent and claim 15 of the ’257 Patent cover systems that include a wireless connection between the server and the network.

206. The claimed structures and architecure are embodied in accused BIOTS systems, as described in detail above and throughout this complaint. BIOTS middleware is hosted by Defendants or on a customer-hosted hardware platform.



207. Bosch middleware (e.g., Bosch IoT Suite) executes on the BIOTS hardware platform. BIOTS gateway software executes on the BIOTS hardware platform and BIOTS gateway infrastructure.

2. Software service

208. Generate Software Services. In embodied systems of claim 1, claim 16, and claim

20 of the ’063 Patent and claim 1 of the ’257 Patent, for each active object, middleware generates software services that represent active objects. Similarly, in an embodied system of claim 16, 20, and 36 of the ’063 Patent and claims 1 and 17 of the ’257 Patent, software services are generated to represent active objects, where generating the software services is based on a received driver that includes information and behavioral components of the active objects.

209. Claim 17 of the ’063 Patent, claim 20 of the ’063 Patent, and claim 17 of the ’257 Patent similarly cover, for additional active objects, generating additional software services based on additional drivers that are received.

210. Claim 35 of the ’063 Patent and claim 32 of the ’257 Patent cover generating a second software service that represents an active object.

211. Accused instrumentalities generate software services for Things. (Exhibit 16).

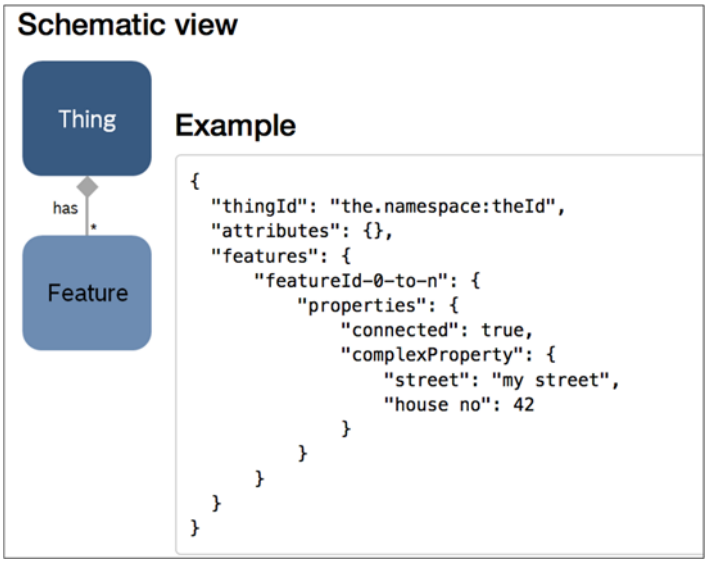
Thing

Things are very generic entities and are mostly used as a “handle” for multiple features belonging to this Thing.

Examples:

- Physical Device: a lawn mower, a sensor, a vehicle, a lamp
- Virtual Device: a room in a house, a virtual power plant spanning multiple power plants, the weather information for a specific location collected by various sensors
- Transactional entity: a tour of a vehicle (from start until stop), a series of measurements of a machine
- Master data entity: a supplier of devices or a service provider for devices, an entity representing a city/region
- Anything else - if it can be modeled and managed appropriately by the supported concepts/capabilities

212. Things have features. (Exhibit 16).



213. In an example Accused System, middleware executing on the Bosch IoT Hub or Bosch Remote Manager generates a Thing ID, which is a Java Object that represents an active object in the network and may have Features (identified by Feature ID) and Attributes (identified by Attribute ID). The Java Object is an examplary software service. A Feature is used to manage all data and functionality of a Thing that can be clustered in an outlined technical context. For different contexts or aspects of a Thing different Features can be used which are all belonging to the same Thing and do not exist without this Thing. Data related to Features is managed as lists of properties. These properties can be categorized, e.g. to manage the status, the configuration or any fault information. Each property itself can be either a simple/scalar value or a complex object. Allowed is any JSON object. See, generally, Bosch IoT Things Service aka Central Registry and Developer Guide; Exhibits 16-20, 22-23, 25, 34, 56-57 and 61.

214. Editing the Software Service. Claim 19 of the ’257 Patent covers systems that permit editing the software service. Claim 20 of the ’257 Patent permit editing the software service remotely.

215. A command can be sent to the Things service to request a modification of a thing.

When the Things service handled the command successfully, i.e. the updated thing is persisted, the service publishes an event. An event is the unit that describes a modification of a thing, e.g. a property change, or an attribute change. See, generally, Bosch IoT Things Service aka Central Registry and Developer Guide; Exhibits 16-20, 22-23, 25, 30, 34, 44, 51 and 56-57.

216. The Integration Client enables users to create and/or manipulate entities like Thing, Feature, Relation etc. programmatically. See, generally, Bosch IoT Things Service aka Central Registry and Developer Guide, Bosch IoT Things Java API;[5] Exhibits 16-20, 22-23, 25, 34, 56-57 and 61.

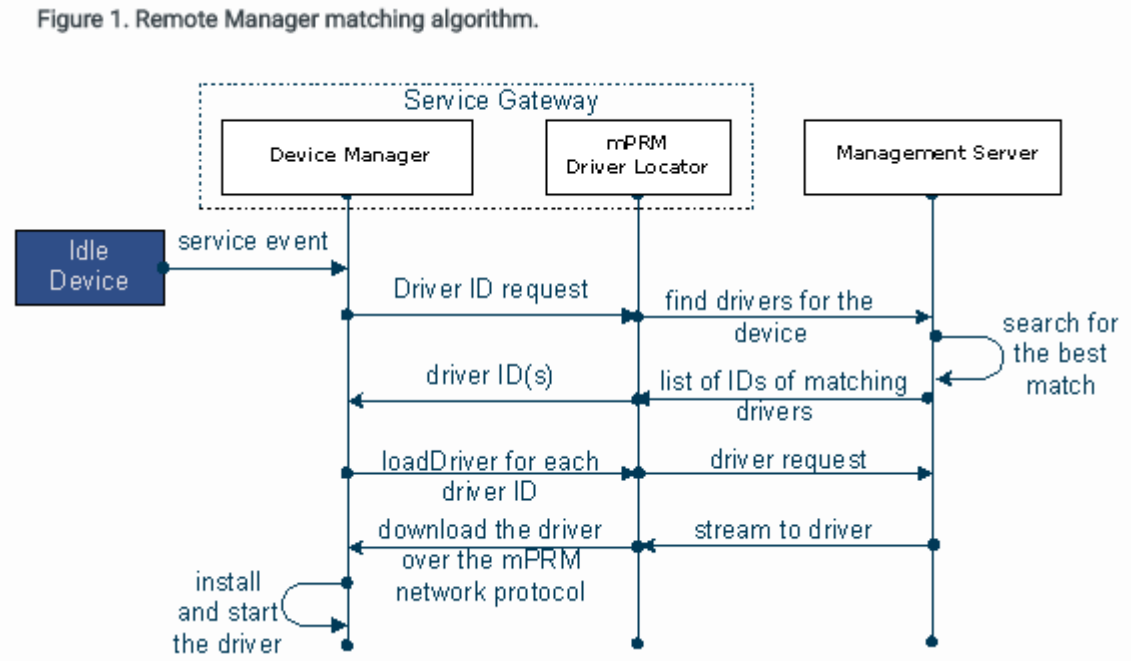
3. Drivers

217. Receiving a Driver. Claim 16 of the ’063 Patent, claim 20 of the ’063 Patent, and claim 17 of the ’257 Patent cover systems that perform method steps including receiving a driver that has information and behavioral components for interacting with an active object (e.g., actuator and/or sensor device) connected to a hardware platform.

218. In normal operation, BIOTS performs this step, for example, when an active object is connected and the BIOTS Remote Manager Driver Locator identifies the appropriate driver in the Remote Manager software repository, downloads and installs it.

219. BIOTS Remote Manager Driver Locator provides for receiving suitable driver bundles having information and configuration information for interacting with sensors/actuators. See, e.g., Exhibits 43-46, 48-49).

[5] https://things.apps.bosch-iot-cloud.com/dokuwiki/doku.php?id=005_dev_guide:005_java_api:005_java_api.



220. Driver Configuration Information. Claim 24 of the '063 Patent and claim 23 of the '257 Patent cover systems where the driver includes configuration information from the hardware platform. Further, for the systems, the configuration information includes indicia of the hardware platform (or interface of the hardware platform). The accused instrumentalities meet these limitations by providing back-end driver bundles and configuration information about the hardware platform and interface. For example, backend support is wrapped in a separate bundle referred to as the Driver Locator Handler, which includes information about the location of device drivers on the hardware platform. (Exhibit 49)

221. Driver Storage Location. Claim 25 of the '063 Patent and claim 24 of the '257 Patent cover systems in which the driver is stored on the hardware platform, the active object, the server, a local repository, or a remote repository. In BIOTS, drivers may be stored in the driver software repository on a server (Exhibit 49):

> **Remote Manager Driver Locator**
>
> The *driver locator* allows suitable driver bundles to be located in the Remote Manager software repository through the Remote Manager Driver Locator, to be

> downloaded, and installed on demand, when new devices are detected on the device. The Remote Manager Driver Locator is compliant with the OSGi Device Access Specification, therefore it provides integration between the OSGi-defined Device/Driver model and Remote Manager software delivery functionality.
>
> A Driver Locator encapsulates the knowledge of how to fetch the Driver bundles needed for a specific Device service through a matching filter. The Remote Manager Driver Locator is a driver locator implementation which uses the Remote Manager management server for driver location and download.

222. Driver Download. Claim 26 of the ’063 Patent and claim 25 of the ’257 Patent cover systems where the middleware module downloads the driver from a local repository or remote repository, based on resource location information received from the hardware platform or active object.

223. Claim 27 of the ’063 Patent and claim 26 of the ’257 Patent cover systems in which the resource location information includes a Uniform Resource Locator (URL) for locating the driver via the Internet.

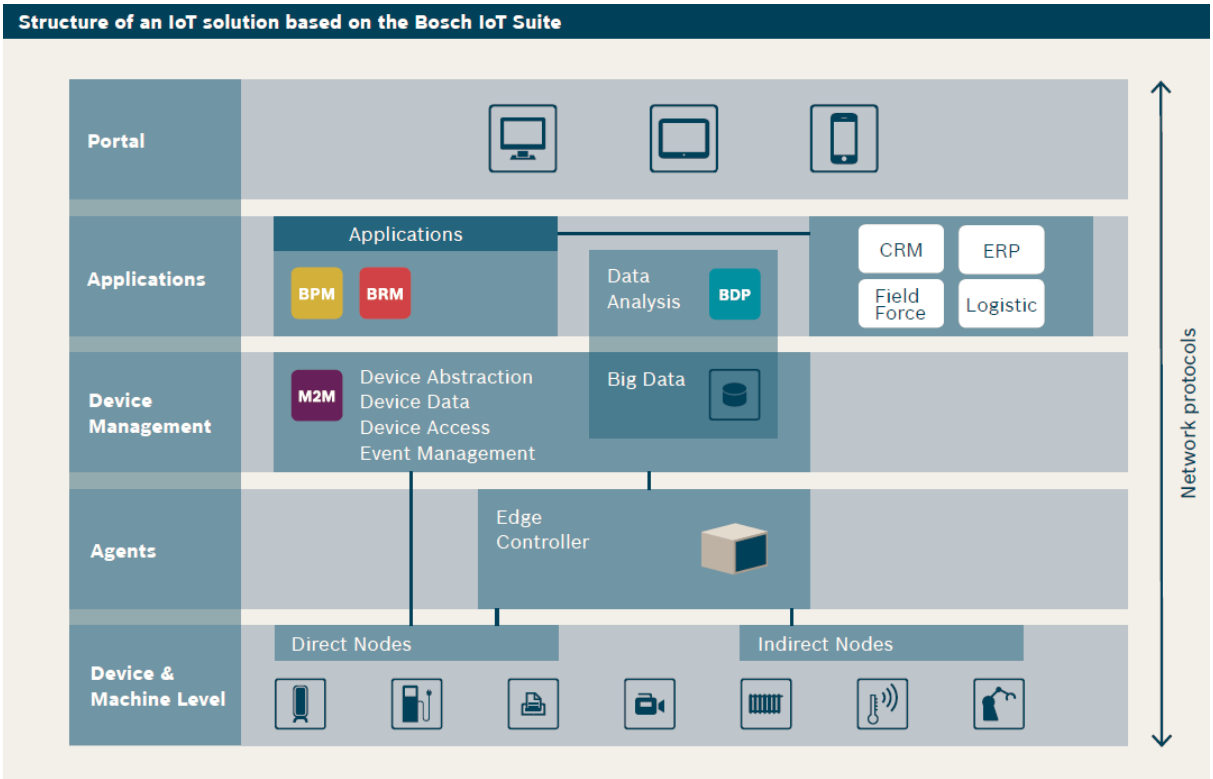
224. Accused BIOTS instrumentalities meet these limitations, for example, by providing provisioning services in connection with Remote Manager operation. Defendants’ OSGi device provisioning mechanism provides a procedure for supplying a device with all components (management agent bundles and configuration properties) necessary for management through a backend system. Resource location information may be a network location or URL. For example, a provisioning data provider value must contain the URL pointing to the backend server. See Bosch IoT Remote Manager – Initial Provisioning of OSGi Devices, and Exhibit 62.

225. Receiving an Additional Driver. Claim 17 and claim 20 of the ’063 Patent cover systems that receive an additional driver (that includes information and behavioral components required to interact with an additional active object). Remote Manager is used to provide device management services for updating device drivers. (Exhibit 62)

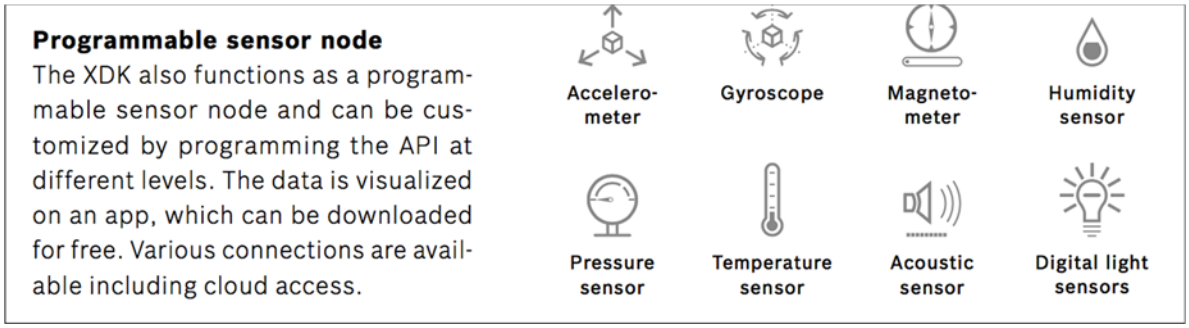
4. Commands

226. Receive High Level Commands. In an embodied system of claim 1 of the '063 Patent and claim 1 of the ’257 Patent, the middleware receives commands from higher level software applications via the software service that represents the active objects. Similarly, an embodied system of claim 16 of the ’063 Patent and claim 2 of the ’257 Patent receives (via the software service) one or more commands from an application written in a higher level language. Similarly, claim 18 of the ’063 Patent cover systems that receive (via an additional software service) one or more commands from a second application written in a higher level language.

227. Accused Systems include the Bosch IoT Suite receiving commands from higher level language applications via the software service that represents an active object. For example, as shown below, the Bosch IoT system receives commands from BPM or BRM via Device Management that represent a device through Device Abstraction. Bosch IoT integrates with third-party services and systems including higher level applications.



228. In Accused Systems, an XDK Node (described as "the universal programmable sensor device") operates with higher-level software packages for interacting with a software service (that represents an active object) as required by the claims.

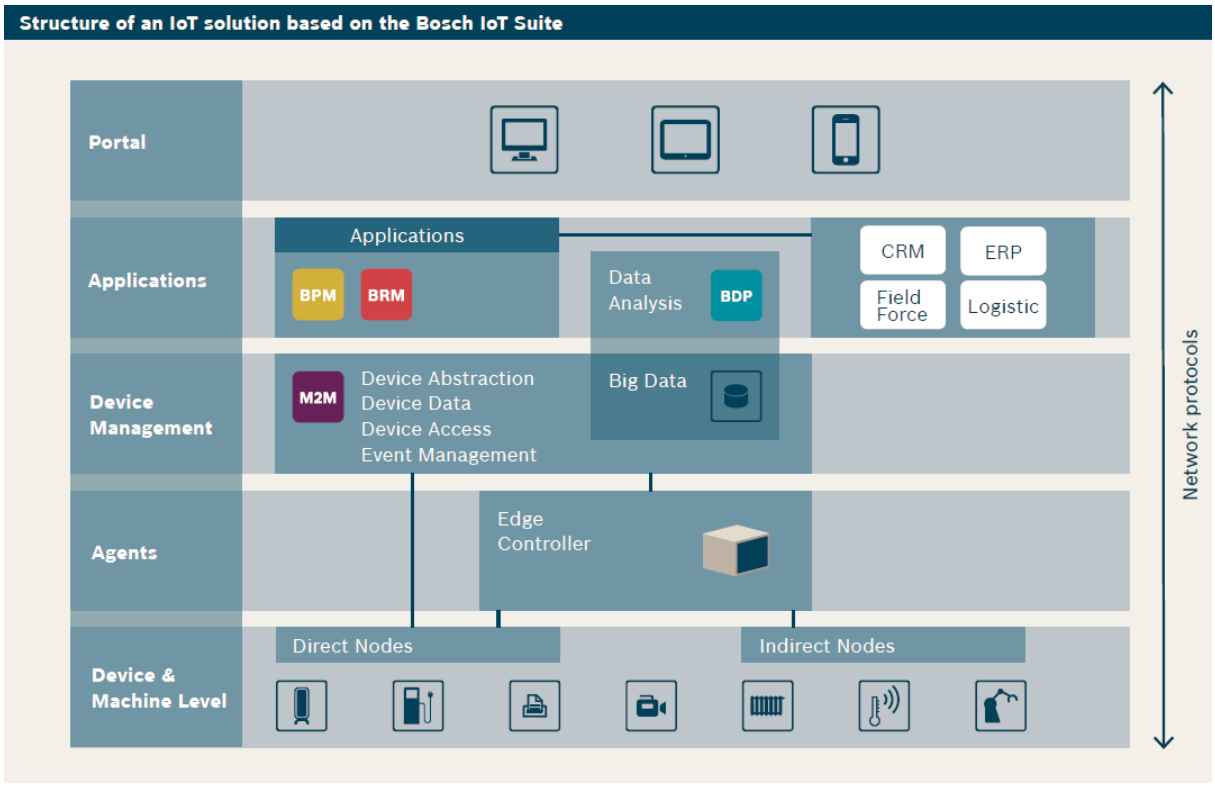


229. Convert to Low-Level Commands. In an embodied system of '063 Patent claim 1 and claims 1 and 2 of the '257 Patent, the middleware converts the commands into low-level commands that can be understood by the active objects. Similarly, an embodied system of claim 16 of the '063 Patent converts commands received from a higher level language application into more low-level commands capable of controlling the operation of the active object. Similarly, an

embodied system of claim 18 of the ’063 Patent coverts one or more commands (received via a software service and written in a higher level language) into low-level commands capable of controlling the operation of active objects.

230. Claims 37 and 38 of the ’063 Patent and claim 35 of the ’257 Patent cover systems in which one or more commands are received by a second software service and converted into one or more low-level commands capable of controlling the operation of an actuator.

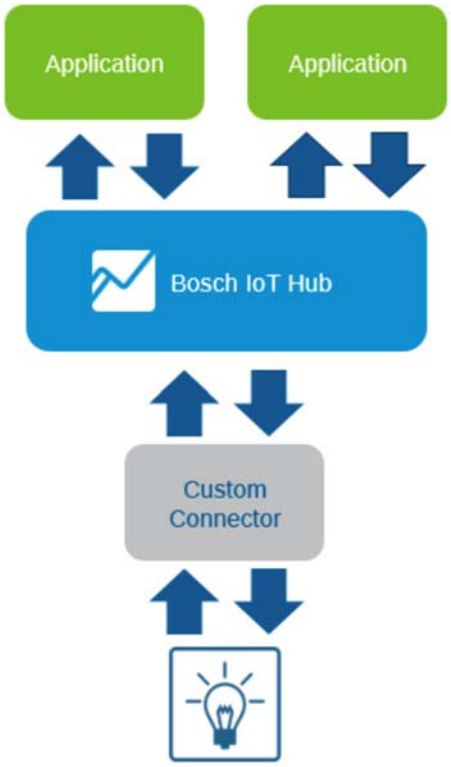
231. The Bosch IoT Suite receives commands from higher level language applications via multiple software services that represent active objects. For example, as shown below, the Bosch IoT system receives commands from BPM or BRM via Device Management that represent multiple devices through Device Abstraction. As discussed above, Bosch IoT also integrates with third-party services and systems including higher level applications.



232. Transmitting Low-Level Commands to Active Objects. Claims 18, 37, and 38 of

the ’063 Patent and Claim 2 of the ’257 Patent further require transmitting the low-level commands to active objects via the hardware platform.

233. The accused instrumentalities perform these steps. As an example, business applications in BIOTS send messages in order to remotely control devices, sending commands to sensors and actuators. (Exhibit 37-38, 40-42 and 44)



234. BIOTS Control Unit Providers communicate with devices to perform functions by command. (Exhibits 43, 53, 55 and 59)

235. Applications Written in Higher Level Language. Claim 1 of the ’257 Patent covers systems with one or more applications written in a higher level language systems. The applications are configured to receive usable data from at least one software service.

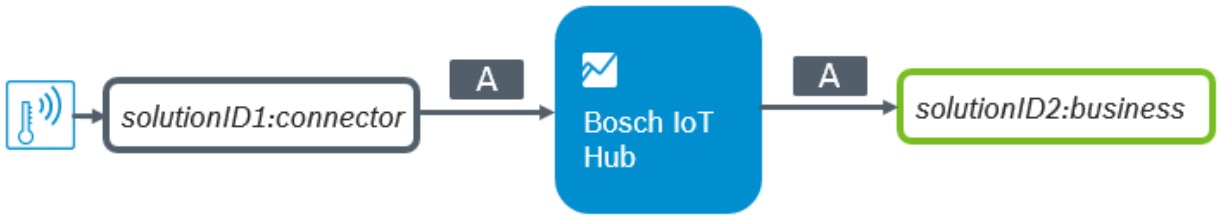
236. Convert to Low-Level Commands. In an embodied system of ’063 Patent claim 1 and claims 1 and 2 of the ’257 Patent, the middleware converts the commands into low-level

commands that can be understood by the active objects. Similarly, an embodied system of claim 16 of the ’063 Patent converts commands received from a higher level language application into more low-level commands capable of controlling the operation of the active object. Similarly, an embodied system of claim 18 of the ’063 Patent coverts one or more commands (received via a software service and written in a higher level language) into low-level commands capable of controlling the operation of active objects.

237. Claims 37 and 38 of the ’063 Patent and claim 35 of the ’257 Patent cover systems in which one or more commands are received by a second software service and converted into one or more low-level commands capable of controlling the operation of an actuator.

238. The Bosch IoT Hub enables exchange of device-related messages between and among services and devices and applications.

239. In a simplified example, low-level messages are transmitted between application, IoT Hub, connector, and device.



240. Bosch IoT Hub enables IoT solutions to uniformly and transparently send and receive device-related messages between services in the cloud and supports flexible message exchange patterns to enable data ingestion (telemetry) as well as command and control messages.[6]

241. Transmitting Low-Level Commands to Active Objects. Claims 18, 37, and 38 of the ’063 Patent and Claim 2 of the ’257 Patent further require transmitting the low-level commands

[6] https://hub.apps.bosch-iot-cloud.com/dokuwiki/doku.php?id=010_learn_about:010_learn_about&s[]=command

to active objects via the hardware platform. The accused intrumentalities have this functionality. The IoT Hub transmits low-level commands via connectors to active objects via the BIOTS hardware platform.

242. In BIOTS, and particularly in normal operation of the Bosch IoT Hub, sending a command message implies that a response message is correlated to the command message and sent back to the command sender. Command messages from other systems to a device, a group of devices, or other services to perform specific activities are service-initiated instructions sent to the device. BIOTS commands can tell a device to provide information about its state, or to change the state of the device, including activities by an actuator.[7]

243. Applications Written in Higher Level Language. Claim 1 of the '257 Patent covers systems with one or more applications written in a higher level language systems. The applications are configured to receive usable data from at least one software service.

5. Receiving and Processing Raw Data

244. Sensor: Raw Data Receipt and Conversion. Claim 2 of the ’063 Patent and claims 1 and 3 of ’257 Patent cover systems in which an active object includes a sensor. For claim 2 of the ’063 Patent and claims 1 and 3 of the ’257 Patent, the hardware platform receives raw data from the active objects and passes the raw data to the middleware module. In turn, for claim 2 of the ’063 Patent and for claim 1 of the ’257 Patent, the middleware module converts the raw data into usable data (usable by higher-level language applications) and passes the usable data to the software service for the active object.

245. For systems covered by claim 20 of the ’063 Patent and claim 3 of the ’257 Patent, raw data is received from the active object by the hardware platform and converted into usable

[7] https://hub.apps.bosch-iot-cloud.com/dokuwiki/doku.php?id=010_learn_about:functionality&s[]=command

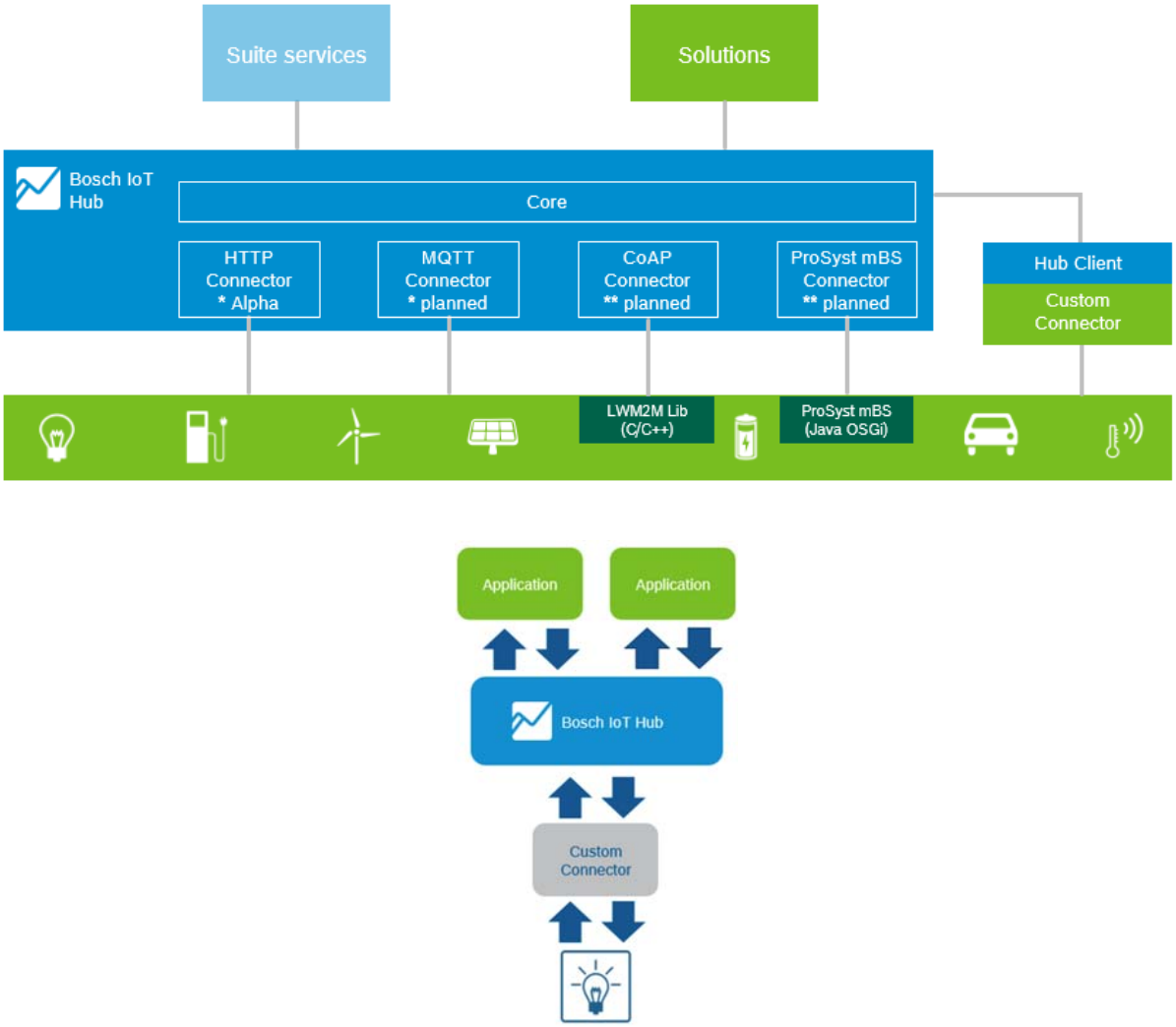
data. Further, the data is passed to the software service.

246. For systems covered by claim 30 of the ’063 Patent and claim 27 of the ’257 Patent, the middleware module converts the raw data to usable data. For systems covered by claim 31 of the ’063 Patent and claim 28 of the ’257 Patent, the usable data is passed from the software service to the application.

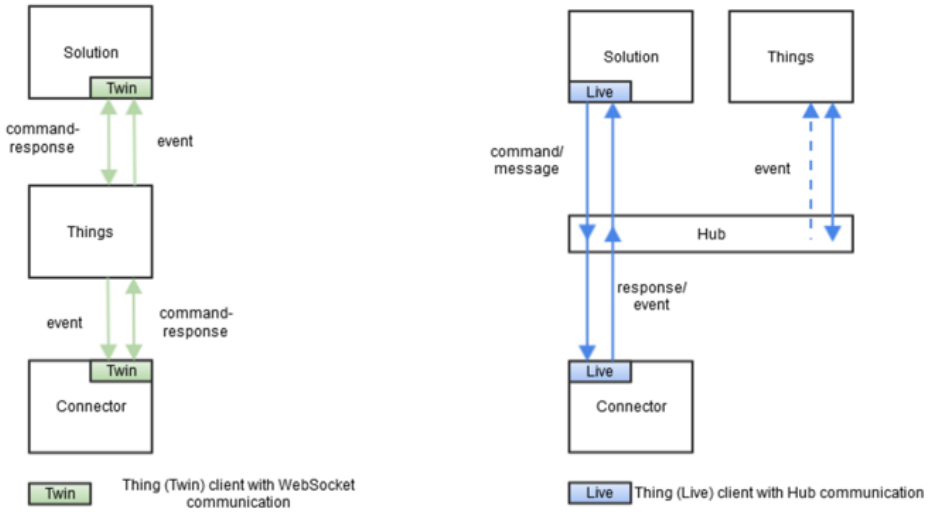
247. Similarly, for claim 16 of the ’063 Patent and claim 17 of the ’257 Patent, an embodied system receives raw data from an active object via the hardware platform, converts the raw data into usable data, and passes the usable data to a second software service (which is written in a higher level language) (e.g., an IoT application). Further for claim 16 of the ’063 Patent and claim 17 of the ’257 Patent, the usable data can be used by a second application (which is configured to receive the usable data from the second software and service).

248. The accused instrumentalities function as recited in these claims. In the accused BIOTS instrumentalities, raw data from active objects (i.e., sensors and actuators) is passed via the network, whether wired or wireless, from a device. Such raw data may include, for example, values corresponding to a sensor reading like temperature or humidity. The hardware platform receives the raw data according to the protocol being used to commuincate with the IoT device.

249. A BIOTS connector for a device, for example, communicates its measurements to interested applications on the BIOTS hardware platform. The flow diagrams below illustrate the flow of raw data from devices through middleware connectors on the hardware platform to an application.



250. For example, the Bosch IoT Hub receives commands and responses from IoT applications and devices via software services representing devices. Information flow is represented in the digital twin implementation below:



251. Raw data is communicated in message payloads.

252. Further Raw Data Receipt and Conversion. For claim 19 and claim 20 of the '063 Patent, an embodied system receives raw data from an additional active object via the hardware platform, converts the second raw data into second usable data, passes the second usable data to the additional software service. In turn, the second usable data can be used by the second application written in a higher level language. The second application is configured to receive a second usable data from the additional software service.

253. The accused intrumentalities provide for service-to-service communication and inter-application messaging to enable the passage of usable data to an additional software service.

6. Interface, Protocols, Registries, etc.

254. Plug and Play. Claim 3 of the ’063 Patent and claim 4 of the ’257 Patent cover systems in which the middleware module is configured to generate each software service after the active object is connected to communicate with the hardware platform.

255. The accused instrumentalities provide automatic connection of active objects or manual configuration. In either scenario, the BIOTS middleware generates a java object to provide an abstracted handle for the active object, creating a software service available to applications to

interact with, exchange commands, data, and messages with, and further configure, update, and manage the active object.

256. Uniform Interface. Claim 4 of the '063 Patent and claim 5 of the '257 Patent cover systems in which the software service complies with the standard, uniform interface.

257. Exemplary accused systems include BIOTS Eclipse technology (e.g., Hono) that provides and defines standard, uniform interfaces for software services. Eclipse Hono provides uniform (remote) service interfaces for connecting large numbers of IoT devices to a (cloud) backend. It supports scalable and secure data ingestion as well as command and control message exchange patterns. Further, it provides interfaces for provisioning and managing device identity and access control rules. (Exhibit 8)

258. Interface Module. Claim 9 the '063 Patent and claim 10 of the '257 Patent cover systems in which the hardware platform includes an interface module configured to connect active objects and the hardware platform.

259. Claim 10 of the ’063 Patent and claim 11 of the ’257 Patent require the interface module to be configured to connect the hardware platform with active objects via a wireless connection. Accused Systems include WiFi, Bluetooth, ZigBee, RFID, and GMS/GPRS wireless technologies, as examples. *See* http://www.bosch-connectivity.com/en/what_we_offer/expertise_1/connectivity/connectivity.

260. Claim 11 of the ’063 Patent and claim 12 of the ’257 Patent further require that the wireless connection is passively powered. Defendants have deployed passively powered radio frequency identification sensors in industrial applications meeting the additional limitations of these claims.

261. Defendants provide services to deploy RFID-based systems and “identify potential

for RFID-aided digitalization and other Industry 4.0 applications." (Exhibit 63)

262. Service Registry. Claim 37 of the '257 Patent cover systems which provide via a service registry information about the active object in the additional active object.

263. An example Accused System includes Wiki for the Bosch IoT Things service, which is also known as Central Registry. Bosch IoT Things allows all connected devices (e.g., tools, cars, sensors, and other web-enabled things) to be integrated into cloud services or other applications and to interact with each other. Accordingly, applications, other cloud services, and devices can manage their asset data and share it across IoT solutions.What is more, IoT solutions can store the data for further analysis, remotely request the change of properties for physical devices, and describe relationships of your domain's IoT assets in a simple, convenient, robust, and secure manner. *See* https://things.apps.bosch-iot-cloud.com/dokuwiki/doku.php?id=start.

264. Service Framework. Claim 38 of the '257 Patent covers systems in which the software service and the additional software service are registered and hosted in a service framework.

265. In example Accused Systems, Defendants offer the Bosch IoT Suite as a Platform as a Service (PaaS) that provides a comprehensive toolbox in the cloud. Using the software services, developers build, implement, and operate cloud-based, scalable IoT applications. Services, frameworks, and containers feature technology by ProSyst, the company Bosch acquired in 2015, and is under expansion. These systems implement IoT scenarios in managing devices, machines, and gateways; providing secure access management; executing software roll-out processes; connecting third-party systems and services; and analyzing data. See https://www.bosch-si.com/corporate/insights/features/bosch-iot-suite.html.

266. OSGi Specification. Claim 39 of the '257 Patent requires that the service

framework is based on either the OSGi standard specifications or is a .NET environment.

267. Defendants' service framework, implemented in Remote Manager for example, is based on the OSGi standard.

268. The following excerpt from Defendants' Remote Manager documentation describes BIOTS components enabling OSGi Device Management:

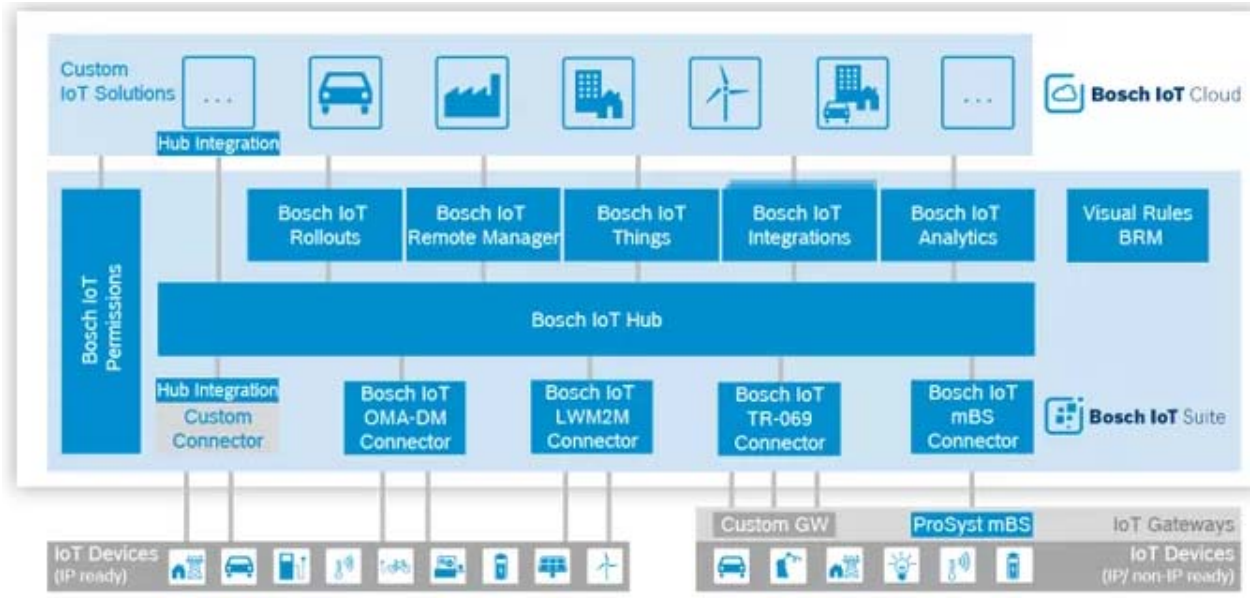
> The management of remotely managed devices is possible through a set of components, called Remote Manager agents, activated as bundles on the target devices. These components are delivered and installed on the device during the Initial Provisioning of OSGi Devices phase. Some of these components are required. Such are the Remote Manager Core Management Agent and communication provider bundles. Others are optional, such as the optional Remote Manager agents. In addition, the Remote Manager management model requires an OSGi-specified component to be available on the device: the OSGi Initial Provisioning Service.

(Exhibit 60)

269. BIOTS implements the OSGi standard specifications for software service hosting and registration on ProSyst Gateway Software:

270. Networking Protocols. Claim 15 of the '063 Patent and claim 16 of the '257 Patent require that the communication module supports Internet Protocol (IP), ad hoc networking protocol, mesh networking protocol, or some combination thereof.

271. In BIOTS, network-side communication modules are IP-based.



272. Defendants rely on IP-based networking infrastructure to ensure compatibility and provide support including for example software rollouts.

COUNT I
INFRINGEMENT OF U.S. PATENT NO. 7,895,257

273. Rokiot USA incorporates paragraphs 1 through 272 herein by reference.

274. Defendants jointly and individually practice the ’257 Patent by providing, testing, distributing, developing, making, selling, offering for sale, licensing, and/or importing the accused instrumentalities without consent or authorization.

275. The facts alleged above including the referenced publicly available materials published by Defendants and listed at Exhibit C show that Defendants practice each and every element or step of claims 1, 2, 3, 4, 5, 6, 7, 13, 14, 16, 17, 18, 19, 20, 22, 24, 25, 26, 27, 28, 32, 33, 34, 35, 37, 38, 39, and 40 of the ’257 Patent.

276. Defendants directly infringe each and every asserted claim literally, and to the extent an element or step is found not to be literally met by or in the accused instrumentalities, it is met under the doctrine of equivalents.

277. Defendants individually and jointly infringe the ’257 Patent. Defendants are related companies under the control of the parent, Robert Bosch GmbH. To the extent all steps or limitations of any asserted claim are not practiced by a single defendant entity, then all steps or limitations, as the case may be, are practiced by or attributable to Defendant Robert Bosch GmbH.

278. The Bosch Defendants are on notice of the ’257 Patent and how the accused instrumentalities infringe the asserted claims. Defendants’ continued acts of infringement including inducing, encouraging, aiding, abetting, directing, and instructing others, namely their customers, developers, and end users under 35 U.S.C. 271(b) and 271(c) of the accused instrumentalities, to practice the ’257 Patent constitutes indirect infringement, including by providing user guides, instruction materials and customer support.

279. Defendants provide, make, sell, use, license, offer to sell, and promote the Bosch IoT platform and the specifically accused products having features and functionality described herein with the specific intent that end users and customers use the accused instrumentalities in an infringing manner on and in conjunction with the Bosch IoT platform.

280. As alleged herein, Defendants provide to such third parties the Bosch IoT modular platform that enables automatic integration of heterogenous devices, sensors, and actuators in heterogeneous networks as described and claimed in the asserted claims.

281. As alleged herein, the Bosch IoT platform components are material to practicing the ’257 Patent, have no substantial non-infringing use, and are known to Defendants by notice provided in this complaint to be especially made or adapted for use in infringing the ’257 Patent.

282. Defendants have notice and knowledge of the ’257 Patent by this complaint.

283. Rokiot USA has been harmed as a result of Defendants’ infringing conduct. Defendants are liable to Rokiot USA in an amount that adequately compensates it for their

infringement, which compensation cannot be less than a reasonable royalty together with interest and costs as fixed by this Court under 35 U.S.C. § 284.

COUNT II
INFRINGEMENT OF U.S. PATENT NO. 8,631,063

284. Rokiot USA incorporates paragraphs 1 through 283 herein by reference.

285. Defendants jointly and individually practice the ’063 Patent by providing, testing, distributing, developing, making, selling, offering for sale, licensing, and/or importing the Accused Instrumentalities without consent or authorization.

286. The facts alleged above including the referenced publicly available materials published by Defendants and listed at Exhibit C, show that Defendants practice each and every element or step of claims 1, 2, 3, 4, 9, 12, 13, 16, 17, 18, 19, 20, 21, 24, 25, 30, 31, 35, 37, 38 of the ’063 Patent.

287. Defendants directly infringe each and every asserted claim literally, and to the extent an element or step is found not to be literally met by or in the accused instrumentalities, it is met under the doctrine of equivalents.

288. Defendants individually and jointly infringe the ’063 Patent. Defendants are related companies under the control of the parent, Robert Bosch GmbH. To the extent all steps or limitations of any asserted claim are not practiced by a single defendant entity, then all steps or limitations, as the case may be, are practiced by or attributable to Defendant Robert Bosch GmbH.

289. The Bosch Defendants are on notice of the ’063 Patent and how the accused instrumentalities infringe the asserted claims. Defendants’ continued acts of infringement including inducing, encouraging, aiding, abetting, directing, and instructing others, namely their customers, developers, and end users under 35 U.S.C. 271(b) and 271(c) of the accused instrumentalities, to practice the ’063 Patent constitutes indirect infringement, including by

providing user guides, instruction materials and customer support.

290. Defendants provide, make, sell, use, license, offer to sell, and promote the Bosch IoT platform and the specifically accused products having features and functionality described herein with the specific intent that end users and customers use the accused instrumentalities in an infringing manner on and in conjunction with the Bosch IoT platform.

291. As alleged herein, Defendants provide to such third parties the Bosch IoT modular platform that enables automatic integration of heterogenous devices, sensors, and actuators in heterogeneous networks as described and claimed in the asserted claims.

292. As alleged herein, the Bosch IoT platform components are material to practicing the ’063 Patent, have no substantial non-infringing use, and are known to Defendants by notice provided in this complaint to be especially made or adapted for use in infringing the ’063 Patent.

293. Defendants have notice and knowledge of the ’063 Patent by this complaint.

294. Rokiot USA has been harmed as a result of Defendants’ infringing conduct. Defendants are liable to Rokiot USA in an amount that adequately compensates it for their infringement, which compensation cannot be less than a reasonable royalty together with interest and costs as fixed by this Court under 35 U.S.C. § 284.

NOTICE OF REQUIREMENT OF LITIGATION HOLD

295. Defendants are hereby notified it is legally obligated to locate, preserve, and maintain all records, notes, drawings, documents, data, communications, materials, electronic recordings, audio/video/photographic recordings, and digital files, including edited and unedited or “raw” source material, and other information and tangible things that Defendants know, or reasonably should know, may be relevant to actual or potential claims, counterclaims, defenses, and/or damages by any party or potential party in this lawsuit, whether created or residing in hard copy form or in the form of electronically stored information (hereafter collectively referred to as

"Potential Evidence").

296. As used above, the phrase "electronically stored information" includes without limitation: computer files (and file fragments), e-mail (both sent and received, whether internally or externally), information concerning e-mail (including but not limited to logs of e-mail history and usage, header information, and deleted but recoverable e-mails), text files (including drafts, revisions, and active or deleted word processing documents), instant messages, audio recordings and files, video footage and files, audio files, photographic footage and files, spreadsheets, databases, calendars, telephone logs, contact manager information, internet usage files, and all other information created, received, or maintained on any and all electronic and/or digital forms, sources and media, including, without limitation, any and all hard disks, removable media, peripheral computer or electronic storage devices, laptop computers, mobile phones, personal data assistant devices, Blackberry devices, iPhones, video cameras and still cameras, and any and all other locations where electronic data is stored. These sources may also include any personal electronic, digital, and storage devices of any and all of Defendants' agents, resellers, or employees if Defendants' electronically stored information resides there.

297. Defendants are hereby further notified and forewarned that any alteration, destruction, negligent loss, or unavailability, by act or omission, of any Potential Evidence may result in damages or a legal presumption by the Court and/or jury that the Potential Evidence is not favorable to Defendants' claims and/or defenses. To avoid such a result, Defendants' preservation duties include, but are not limited to, the requirement that Defendants immediately notify its agents and employees to halt and/or supervise the auto-delete functions of Defendants' electronic systems and refrain from deleting Potential Evidence, either manually or through a policy of periodic deletion.

NOTICE

298. Rokiot USA has complied with the notice requirements of 35 U.S.C. § 287.

JURY DEMAND

Rokiot USA hereby demands a trial by jury on all claims, issues, and damages so triable.

PRAYER FOR RELIEF

Rokiot prays for the following relief:

a. That Defendants be summoned to appear and answer;

b. That the Court enter an order declaring that Defendants have infringed the ’257 and ’063 Patents;

c. That this is an exceptional case under 35 U.S.C. § 285;

d. That the Court grant and award Rokiot USA judgment against Defendants for all actual, consequential, special, punitive, exemplary, increased, and/or statutory damages, including any applicable additional damages pursuant to 35 U.S.C. § 284, and, if necessary, an accounting of all damages; pre and post-judgment interest as allowed by law; and reasonable attorneys’ fees, costs, and expenses incurred in this action; and

e. That Rokiot USA be granted such other and further relief as the Court may deem just and proper under the circumstances.

Dated: July 10, 2017

Respectfully submitted,

s/Richard B. Megley, Jr.

Richard B. Megley, Jr. (rmegley@leesheikh.com)
Brian E. Haan (bhaan@leesheikh.com)
LEE SHEIKH MEGLEY & HAAN
111 West Jackson Blvd., Suite 2230
Chicago, IL 60604
Ph: (312) 982-0070
Fax: (312) 982-0071

Cabrach J. Connor (TX State Bar # 24036390)
(cconnor@taylordunham.com)
Admission *pro hac vice* pending
Jennifer Tatum Lee (TX State Bar # 24046950)
(jtatum@taylordunham.com)
Admission *pro hac vice* pending
TAYLOR DUNHAM AND RODRIGUEZ LLP
301 Congress Ave., Suite 1050
Austin, Texas 78701
Ph: (512) 473-2257
Fax: (512) 478-4409