Collaborative Open Market to Place Objects at your Service

D9.1.1 Initial Standardization Report

<table>
<thead>
<tr>
<th>Project Acronym</th>
<th>Compose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Title</td>
<td>Collaborative Open Market to Place Objects at your Service</td>
</tr>
<tr>
<td>Project Number</td>
<td>317862</td>
</tr>
<tr>
<td>Work Package</td>
<td>9</td>
</tr>
<tr>
<td>Lead Beneficiary</td>
<td>9</td>
</tr>
<tr>
<td>Editor</td>
<td>Dave Raggett</td>
</tr>
<tr>
<td>Reviewer</td>
<td>Iacopo Carreras</td>
</tr>
<tr>
<td>Reviewer</td>
<td>Lukasz Radziwonowicz</td>
</tr>
<tr>
<td></td>
<td>Fraunhofer FOKUS</td>
</tr>
</tbody>
</table>

© 2013 D9.1.1 Initial Standardization Report
Abstract

The aim of this deliverable is to identify existing standards relevant to the Compose project's goals, and to pave the way for further standardization. The starting point is to review the requirements and technologies needed for enabling open markets of services for the Internet of Things, to provide pointers to additional information, and to identify relevant standards development organizations and relevant existing standards. This will provide the foundations for further work on identifying gaps where new standards would be valuable.

In addition, this deliverable covers work in progress, e.g. at the W3C to build a community around use cases, requirements and technologies for the Web of Things, and plans for a W3C workshop to advise on standardization priorities.

Table of Contents

1 Introduction ................................................................. 3
2 Application Areas .......................................................... 5
3 General considerations for the Web of Things ........................ 11
4 Requirements for open markets of services .......................... 19
5 Communications technologies for the Web of Things ............. 28
6 Techniques for specifying services. .................................... 34
7 Ethical considerations for the Web of Things ....................... 44
8 W3C Web of Things Community Group ............................. 47
9 W3C Web of Things Standardization Workshop .................. 48
10 Conclusions ............................................................... 49
11 Related Work ............................................................. 50
1 Introduction

The Web of Things includes sensors and actuators, physical objects and locations, and even people. The Web of Things is essentially about the role of Web technologies to facilitate the development of applications and services for things and their virtual representation. Some relevant Web technologies include HTTP for accessing RESTful services, and for naming objects as a basis for linked data and rich descriptions, and JavaScript APIs for virtual objects acting as proxies for real-world objects.

The Compose project aims to enable open marketplaces of services for the Web of Things. The plan is to apply Web Technologies and to build a working implementation as a testbed for the ideas, and to use that to bootstrap a community of users and developers of innovative services. This can build upon a wide variety of existing standards, and this report provides a survey of uses cases, requirements, architectural concepts and technologies as a basis for identifying relevant standards and standards development organizations.

Standards are needed to realize the economic and human potential, and to avoid the risk of fragmentation cased by a plethora of non-interoperable proprietary solutions. Innovation is spurred by standards that can be implemented without the need to pay royalties, and with licenses that offer non-restrictive terms and conditions.

Open marketplaces allow participants to trade freely with minimal constraints. For Compose, this involves the mechanisms to support different roles that participants can take, e.g. end users seeking to discover interesting apps and services, and developers who seek to provide these. Other roles include the provision of the marketplace itself, and the infrastructure and devices upon which the services rely. The business models may be based upon engagement with advertisers, or through various forms of payment (one off, per use, or subscription). Trust will depend on the availability of reviews, and a focus on security, e.g. automatic updates for security patches.

1.1 Interacting with the Physical World

The Web of Things is about interacting with the physical world around us through sensors, actuators and services. The following introduces some examples.

- Examples of Sensors
  - Thermometer -- maps the temperature of its immediate surroundings into a digital reading
  - Motion detector -- detects things moving in its field of view
  - Security sensor -- e.g. that a door or window is open
  - Microphone -- maps audio into an electrical equivalent
  - Camera -- takes photos or video of its field of view

- Examples of Actuators
  - Switch -- e.g. to open a lock, turn on a light, heating system, or valve
  - Thermostat -- e.g. to set the upper and lower values that control an air conditioner
• Movement -- e.g. to pan and zoom a camera
• Loudspeaker -- transform an electrical signal into audio
• Display -- display a static image or a video stream

• Examples of Generic Services
  • Storage -- buffering data between when it is produced and when it is consumed
  • Transformation -- mapping data from one form into another
  • Coordination -- coordinating the behaviour of a set of actuators

1.2 Ensuring that it works

If we have a rich ecosystem of services, we will need ways to know which services can be used with what sensors and actuators, and likewise which services can be chained together to form composite services, where the output of one service is connected to the input of another.

The Linux operating system provides an analogy with its support for software packages. These bundle software together and describe the dependencies on other packages. When installing a given version of a package, this makes it possible to install all of the packages it depends on. It also allows checks for clashes, where installing a new package would prevent an previously installed one from working. To enable an open marketplace of services for the Web of Things, we need a similar approach with machine interpretable descriptions of dependencies.
2 Application Areas

This page is a collection of application areas for the Web of Things. Agreed use cases are essential for driving work on standards. The following is really just a starting point for elaborating use cases into stories that provide sufficient details to derive functional requirements. Here is an example:

2.1 what is raiding my vegetable patch

A story about users and developers of a cloud-based Web of Things platform. This was written this as basis for discussion about functional requirements for the platform.

Sue goes to the mall and purchases the ACME home security camera as she wants to be able to monitor the wild animals that are eating the vegetables she has planted in her garden.

The camera has an infra red light, a motion sensor, and can be remotely panned and zoomed.

Sue goes online and logs into her account with the Web of Things platform. She searches for her model of camera and finds that secure-me.com are selling the service she needs. She clicks the button to confirm the purchase for the service (2 USD per month). She starts the secure-me service and is asked for some details in her browser. She opts to keep the data private so that only she can access the data, and to save 5 minutes of video when the camera's motion detector senses something moving in the field of view. She then taps a button to associate the camera with her service. This makes use of local discovery to find the camera, and prompts her to confirm that this is her camera. She does so.

Local discovery is through a discovery API exposed by the browser to trusted web page scripts. Associating a device with a service could involve establishing some form of secure mutual authentication between the device and service.

Sue's new home security service is hosted by a Web of Things cloud-based platform with which she has a user account. The service is an instance of the service class developed by secure-me.com. Secure-me.com created their service class with their account on the Web of Things platform developer portal. A service class has some meta-data that describe the service in a machine interpretable way, some scripts that define the behaviour, and some additional resources such as the icon to be used to depict the service.

The various scripts deal with: 1) associating the device with the service, 2) setting the user preferences, 3) communicating with the device and saving data in the Web of Things platform storage subsystem, and 4) implementing the run-time API that the service exposes to consumers on the Web of Things platform (i.e. apps and composite...
services). The roles of the scripts and the kinds of meta-data are specified by the Web of Things platform.

A few days later Sue logs back into Web of Things platform and looks at her services. She sees a brief summary for each service. Her security camera service shows that the camera has been activated 5 times. She taps on the service to find out more. It opens up into a display of the activity log. Each entry has the time the camera was activated and a 4 small representative photos from the video stream. She taps on one of entries to view the video and sees a badger busily digging up and eating her carrots.

The service includes an app to view the data it collects. It can allow users to be sent real-time alerts to their mobile phone.

Sue decides she wants the alerts. The next night, her phone chimes and she taps the notification to get live access to her security camera. She is curious to see where the badger goes after it has finished eating. She pans and zooms the camera to follow the badger as it disappears into a hole under her hedge.

Sue has registered her phone with the Web of Things platform so the service can make use of a platform API to send a notification to her phone. The notification is linked to an app, so that when Sue taps the notification, the app is launched in the phone's browser.

The app has access to the service API and this allows it to pan and zoom the camera.

This user story shows that service developers can include apps that can access the data logged by the service, and the live API exposed by the service. This relies on a means to bind the apps to the specific service, so that only Sue can view data from her camera, and likewise prevents her from viewing other cameras using the same service class.

This approach treats services as instances of classes defined by developers. Services are associated with user accounts. Each service is subject to access control policies. Sue chose to make her service private to her. John by contrast has positioned his camera to view his bird table and he has chosen to make the camera publicly accessible.

A given service could be bound (associated) to devices, it could consume data from other services, and it can expose data to yet other services. A service can also be associated with apps to provide a rich user interface as in the above use case.

The Web of Things platform needs to provide a way to execute services, either lazily upon demand when data is sought from a service, or eagerly when data becomes available, that the service consumes.
2.2 Smart Cities

This is a disparate collection of use cases.

2.2.1 Public Transportation

- Webcams for public places
- Current data on traffic
- Current location of buses, trams and trains
- Forecasts, e.g. for congestion maps
- Predictions for journey times
- Navigation aids for walking, cycling, driving, etc.
- Guiding people to available parking spaces
- Guiding people back where their car is parked
- Services for smart use of shared bicycles
- Long term data for infrastructure planning
- Collecting statistics on network performance

2.2.2 Smart Utilities

- Smart meters as a means to encourage smarter consumer habits
- Smart plugs as a means to spread load by using off peak tariffs
- Smart watering of lawns and gardens
- Consumer generated power that feeds back into the Grid
- Energy storage devices for the city, block or individual homes

2.2.3 Context based search

- Helping people to find what's on where
- Information on shops, restaurants, theatres, & pubs, area by area
- Information on schools, clinics, parks, and other local services
- Location and time of day based search

2.2.4 Emergency overrides

The day to day policies for access to given sensors and actuators in a building may need to be overridden in emergency situations, e.g. in case of fire or flooding. Police or Fire brigade staff may need to be able to unlock the doors and turn off equipment etc.

2.2.5 Smart Cities and Big Data

Smart city use cases will often be associated with really large datasets that become hard to process using traditional database technologies. New techniques for Big Data exploit large clusters of computers. What are some of the use cases and algorithms?

to be expanded
2.3 Smart Resorts

Helping tourists to get the most out of their vacations. An example is a ski resort where people could get information on what's on, opening and closing times of lifts and the current queues, how busy each piste is, their current location on the piste map, the weather conditions and so forth.

2.4 Smart Homes

This covers a range of ideas including home automation. **Personal Zone** is a name for the abstract space containing all of your personal devices, apps and services. Note the distinction between personal devices, e.g. your smart phone, and devices that are shared, e.g. a living room TV. One approach is for every device to have an owner who delegates rights to others to use it, e.g. as a responsible parent, you could put controls on your TV to limit the kinds of content your children can see.

2.4.1 Home Security

A smart home can provided improved security through a network of motion sensors, sensors on doors and windows, video cameras and so forth. Another benefit could be reduced insurance bills, and the ability to check on your home when you are away.

2.4.2 Smart Heating and Cooling

Saving energy through better data on room temperature and which rooms are occupied now, or are soon expected to be based upon data mining of past behaviour. In future, we can expect a richer range of mechanisms for managing the environment in our homes: automatic shutters to deflect sunlight, and heat pumps that can pump heat to/from underground storage areas (e.g. underground pipes).

2.4.3 Smart Lighting

Avoiding wasting electricity on lighting when you are out of the room. This likely to become less valuable as we switch to power efficient LED lighting.

2.4.4 Extended Warranties

You pay for an extended warranty service in which you enable a provider to monitor devices such as your washing machine, central heating and air conditioning system and so forth. This allows them to detect problems in advance of breakdowns and to proactively schedule maintenance visits at a time convenient to you. This is based on the diminishing costs for adding network connectivity to consumer products. A side effect is being able to check on you devices remotely, and to get alerts e.g. when the washing cycle has finished.
2.4.5 Media Sharing

This is about a combination of local storage and network streaming for a variety of media: electronic photo frames, audio and video. We will want easy access from whatever room we are in.

2.5 Smart Enterprises

Many of the ideas for Smart Homes could be applied to the workplace. Smart enterprises could provide staff with access to rich information about the organization's structure, where people are located physically, meeting rooms, the available devices, e.g. printers, scanners, projectors, and so forth. Access control policies can be set based upon your role within the organization, e.g. as the departmental manager for human resources.

2.6 Home Healthcare

The cost of healthcare continues to rise and rise. Home healthcare promises to reduce the costs and improve outcomes. This based upon devices for monitoring your health and your use of medications. We can expect an increasing range of sensors including blood sugar levels, heart rate and patterns, temperature and so forth. There are plenty of challenges, e.g. designing devices for use by people who may not see or hear well, or suffer from mental confusion. Security and privacy will be key to broad acceptance.

2.7 Retail and Richer Shopping Experiences

Shops will seek ways to offer enriched shopping experiences, e.g. helping you to find products and to get information about them and other products that you may want, e.g. matching items for a cool fashionable look, or food items for appealing recipes. You may want to know something about where an item comes from and whether the suppliers received a fair proportion of the final price. People with family members with allergies would like instant feedback on whether a given product is safe without having to read the small print for the ingredients. If you are on a diet, you could get suggestions for what to buy for enticing healthier meals. You may want to get independent product reviews before deciding to purchase a given item. You may be interested in what items your friends have bought or looked at recently.

This is all possible with barcodes and NFC, assuming your device can access the Web, and we can expect more and more stores to offer WiFi to enable this. we can expect to see third party services, whether these are in cooperation with the store or completely independent.

A relevant technology is indoor location sensing, and the means to detect when you are moving into and out of particular zones. This could be used to show promotions relevant to the section of a store or mall you have just entered. The demand on battery life could be reduced by pushing the details down to the hardware level for efficient execution on microcontrollers.
Another idea is based upon wireless charging of mobile devices, something recently introduced by Starbucks. Imagine walking into a cafe and on noticing the charger icon on the table, you place your mini-tablet down to charge. As it does so, you are automatically connected to the cafe's WiFi network and presented with enticing information and offers, possibly personalized to you based upon your purchasing history and interests. This scenario combines wireless charging, NFC and WiFi, where NFC is used to connect your device to the WiFi and open the browser at the designated website.
### 3 General considerations for the Web of Things

Whilst it is unlikely that there will be one architecture that suits all purposes, it is worth considering some of the ideas that have emerged so far. The following introduces some of the challenges, and readers are advised to look at related work for more details.

#### 3.0.1 Services

Services provide general building blocks for the Web of Things.

- Services as a source of data, e.g. when they are bound to sensors
- Services as a sink of data, e.g. when they are bound to actuators
- Services may process data obtained from other services

In general, a service could be any combination of the above roles. An example is a service for a camera where you can command the camera to pan and zoom, to enable/disable the flash, and to take a photo. In this case, the service takes commands as input and generates photo's as output. Another service could take photos as its input and generate a list of tags for the people present in each photo.

Services can provide zero or more data sources, and zero or more data sinks, but at least one source or sink is required. For services bound to actuators, the sink could be for commands or data, e.g. a service for a digital picture frame that allows you to push pictures to the frame.

For scalability, the services are likely to run on Internet platforms, whilst the sensors and actuators are at the edge of the network. Web protocols such as HTTP can be used to connect services to the sensors and actuators, but complications arise as many devices will be behind security firewalls, and may require a bridge between the Web protocols and the communications technologies supported by devices.

Services can process data to alter the level of abstraction, e.g. taking an audio stream of spoken speech as input and generating a sequence of words as output, i.e. more abstract. In the reverse, the output (an audio stream) is more concrete than the input (a sequence of words). A further possibility is to transform data at the same level of abstraction, e.g. changing the units of measure.

In an open market of services, some companies could develop and market services for others to purchase and install. An example is a service for using web cams for home surveillance, where the user purchases a camera from one supplier and the service software from another. It is necessary to distinguish the definition of a service from its instantiation for a particular purpose.
Applications and Services

An application is something that end users can directly interact with. For Web applications, this is via a Web browser. Applications can make use of services, for example, you could have a web page that shows the predicted time for someone to commute to work based upon services that sense traffic speeds.

Service Definitions

Services are essentially a mix of metadata describing the service, code that defines the service's behaviour, and data that the service consumes or produces. The metadata includes such things as who owns the service, the type definitions of the APIs exposed by the service, access control policies, and so forth. The code could be written in a variety of languages according to what the platform supports, e.g. Java, JavaScript or Python.

Protocols or Objects

A common approach is to host services on servers and to provide access to the service via a protocol like HTTP. This allows for a scalable distributed architecture. However, developers will in many cases prefer to avoid having to explicitly work with HTTP. A natural alternative is to expose the service as interfaces on an object that acts as a proxy for the service. You can then access a service with a simple method call, rather than having to set up an HTTP connection, prepare the request headers, open a TCP/IP connection to send the request and so forth.

The code that implements these proxy objects could be automatically generated from the service description. The proxy objects can be used in web applications, or in services composed from other services. For a Web application the proxy objects can be provided as part of a JavaScript library. For services that run in the cloud, the platform for the Web of Things could automatically expose the proxy objects to the scripts implementing the service's behaviour, based upon the type declarations given in the service description. In other words, when defining the service, you declare what other services you want to use, and the platform provides a means to bind to these services when the service starts running. Your code can then access the properties and methods on the proxy objects.

This illustrated by the following diagram:
The scripts here are either part of a web application or part of a cloud based service. The hidden messaging layer decouples scripts from the details of how devices or other services are accessed. A convenient implementation technology for services using JavaScript is node.js.

### 3.0.2 A multiplicity of evolving communication technologies

There are many different technologies for connecting devices, and the list keeps changing as new technologies are introduced. This makes it desirable to have an architecture that hides details that are best dealt with at a lower level.

- Communications technologies for the Web of Things

### 3.0.3 Battery operated devices

For sensors and actuators that are battery operated, the battery life is an important consideration. Protocols like ZigBee prolong battery life by only powering up the communications system at agreed intervals. This approach can be used for sensor networks, where each devices cooperate to pass data between them. Gateway devices serve as a bridge to the Internet.

### 3.0.4 Versioning of APIs

The lifetime of devices will vary considerably according to the kind of device and the context in which they are used. This makes it inevitable that services will need to cope with different versions of APIs exposed by the devices. How can developers create applications and services that work with yesterday's devices, today's devices and tomorrow's devices? This isn't just about device APIs as change in services is also
inevitable in a large and dynamic ecosystem. In general, this is a hard problem to address.

On the Web, developers have found ways to test for the existence of a given API by testing for the presence of a named property, or by attempting to invoke a method and catching the exception that is raised if the API is unsupported. This approach can fail if the APIs evolve in ways that old code failed to anticipate.

Another approach is to associate each version of an API with an unique identifier, and to provide a means to query what versions are available, and a means to bind to a specific version of the API. This is the approach taken in Microsoft's COM -- the object model upon which ActiveX and OLE (Object Linking and Embedding) is based. For an introduction, see:

- Introduction to COM

Versioning is related to the challenge of adapting to variations in device capabilities. In principle, a device (or service) could be described with machine interpretable metadata. In practice there is a risk that the metadata doesn't provide an accurate account. Developers may apply caution by directly probing to see if a given capability is present rather than relying on the metadata.

### 3.0.5 Scaling issues

Imagine a device exposes a service via HTTP, allowing other devices to connect to it over a network connection. If the service becomes very popular the embedded server will have difficulty in handling the volume of connection requests. This problem can be avoided through the use of proxy servers which can be scaled to match the load.

More generally, services can be cloud-based. That is to say, the service definitions are registered with a cloud-based platform that provides for scalable run-time execution of services. You can have a federated architecture with services running on servers operated by different vendors. This relies on open standards as a basis for interoperability.

### 3.0.6 Security firewalls

Sensors and actuators will often be placed behind security firewalls to prevent unauthorized access. A common example is provided by home gateways that block incoming connections from outside of the local network, and which serve as network address translators (NAT). If you want to establish peer to peer connections for devices hidden behind their own firewalls, a third party may be essential. APIs for configuring firewalls present a high security risk.

Related standards include support for establishing tunnels through NATs:

- STUN (Session Traversal Utilities for NAT) Wikipedia article
- RFC 5389 which defines the STUN protocol
3.0.7 Policy frameworks

Policies are needed to control access to devices, and to safeguard personal privacy. Policies determine who can access a service for what purposes in which contexts. Sticky policies remain associated with the data to which they apply, so that the policies can still be enforced when the data is moved from one location to another. Policies can be expressed as rules which can be applied dynamically when needed. Another approach is to provide capability tokens that a would be user of a service has to present to gain access. There may be a need to provide audit trails as a means to support accountability. In some use cases, there is a need to be able to override policies when the appropriate situation arises, e.g. in an emergency and under the direction of police or fire officials.

XACML

The extensible access control markup language (XACML) is a declarative XML format for access control rules that has been standardized by OASIS. XACML enables a separation of the access decision from the point of use.

- Wikipedia article on XACML
- OASIS XACML TC

The EU PrimeLife project focused on privacy, and explored extensions to XACML to cover data handling requirements. Privacy policies specify the purposes that personal data can be used for, and for how long personal data can be retained.

OAuth

A widespread example of capability tokens on the Web is the OAuth framework for secure authorization. It allows a user of a website to grant that website with temporary and limited access to personal resources on another server, and avoids the need to share user passwords across servers. The "session fixation" security flaw in OAuth 1.0 (RFC 5849) led to the development of OAuth 2.0 (RFC 6749) which relies on SSL for confidentiality and server authentication. OAuth 2.0 itself is considered to have weak security, in part due to implementation issues around URL redirection. More details are given in a post by Egor Homakov.

- Wikipedia article on OAuth
- OAuth.net
- RFC 5849 -- The OAuth 1.0 Protocol
- RFC6749 -- The OAuth 2.0 Authorization Framework
- Egor Homakov: OAuth1, OAuth2, OAuth...?

3.0.8 Conflicts of interest

Data should not be made available where there is a known conflict of interest, e.g. where a breach would give an unfair commercial advantage. To prevent this from
happening, the data needs to be tagged in such a way that the policy engine can detect that a given usage request would result in a conflict of interest. The engine needs to be able to track provenance across service compositions, and to reason over organizational structure and roles. This can be based upon machine interpretable descriptions.

Static code analysis techniques may be feasible in some cases to verifying adherence to conflict of interest policies. Note that this breaks down for applications and services that are not under the control of the policy engine. Daniel Hedin and Andrei Sabelfeld have carried out an analysis of Information-Flow Security for a Core of JavaScript (April 2012).

### 3.0.9 Faults and Deceits

With any really large system the risk of components failing increases to the point where there is a need to function in the presence of faults. How are faults detected? What are the workarounds? How are the implications of faults communicated to end-users? Enemies could attack a system, introducing false data, and causing subsystems to fail. Are there techniques that can be used to monitor operation and trigger alarms when evidence of attacks are discovered? Is it possible to shutdown corrupted subsystems and provide workarounds using components that are still deemed to be healthy? One potentially useful metaphor is the human body's immune system.

### 3.0.10 Personal Zones

Users are likely to purchase devices, applications and services from many different vendors. This makes it desirable to provide users with a means to manage their personal devices, applications and services in a way that works seamlessly across vendors. The Webinos project has explored this in terms of a Personal Zone in which each of your devices include an embedded agent (the Personal Zone Proxy) that integrates the device into the zone. Trusted web applications have access to a suite of APIs exposed by the zone. This includes the means to discover and bind to services on devices within the zone. The Personal Zone Hub exposes the zone on the Internet, enabling inter-zone applications and services, and in principle, putting you back in control over your personal data. Webinos also supports Internet of Things devices, and these are required to embed a Personal Zone Proxy, or be accessed via a device that does. Webinos demos include using a phone to control a TV, home automation and home healthcare.

Proprietary marketplaces like Google Play approach some of these needs for personal devices running the Android operating system. When you sign into the Google Play website you can see which of your registered devices are compatible with apps available from the marketplace. However, users are already concerned with the amount of personal information held by Google, and this together with Google's proprietary focus on Android is likely to limit Google extending its scope to cover devices that aren't running Android.
Browser sync is another step towards the evolution of Personal Zones. Browser vendors like Mozilla, allow users to sync personal data across all of their devices with the Firefox browser. This currently covers bookmarks, history, passwords and open tabs. Synchronization is performed via the cloud, but the data is held in an encrypted form, and the key never leaves the browser. The more general capabilities of Personal Zones include services that run on behalf of the user and independently of browsers. Effective security (and privacy) will depend on defense in depth. This points to the need for further work on distributed authorization frameworks where security breaches will be limited in the damage they can cause.

3.0.11 Perception, Actuation and Coordination

We can learn a lot by studying how animals (and humans) deal with perception and actuation. An example is how we understand spoken utterances using a pipelined process where one stage feeds into another: sound waves, phones, phonemes, words, sentences, resolution of references, and cognition. This involves making inferences at one level based upon uncertain data provided at a lower level and taking the context into account. Such processes are also associated with learning mechanisms that allow us to learn new words and their meaning. For the Web of Things, this corresponds to service compositions, where one service builds on top of others, creating progressively higher level interpretations of data. Learning processes corresponding to data mining algorithms, and may operate over very large amounts of data (Big Data).

A similar analogy applies to actuation, where a high level intent is progressively transformed into lower level commands for actuators, often involving careful coordination across many actuators. The act of speaking is a perfect example, where your thoughts are translated into commands for driving a large set of muscles. Some react quickly, e.g. the tip of your tongue, while others like those positioning your jaw take much longer and need to be set into motion further in advance. Actuators could expose APIs for invoking commands, or for uploading scripts for local execution. The actuation pipeline corresponds to a composition of services which accept higher level commands and map them into lower level ones.

Note: one approach to coordination is to determine relative clock skews for actuators, and to delay actuation to ensure that all of the actuators involved are driven in synchrony. Accuracies of ten milliseconds have been achieved in practice using Web technologies such as Web Sockets.

Coordination involves a timeline. This is about agreements on how variables change with time. This includes transitions where a variable follows some function over time that the device can compute, e.g. a linear or spline function. The rendered frame rate will depend on the speed of the device. A related concept is time warping, where time is sped up, or slowed down.

Some related standards include:

- Synchronized Multimedia Interaction Language (SMIL)
- CSS animation
• CSS transitions
4 Requirements for open markets of services

A flexible and healthy ecosystem for the Web of Things will involve many stakeholders and many roles, for example:

- Designers and manufacturers of sensors, actuators, gateways and security firewalls
- Infrastructure providers for communication technologies
- Providers of cloud hardware and software
- Companies that host markets of services
- Consumers who purchase devices and register them with the Web of Things
- Designers of applications and services, including composite services
- Companies that provide generic or specific services for search & discovery
- Companies that support payments, including currency conversion, and trust brokering
- Security specialists that provide monitoring tools to detect security breaches

Understanding the needs of the different roles will help to clarify the requirements, and in turn, where new standards are needed, and where existing standards are adequate.

4.1 Overview of Marketplaces

Consumers are familiar with the proprietary marketplaces for native apps for Apple's iPhone and iPad, and likewise for Android. This suggests the potential for marketplaces for apps and services for the Web of Things.

4.1.1 Stakeholders

The requirements follow from the needs of the stakeholders.

Marketplace Operators

Companies that operate the marketplace.

Cloud Server Providers

Companies that own the hardware that the marketplace platform is built upon.

Businesses

Businesses will want to monitor the apps and services they provide, to place advertisements and monitor their effectiveness, using a variety of analytic tools.
Device Vendors

Companies that provide the sensors, actuators, gateways and other devices that form the basis for the Web of Things.

End Users

This includes people at home, on the move or at work. Imagine that you want to set up a home automation system. You sign into the marketplace and look for reviews of devices and applications that match your needs. You then purchase the devices online or from a bricks and mortar store. You identify the applications you need and install them from the marketplace. You then need to run the application to configure the devices you're using. Some days later, you receive a notification that your application and the services it uses have been updated (to add new features or to fix bugs).

End users will want ways to manage their apps and services, to carry out searches, read reviews, and to arrange payments. Users may also wish to create their own services using high level tools created by developers and operated by businesses.

Developers

Developers will need tools to support the creation and evolution of the apps and services they sell. This suggests the need for ways to create and manage development projects, and a suite of tools for editing, testing and debugging. This includes syntax colouring for editing scripts and structured data, and diagram based editors for service compositions as a graph of pipes and nodes, see e.g. Yahoo Pipes:

- Wikipedia article on Yahoo! Pipes
- Yahoo! Pipes website

Debugging of scripts, that are running in the cloud, requires an API for setting breakpoints, single stepping, accessing named variables etc. Other requirements include the means for developers to search the marketplace for other services that they can use when defining their own services.

4.2 Functional Requirements

4.2.1 Identity and Accounts

End-users, businesses and developers will need to identify themselves to marketplaces. It is reasonable to expect that people are required to set up accounts and a means of authenticating themselves.

The Web has largely been based upon user-name and password credentials for authentication. This is known to be insecure as users find it difficult to remember strong passwords, and instead tend to use a weak, but easy to remember password across many websites. Email addresses are common for identifying people. This is
bad for privacy as it makes it easy to aggregate information about people's online habits across websites.

Single sign-on schemes avoid the need for users to remember credentials for each website. Examples include:

**OpenID**

- [OpenID](#)
- [Wikipedia article](#)

This allows you to sign in with a relying website using an existing account with a popular website such as Google or Yahoo! which plays the role of identity provider. The relying website trusts the identity provider when it comes to authenticating your identity.

**Mozilla Persona and BrowserID**

- [Mozilla Persona](#)
- [How BrowserID works](#)

This uses your email address as an identity. The browser uses your email address to generate public key pair. The identity provider verifies that you own this email address and generates a digital certificate for this email address and public key. Your email address can then be used to generate a cryptographic proof to the relying website that you are the owner of that email address. Unlike OpenID, there is no need to contact the identity provider when logging into a website. BrowserID is thus said to be more privacy friendly.

- *Does BrowserID provide a different identity for each relying website, unlike OpenID where the identity is shared across all websites using the same identity provider?*

**FIDO Alliance**

- [FIDO Alliance](#)

This approach generates a public key pair for each relying website when registering an account with the site. This key pair is used as a proof of identity on on subsequent occasions. The proof is only generated after the user has proved her presence with a second factor. This could be a physical token (e.g. a usb dongle) or a biometric such as a finger print.

**4.2.2 Search**

Users, businesses and developers will want to able to search the marketplace. **Generic search** is based upon text queries and looks for apps and services or service
definitions that match the terms present in the query. This can be refined by also
looking for terms that are often found together with the given terms, or which are
equivalent in meaning, or selected based upon common spelling errors. The search
results can be sorted based upon the closeness of the match, the popularity of a given
result (e.g. how many people use that service, and its rating by reviewers), and
contextual information based upon who is making the query (personalized search).
Search engines can also propose results or search topics that may be of interest based
upon the aggregated behaviour of other users that are deemed to be similar to the
current user in some respect.

**Specific search** is kind of service that "knows" about a specific search domain or user
intent. A generic search engine could use heuristics to identify user intent from the
search query. The search engine maintains a registry of specific search services for
particular classes of intent. Each intent is associated with an API for invoking the
specific search service and obtaining its results. An example is where a user asks for
flights to Paris next Tuesday. The search engine then invokes search services passing
the destination, the origin (based upon knowledge of the user's location), and the
departure date. The results are concise offers for flights and prices. Specific search
essentially saves users time and effort by carrying out a domain specific task. This
needs to be done very quickly to provide the results back to the search engine for
incorporation with other results.

Specific search services are a specialized class of services for the marketplace. The
results can include actions, that if selected by the user, carries out further tasks that
may take more time than the limits imposed by search engines. This could involve the
dynamic creation of new services of a temporary nature.

4.2.3 Social requirements

A successful marketplace will require rich social interaction. This includes the means
to publish, read and respond to reviews, to rate people, apps and services, to interact
with like minded people interested in particular aspects relating to the marketplace,
and to forge communities around particular topics. Today's proprietary marketplaces
are by comparison rather sterile with limited scope and a lack of opportunities for
enrichment by an ecosystem of third party services.

For social interaction, it is important to keep track of people's status using presence
information, and to permit an exchange of chat messages. Some relevant approaches
include:

- Short message services, e.g. [Twitter](https://twitter.com)
- Audio and Video chat, e.g. [Gmail](https://mail.google.com/)
- Personal message boards, e.g. [Facebook walls](https://www.facebook.com/walls), where friends can post their
thoughts, views, or criticisms for everyone to see.
4.2.4 Payments

The marketplace needs to provide a means for supporting payments, e.g. for one off, per user or subscriptions. There are many proprietary solutions, but open standards for online payments have still to be developed. W3C is planning a workshop on Web payments for March 2014.

4.2.5 Policies and Capability Tokens

The marketplace will need to handle access control, privacy policies, and provenance. See further details.

4.2.6 Updates

There needs to be a means for automatically distributing security updates to applications and service definitions. It should also be possible to encourage users to install updates for functional improvements.

4.2.7 Data Formats and Protocols

Simple sensors provide basic data formats such as numbers (e.g. a thermometer) and booleans (e.g. a sensor for whether a window is open or closed). More generally, you can have structured data with named fields, arrays and so forth. For audio and video, there is often a need for streaming data. Some relevant approaches include:

- SIP, RTP and RTSP for real time streams
- MPEG-DASH for adaptation to changing network conditions
- WebRTC for peer to peer media streams

4.2.8 Security and Malware

A thriving marketplace will be a target for criminal activity. Countermeasures include security audits of new applications and services, and mechanisms for monitoring for unusual behaviour.

The W3C Web Application Security Working Group is developing content security policies whereby web pages can constrain the APIs that the web page scripts can use. This acts as a defense against attacks, both reducing the size of the attack surface and through enabling the browser to detect attempts to breach the constraints. A similar approach could be used for the scripts used by services to define their behaviour. For more details, see:

- http://www.w3.org/2011/webappsec/

A further consideration is to allow services to access native crypto algorithms rather than implementing their own with a consequent risk of security flaws. The W3C WebCrypto Working Group is relevant to this, see:
4.2.9 Applications

Applications essentially run on a device with a user interface, e.g. a notebook computer, a smart phone, tablet or connected TV. Applications can be locally installed onto the device or hosted in the cloud. People will want a means to manage the applications they use, e.g. to search for interesting apps, to read reviews and comparisons, to install apps, and to uninstall them. There also needs to be a means to apply security updates, preferably automatically. Some apps will be free, perhaps funded by an advertising model, whilst others may require a one off payment, a per use payment, or a paid subscription.

4.2.10 Services

Services include metadata and documentation, bindings for specific instances along with the corresponding data, scripts that define the service's behaviour, and additional resources such as graphical icons for use in depicting the service, e.g. when displaying search results, by management tools, or by developer tools. The marketplace could support scripting languages such as JavaScript and Python for developers to define the service's behaviour. Node.js is an event-driven server-side JavaScript environment based on the V8 engine, with a thriving community of developers.

- Nodejs.org
- Python.org
- JavaScript and the Internet of Things

There are many tools for code verification, online editors, JS on Java, etc.

- ST-JS -- Strongly typed JavaScript
- JSLint -- Static code analysis tool for JavaScript
- altJS -- Web coding beyond JavaScript
- Rhino -- open-source implementation of JavaScript written entirely in Java
- Orion -- Open Source Platform For Cloud Based Development
- Ace -- Embeddable code editor written in JavaScript

Service classes versus service instances

There is a clear distinction between developing a class of service and instantiating it in a particular context. For example, a company could specialize in developing services for home security. These can be purchased and bound to particular home security sensors and alarms. The Web of Things platform needs to provide the means for developers to create, market and sell service definitions that others can then instantiate as appropriate to their needs. The marketplace should allow search for both service classes and service instances.
Data flow through services

When services are composed, one service consumes data provided by another. Composite services can be driven lazily on demand by invocations of the service, or eagerly as data is pushed to them by the services they in turn depend on. There is a requirement for being able to securely store data in the cloud to support this. This may be limited for services which produce large amounts of data. The ability to look back in time is important for services that infer events probabilistically on the basis of noisy data. It can also be important for forecasting future data patterns, e.g. traffic flows through a city. Data mining algorithms are applicable to streams of data that are too large to be conveniently stored.

Service descriptions

What are the different kinds of metadata needed for services? Some examples include:

- Information about the developer of a class of service
- Information about the owner of an instance of a service
- The human readable name of the service, and a graphical icon
- A human readable introduction to the service
- Longer, in depth documentation on how to use the service
- Machine readable definitions of the service data sources and sinks
- Machine readable descriptions of the service
- Resource constraints, e.g. how much data can be cached by the run-time platform
- Access control and related kinds of policies

The human readable aspects should be localizable for different languages and cultures.

Services with a UI

In principle, services could expose a user interface rather than a data API. This would be attractive to application developers who could embed such services within their applications. This implies the need for services to be able to include UI resources such as markup, scripts, images, and style sheets.

Service related applications

When it comes to configuring services, and monitoring their use, etc. there is a need for applications that provide the user interface. These applications form an essential part of the service when viewed as a product that can be marketed to end users who are expecting a complete solution for their needs. This is justified further in this [user story.](#)
4.2.11 Marketplace APIs

The marketplace can use internal APIs to support server side generation of web pages for each kind of stakeholder. Another possibility is to expose APIs that can be invoked from clients, e.g. web page scripts. This could be of advantage when it comes to enabling third parties to compete in providing user interfaces to the marketplace. The benefits would include greater personalization and support across a wider range of devices.

User APIs

These are APIs exposed by the marketplace to registered users (end-users, developers, etc.). This includes the ability to manage their account details, to manage the apps and services they have purchased, and to search for additional apps and services.

Business APIs

These can be used to publish services, configure advertisements, and to manage analytics tools, etc.

Run-time APIs

These are APIs exposed by the marketplace to monitor and manage running services.

Service APIs

These are the APIs that the platform provides for use by services, e.g. to

• Get data that other services connected to this service's data sinks
• Push data to other services (or apps) connected to this service's sources
• Use Web protocols to communicate with sensors, actuators and gateways
• Access the service's private data storage
• Access the service's metadata
• Invoke library functions provided by the platform, e.g. for crypto algorithms
• Invoke special hardware subsystems, e.g. graphics processors

Development APIs

By defining an open interface to the developer backend, third parties can innovate with different approaches to the front end for integrated development environments. One approach is as a locally installed IDE, e.g. based upon Eclipse. Another idea is a browser based IDE. A further idea is to allow for live editing, analogous to google docs, but for software projects. This requires coordination across the clients participating in a session for a particular development project. Coordination can be cloud based, or client based. The latter can use a means to elect a master that determines what proposed changes are accepted. This involves three way merge
operations on sequences of mutations on structured information (e.g. scripts, JSON, or XML).
5 Communications technologies for the Web of Things

There are many different technologies for connecting devices, and the list keeps changing as new technologies are introduced. This makes it desirable to have an architecture that hides details that are best dealt with at a lower level. This page aims to collect pointers to communication technologies relevant to the Web of Things. Please help to expand it and to add references to appropriate external sources.

5.1 HTTP

The primary protocol for accessing resources over the Web. HTTP is popular for accessing Web of Things services, often in conjunction with JSON. HTTP can be used with transport layer security for encrypting the connection.

- Wikipedia article

5.1.1 REST

Representational State Transfer (REST) is a way of designing applications using HTTP and based upon a limited set of client-server communication primitives: GET, POST, PUT, PATCH and DELETE. HTTP together with REST is commonly used for remote access to services, where data is transferred either as XML or as JSON.

For asynchronous APIs, it is necessary to provide a means for asynchronous notifications (e.g. events) to be passed back to the client. For cloud based applications, one solution is to implement an HTTP server that can be used to deliver events. The URL for the server is passed across with the HTTP request used for invoking the service. This approach won't work for devices behind firewalls that block incoming connections. This is also the case for web browsers and it is uncommon for browsers to also function as servers. A work around is to use Server Sent Events or WebSockets.

5.1.2 Server-Sent Events

This is an API for web applications to listen to a stream of events over HTTP. The browser connects to a URL passed in the API, and is able to work through firewalls that block incoming connections.

- Wikipedia article
- W3C specification
5.2 Web Sockets

This is a web technology that allows for asynchronous exchange of messages over a TCP/IP connection. The protocol is an IETF standard defined by RFC 6455. Web applications can access WebSockets via the WebSocket API standardized by the W3C. WebSockets can be used with a variety of data types for messages, e.g. byte arrays for binary data, strings for text, and JSON for structured data. JSON-RPC can be used for remote method invocation where the method call and return data are encoded as JSON messages. The asynchronous nature of WebSockets makes it useful for passing events. WebSockets can be used with transport layer security for encrypting the connection.

- RFC 2455
- WebSocket API

5.3 Cellular Networks

Wireless technology used for mobile phones and increasingly used for mobile access to the Internet. Data can be sent in band (e.g. GPRS), via the short message service (SMS) or via AT control signals. Maximum data rates can exceed fixed line broadband connections.

- Wikipedia article on 4G

5.4 WiFi

A local RF communication technology defined by IEEE 802.11 for use with wireless networks. WiFi (or WLAN) can be operated in a managed or an ad hoc mode, and has been extremely successful in enabling portable devices such as laptop computers, tablets and smart phones to connect to the Internet.

- Wikipedia article

5.5 White Spaces

National and international authorities allocate different parts of the radio frequency spectrum for specific purposes. Guard bands are commonly allocated to avoid interference. For digital television transmissions, geographically neighboring regions avoid using the same channels to prevent interference. There is increasing interest in the potential for unlicensed use of the white spaces left by the gaps in the RF spectrum, provided that such use takes care to avoid interference with other users. The White Spaces Coalition formed by Google, Dell, EarthLink, Hewlett-Packard, Intel, Microsoft, and Philips was set up in 2007 to lobby the FCC to establish appropriate interference standards for the development of fixed and mobile devices that utilize gaps in the RF spectrum. Trials are now underway in the USA and the UK. The success of WiFi which uses unlicensed spectrum points to a huge potential for white spaces, especially given that the amount of spectrum available for white spaces is very
much larger than that used for WiFi. This is likely to create disruptive competition to existing mobile networks for machine to machine communications (M2M).

- Wikipedia article

5.5.1 Weightless

Weightless is a protocol for using white space spectrum for exchanging data between a base station and thousands of client devices. Base stations are directly connected to the Internet. Clients are allocated a schedule of times and frequencies to communicate with their base station. A database is used to avoid interference with local terrestrial TV broadcasts.

- Weightless Special Interest Group
- Wikipedia article

5.6 ZigBee

Low power wireless communications technology optimized for devices requiring a very long battery life. ZigBee devices power up the RF circuitry at prearranged intervals to save power. ZigBee can be used for sensor networks where devices pass on messages as part of ad hoc mesh networks. ZigBee is layered on top of the IEEE 802.15.4 standard for the physical layer and media access control for personal area networks.

- Wikipedia article

5.7 6LoWPAN

6LoWPAN is an acronym of IPv6 over Low power Wireless Personal Area Networks. It is layered on top of the IEEE 802.15.4 standard for the physical layer and media access control for personal area networks.

- Wikipedia article

5.8 NFC

NFC or near field communications is a short range wireless communications technology where devices must be brought into close contact (within a few centimeters). NFC tags can be used as an alternative to bar codes, e.g. for smart posters where touching an NFC equipped smart phone to the tag launches the phone's browser to a specific web page. Some example uses include ticketing on public transport, electronic door locks, exchange of electronic business cards, and digital payments with contact-less debit or credit cards. NFC tags are powered by an electromagnetic field generated by NFC readers. Self powered NFC devices can also act in a peer to peer mode, e.g. when a smart phone is presented to a point of sales terminal.
5.9 Bluetooth

A short range wireless communications technology commonly used for hands free use of mobile phones in cars, and for wire free keyboard, computer mouse and printer devices.

5.9.1 Bluetooth Low Energy (BLE)

A lower power variant of Bluetooth, defined by Bluetooth 4.0, with profiles for healthcare, sports and fitness, and proximity sensing, with a range of up to 50 metres. The technology is optimized for short packets with small amounts of data. Devices can advertise themselves, and pass data in a peer to peer model. The data can be encrypted for greater security. It is appearing in smart phones and watches, running shoes, heart monitors and so forth. BLE is gaining popularity with its adoption in Apple's iPhone (under the name of iBeacon) and PayPal's Beacon.

PayPal describes how customers can be recognized and greeted by name as they enter a restaurant and given personalized offers. On check-out, they could pull out their phone, review the check/bill and confirm payment using the PayPal app. In stores BLE could be used together with other technologies, e.g. barcodes or NFC to recognize and add individual items to the virtual shopping basket on your phone and on the store's server.

With a BLE enabled phone, you can view which devices are present at your location. Devices can advertise a URL for the web app to pass their sensor data to. The phone and app act as a bridge between the low power BLE device and the cloud. Barcodes and NFC allow you to identify a specific thing by scanning the code or touching the tag. This is important when you want further information about a specific product. Likewise, you could identify yourself to a sales assistant by tapping your phone to a reader. It thus seems that BLE and NFC are complementary.

5.10 ANT

A proprietary sensor network technology operating in the 2.4 GHz band. It can be used to transfer small amounts of data across networks with hundreds of sensors.

• see Wikipedia article on ANT
5.11 Infrared

Widely used for remote control of TVs, air conditioners etc. Infrared was popular for PDAs and laptops in the late 90's and early 2000's, but lost ground to RF technologies such as WiFi and Bluetooth. Infrared is making a comeback for fast transmission of photos from phones to printers etc.

- see Wikipedia articles on IrDA and Consumer IR

5.12 USB

The Universal Serial Bus (USB) is an industry standard defining cables, connectors and protocols. It is widely used for connecting devices to computers, e.g. keyboards, mouse pointers, hard drives for storage, game controllers, and also for connecting to printers, scanners, digital cameras, smart phones and tablets. USB is designed to power devices and is commonly used for charging device batteries, replacing the need for a separate cable. There are several kinds of USB connectors, e.g. standard, mini and micro connectors, with the latter being widely used for smart phones and tablets. Data rates have increased substantially over successive generations of USB standards, e.g. USB 3.0 supports a rate of 5 Gbits/s. There are standards for common types of device, e.g. keyboard, pointing devices and mass storage, but for other devices, a device specific driver is typically required.

- Wikipedia article
- USB.org

5.12.1 Wireless USB (WUSB)

A standard for connecting devices using a wide band protocol in the 3.1 GHz to 10.6 GHz region. The range is 3 to 10m.

- Wikipedia article

5.13 IEEE 1394 (Firewire)

A serial connection designed for high speed transfers, and similar in some ways to USB. IEEE 1394 has lost ground to USB as the latter has increased in speed, and due to the need for a separate power connection for Firewire devices.

- Wikipedia article

5.14 DASH7

Designed for long lived battery operated sensor networks, it works in the 433 MHz unlicensed band. The range is up to 1000m depending on power levels and data rates. Like ZigBee and BLE, DASH7 is aimed at transferring small amounts of data, and unsuitable for audio or video.
5.15 KNX for buildings

KNX is a standardized (EN 50090, ISO/IEC 14543), OSI-based network communications protocol for intelligent buildings. KNX is the successor to, and convergence of, three previous standards: the European Home Systems Protocol (EHS), BâtiBUS, and the European Installation Bus (EIB or Instabus). The KNX standard is administered by the KNX Association.

KNX can be realized over a mix of networking technologies, e.g. twisted pair cable, powerline networking, radio (KNX-RF), infrared and conventional ethernet.

5.16 EnOcean

EnOcean is a similar protocol to KNX for sensors that are self powered, e.g. harvesting energy when you push a switch that is sufficient for sending 2 or 3 packets. EnOcean was spun off by Siemens (2006?). The sensors are quite expensive (e.g. 60 CHF) but available for motion sensors (light and thermal IR), beds, seats, window handles and so forth.
6 Techniques for specifying services

6.1 Machine Interpretable descriptions

This section covers the means to support machine interpretable descriptions of services. The starting point is the idea of using globally unique names for things and relationships between them. The names are expressed as URIs, e.g. HTTP URLs such as: http://purl.org/dc/terms/abstract which is a term for a human readable abstract. This approach was first popularised in the following Scientific American article:

• The original 2001 Scientific American article by Tim Berners-Lee, Jim Hendler, and Ora Lassila

6.1.1 RDF

W3C's Resource Description Framework (RDF) is based upon labeled relationships in the form of a three place tuple, or triple:

• <subject, verb, object>

Where the subject, verb (the label), and the object are all named with HTTP URLs. RDF was originally proposed with an XML serialization, but other formats have since become widely used, e.g. Turtle, N3 and more recently JSON-LD.

The following link is to the W3C page describing the relationships between the suite of specifications covering RDF. The Wikipedia article provides links to additional resources and tools relating to RDF.

• RDF Current Status
• Wikipedia article

6.1.2 Linked Data

RDF forms the basis for the Web of linked data. This is formed by the collection of servers that host linked data. The number of sites has risen dramatically in the last few years, mirroring the growth of the Web in the 1990's. The following illustration of the Web of linked data is from the The linking open data cloud diagram from September 2011.
The following table gives the corresponding number and size of datasets by domain, and is taken from State of the LOD Cloud document:

<table>
<thead>
<tr>
<th>Domain</th>
<th>Data Sets</th>
<th>Triples</th>
<th>Percent</th>
<th>RDF Links</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Media</td>
<td>25</td>
<td>1,841,852,061</td>
<td>5.82</td>
<td>50,440,705</td>
<td>10.01</td>
</tr>
<tr>
<td>Geographic</td>
<td>31</td>
<td>6,145,532,484</td>
<td>19.43</td>
<td>35,812,328</td>
<td>7.11</td>
</tr>
<tr>
<td>Government</td>
<td>49</td>
<td>13,315,009,400</td>
<td>42.09</td>
<td>19,343,519</td>
<td>3.84</td>
</tr>
<tr>
<td>Publications</td>
<td>87</td>
<td>2,950,720,693</td>
<td>9.33</td>
<td>139,925,218</td>
<td>27.76</td>
</tr>
<tr>
<td>Cross-domain</td>
<td>41</td>
<td>4,184,635,715</td>
<td>13.23</td>
<td>63,183,065</td>
<td>12.54</td>
</tr>
<tr>
<td>Life sciences</td>
<td>41</td>
<td>3,036,336,004</td>
<td>9.60</td>
<td>191,844,090</td>
<td>38.06</td>
</tr>
<tr>
<td>User generated</td>
<td>20</td>
<td>134,127,413</td>
<td>0.42</td>
<td>3,449,143</td>
<td>0.68</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>295</strong></td>
<td><strong>31,634,213,770</strong></td>
<td><strong>0.42</strong></td>
<td><strong>503,998,829</strong></td>
<td><strong>0.68</strong></td>
</tr>
</tbody>
</table>

For small data sets, it is possible to use simple scripts for custom queries. For large data sets, you can use the SPARQL query language. Here is an example for a query that asks "What are all the country capitals in Africa?":

```
PREFIX abc: <http://example.com/exampleOntology#>
SELECT ?capital ?country
WHERE {
  ?x abc:cityname ?capital ;
  abc:isCapitalOf ?y .
  ?y abc:countryname ?country ;
  abc:isInContinent abc:Africa .
}
```
The W3C Linked Data Platform Working Group is developing standards for HTTP-based (RESTful) application integration patterns using read/write Linked Data, see:

- W3C Linked Data Platform Working Group

Websites can add structured data in RDF to HTML pages for smarter support by search engines. The approach is called Rich Structured Data Markup for Web Documents, or more simply RDFa. For an introduction to using RDFa, see:

- RDFa primer

The following book by Tom Heath and Christiam Bizer provides an extended introduction to linked data:

- Linked Data: Evolving the Web into a Global Data Space

### 6.1.3 Some relevant vocabularies

The marketplaces for the Web of Things relates to the Web of linked data in two ways:

- Services that generate linked data, e.g. from sensors and interpretations thereof
- Service descriptions expressed using linked data

RDF can be applied to data modelling. The starting point is the RDF Schema (RDFS), which defines terms for resource, class and literal. The Web ontology language (OWL) adds a richer set of modelling terms. SKOS is a standalone vocabulary intended for modelling thesauri in RDF.

Further work is needed to survey all of the potentially relevant data vocabularies.

A few common vocabularies include:

- Friend of a Friend (FOAF)
- Description of a Project (DOAP)
- Dublin Core (DC)
- Semantically-Interlinked Online Communities (SIOC)
- of Interlinked Datasets (VoiD)
- vCard
- RDF Schema (RDFS)
- Web Ontology Language (OWL)
- Simple Knowledge Organization System (SKOS)
- Semantic Sensor Net Ontology

The Linked Open Vocabularies (LOV) dataset covers the RDFS vocabularies or OWL ontologies for and used by datasets in the Linked Data Cloud. A similar collection is available at:
Note: the work the Semantic Sensor Net Ontology was developed by the W3C Semantic Sensor Network Incubator Group. This work has now been taken over by the Semantic Sensor Networks Community Group.

The main search engines (Bing, Google and Yahoo!) jointly started Schema.org in June 2011 to collect terms that webmasters can use to markup their sites to improve the display of search results. The schems.rdfs.org site provides access to the schema.org terms in various formats including RDF/Turtle, RDF/XML and RDF/NTriples. They also maintain mappings from Web Data vocabularies such as the DBpedia ontology to Schema.org terms. This should also be applicable to search across open marketplaces for the Web of Things.

Note: DBpedia is crowd-sourced community effort to extract structured information from Wikipedia and make this information available on the Web. DBpedia allows you to ask sophisticated queries against Wikipedia, and to link the different data sets on the Web to Wikipedia data.

6.2 Defining Service data sources and sinks

This section covers techniques that can be used to specify the protocol or API for accessing services for the Web of Things. This includes the use of WebIDL for APIs, XML and JSON for structured data, Web Services and RESTful HTTP.

6.2.1 Web IDL

Web IDL is an interface definition language that is commonly used in W3C specifications for JavaScript APIs exposed by web browsers. It includes a suite of predefined data types including numbers, strings, arrays, as well as support for enumerations, unions and call backs. The Web IDL specification includes a binding to JavaScript (formally known as ECMAScript), but bindings to other languages are possible.

- Editor's draft for Web IDL
- ECMA-262 3rd Edition

Here is an example of a Web IDL fragment:

```idl
exception GraphicsException {
    DOMString reason;
};

interface Paint {
};

interface SolidColor : Paint {
    attribute float red;
    attribute float green;
    attribute float blue;
};
```
interface Pattern : Paint {
   attribute DOMString imageURL;
};

[Constructor]
interface GraphicalWindow {
   readonly attribute unsigned long width;
   readonly attribute unsigned long height;

   attribute Paint currentPaint;

   void drawRectangle(float x, float y, float width, float height);

   void drawText(float x, float y, DOMString text);
};

6.2.2 XML

The Extensible Markup Language (XML) is a simple and flexible text format derived from SGML (ISO 8879). Angle brackets are used to introduce mark-up tags into a text stream. The permitted structure of an XML document can be constrained to match a formal grammar, e.g. using the XML Schema language. XML is frequently used for representing structured data and as such can be used for messages as part of protocol used for accessing a service. XML is defined by a suite of standards produced by the W3C.

- [W3C XML Activity page](#)

Here is a small example of a part of an XML document:

```
<part number="1976">
   <name>Windscreen Wiper</name>
   <description>The Windscreen wiper automatically removes rain from your windscreen, if it should happen to splash there. It has a rubber <ref part="1977">blade</ref> which can be ordered separately if you need to replace it.
</description>
</part>
```

6.2.3 JSON

The JavaScript Object Notation (JSON) is a popular lightweight syntax for data exchange. It includes collections of name/value pairs, and a set of built-in data types, e.g. numbers, booleans, strings and arrays. JSON can be readily parsed into the native data types for languages such as JavaScript and Java.

- [JSON.org](#)
- [JSON (RFC 4627)](#)
Here is an example:

```json
{"menu": {
   "id": "file",
   "value": "File",
   "popup": {
      "menuitem": [
         {
            "value": "New", "onclick": "CreateNewDoc()"
         },
         {
            "value": "Open", "onclick": "OpenDoc()"
         },
         {
            "value": "Close", "onclick": "CloseDoc()"
         }
      ]
   }
}}
```

JSON has been extended to support pointers, linked data and schemas:

**JSON Pointer**

JSON Pointer defines a syntax for identifying a specific value within a JSON document.

- **JSON Pointer (RFC 6901)**

The following examples illustrate the use of JSON pointers in URI fragments for a JSON text document located at `http://example.com/example.json`

```json
{
   "foo": {
      "bar": [ "element0", "element1" ],
      "inner object": {
         "baz": "qux"
      }
   }
}
```

- `http://example.com/example.json#` Resolves to the object value at the root of the JSON text document.
- `http://example.com/example.json#/foo` Resolves to the object value of the "foo" member in the root object.
- `http://example.com/example.json#/foo/inner%20object/baz` Resolves to the string value "qux", which is the value of the "baz" member in the "inner object" member in the "foo" member in the root object.
- `http://example.com/example.json#/foo/bar/0` Resolves to the string value "element0", which is the first value in the "bar" array in the "foo" member in the root object.

*with thanks for the examples to Paul C. Bryan.*
JSON-RPC

JSON-RPC is a framework for remote procedure calls expressed using JSON. An identifier is used to associate each response with its request. JSON-RPC is often used in conjunction with WebSockets.

- JSON-RPC.org

Here is an example for a service that adds one to the integer passed to it:

```json
--> { "method": "increment", "params": [65], "id": 7 }
<-- { "result": 66, "error": null, "id": 7 }
```

JSON Linked Data

JSON-LD is a set of conventions for representing linked data in JSON. Linked data provides a way to create a network of standards-based, machine-readable data across Web sites. For the Web of Things JSON-LD offers a way of exchanging rich metadata about things and services based upon them.

- JSON Linked Data (W3C)
- JSON-LD.org

Here is a small example:

```json
{
    "http://schema.org/name": "Manu Sporny",
    "http://schema.org/url": { "@id": "http://manu.sporny.org/" },
    "http://schema.org/image": { "@id": "http://manu.sporny.org/images/manu.png" }
}
```

This is equivalent to the following JSON:

```json
{
    "name": "Manu Sporny",
    "homepage": "http://manu.sporny.org/",
    "image": "http://manu.sporny.org/images/manu.png"
}
```

JSON-LD uses IRIs to identify the name space for each name. @id is a predefined term that indicates that the value is an IRI (essentially a variant of web addresses that allows for non-western characters).

JSON Schema

JSON Schema can be used to define the structure of JSON data for the purposes of validation, documentation, and interaction control.

- JSON-Schema.org
Here is a small example of a schema:

```json
{
  "title": "Example Schema",
  "type": "object",
  "properties": {
    "givenName": {
      "type": "string"
    },
    "familyName": {
      "type": "string"
    },
    "age": {
      "description": "Age in years",
      "type": "integer",
      "minimum": 0
    }
  },
  "required": ["givenName", "familyName"]
}
```

and here is an example of a JSON document that conforms to it:

```json
{
  "givenName": "John",
  "familyName": "Smith",
  "age": 42
}
```

### 6.2.4 Web Services

This term generally refers the the use of XML for services based upon the Simple Object Access Protocol (SOAP). Services can be specified using the Web Services Description Language (WSDL).

- [W3C Web Services Architecture](#)
- [W3C Web Services Activity](#)

The W3C Web Services Activity page provides links to a suite of Web Services specifications.

Here is an example message:

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:m="http://www.example.org/stock">
  <soap:Header/>
  <soap:Body>
    <m:GetStockPrice/>
  </soap:Body>
</soap:Envelope>
```
Some related standards:

- **WSDL** -- Web Services Description Language
- **SOAP** -- Simple Object Access Protocol
- **UDDI** -- Universal Description, Discovery and Integration

Web Services have been widely used for enterprise applications, but in recent years its popularity on the public Internet has declined in favour of lighter weight alternatives involving RESTful HTTP together with JSON.

### 6.2.5 RESTful HTTP

HTTP provides a popular means for accessing services. In some cases, a device can integrate sensors, actuators and an HTTP server, or a device can act as a gateway, e.g. using ZigBee to communicate with sensors, and exposing their data via HTTP. In many cases, devices will be isolated behind security firewalls. In this situation, services can be exposed by servers in the cloud that hide the details of the communication with the sensors and actuators.

HTTP is a request/response protocol that can be used to implement remote procedure calls. The term representational state transfer (REST) was introduced and defined in 2000 by Roy Fielding, one of the principal authors of the HTTP specification. REST is essentially an architectural style for using HTTP in a scalable way.

- **REST as defined in Roy Fielding's dissertation**

The request specifies an HTTP method, and the appropriate choice of method depends on the nature of the service that is invoked. Here are the methods used for RESTful services:

**GET**

Is used for services that are side effect free, as this allows the response to be cached.

**POST**

Is used for services that aren't side effect free, e.g. operating an actuator.

**PUT**

Is used to store a specified resource, e.g. saving an image passed in the request.

**PATCH**

Is used to make a partial modification to the specified resource.
DELETE

Is used to delete the specified resource.

PUT and DELETE are idempotent, i.e. multiple identical requests should have the same effect as a single request.

HTTP requests and responses consist of a set of named headers and a body. If the body is present, its type is indicated by the Content-Type header. Responses also include a status code, e.g. "200 Okay" or "404 Not found". To specify HTTP based services, it is therefore necessary to define the data passed in the request, the method used, the data returned in the response and the response code.

Some services will result in asynchronous notifications or events. There are several approaches to implementing this with HTTP. One is to pass an HTTP URL to the service for where to deliver the notification. Another is to poll the service (using POST) to see what new notifications are available. A refinement is to have an extended HTTP response where the connection to the client is kept open, and the body is sent as a sequence of messages. The first approach won't work if the recipient for notifications is hidden behind a firewall, as in the firewall will block the incoming connection unless the firewall has been specifically configured to allow it. The latter two approaches are often used for pushing notifications to web page scripts executing in browsers. Service definitions need to specify the details for the approach used for notifications.
7 Ethical considerations for the Web of Things

With sensors and actuators throughout cities and homes, there is a risk that the Web of Things is abused by regimes wanting to take surveillance of their citizens to the extreme, so that Big Data == Big Brother (see George Orwell's dystopian novel "Nineteen Eighty-Four"). Other fears relate to abuse of privacy by companies that collate information to build detailed profiles on users, or which purchase or otherwise obtain data collected by the state. If insufficient attention is paid to security, this could open the door to discrimination and criminal activities.

A more recent perspective is given in Dave Egger's novel "The Circle". The Guardian columnist Simon Jenkins (front page, 25 October 2013) describes the novel as satirising Silicon Valley's "completion of the singularity". Each person becomes a global avatar, bugged and followed 24 hours a day. The trail we leave in our online activities together with pervasive sensors, and ever smarter software will make it ever easier to track people throughout their lives in intimate detail.

The PEW report on The Future of Big Data includes quotes by people who are concerned by the potential negative impact. Separately, Ajit Jaokar suggests that Open Source could be helpful as many eyes can help to detect security backdoors and potential security exploits that snoopers could take advantage of.

This page is intended for an exploration of the ethical issues associated with the Web of Things, and some of the ways that we can create technical, social and legal safeguards to limit abuse.

7.1 Pervasive Surveillance

The disclosures by Edward Snowden of the activities by the US NSA and the UK GCHQ have shone a light on the ability for state institutions to gather information on a massive scale by tapping Internet cables, and through backdoor arrangements with Internet companies.

- Wikipedia article on Edward Snowden

The Web of Things threatens to enable even richer surveillance through sensors in people's homes, their person (smart phone, health monitors), in the street, in stores and at work. Whilst most people agree with the need for surveillance to counter terrorism, the fear is that once gathered the information will be accessible for other purposes, and could be used to discriminate people based upon their personal habits, employment status, religion, their friends and relations, and so forth.

To avoid this, laws and codes of practice have been drawn up, e.g. the following document on the use of the widely deployed surveillance cameras in the United Kingdom:

- UK Home Office Camera Code of Practice
According to the New York Times US agencies have pressed the NSA for access to surveillance data. This includes agencies working to curb drug trafficking, cyberattacks, money laundering, counterfeiting and even copyright infringement. The NSA claims to limit use of their data to avoid misuse that violate Americans' privacy.

In the UK, restrictions have been placed upon local authorities to limit their ability to authorise directed surveillance, e.g. to investigate benefit fraud or to gather evidence of anti-social behaviour. The revisions to the law states that authorising officers may not approve directed surveillance unless it is for the purpose of preventing or detecting a criminal offence.

Unfortunately, the laws and codes of practice may be breached by individuals able to work around the system. Examples include:

- News International phone hacking scandal
- Hundreds of police officers caught illegally accessing criminal records computer
- Private investigators accessed police files, says leaked report

The risk of such abuse is likely to rise as the Web of Things expands. Counter measures include increased security, and audit trails of who accessed what data, when and for which reasons.

7.2 Combining the Internet of Things with other sources of information

Search engines, advertisers, websites and others collect information on how people are using their services. This can be used to provide personalized services and more relevant advertising, but it presents a risk of abuse when data is put to purposes other than the intended ones. Data retention is an issue, e.g. where teenage photo's surface in later life. A further problem is where mistakes are made as it can be hard to track down and correct them. Such mistakes could be intentional, e.g. an attempt to place misinformation to damage someone's public standing or credit worthiness. Criminals can break into corporate websites to gain illegal access to records, e.g. for identity theft and the use of credit card details for fraud.

There are increasing concerns over fingerprinting where multiple cues are combined to uniquely identify a device, and hence the user, even when that user has indicated a desire not to be tracked. The use of cloud based services threatens to make this worse through the ability to combine personal data from sensors with other sources of information.

- How unique is your browser - Panopticlick, EFF
- Cookieless Monster: Exploring the Ecosystem of Web-based Device Fingerprinting
7.3 Implications for Standardization

Security and privacy are important considerations for proposed standards. The W3C Web Security Interest Group's mission is to serve as a forum for discussions on improving standards and advancing the security of the Web. This includes review of specifications proposed by other W3C groups. The W3C Device APIs Working Group, for example, closely examines API specifications to minimize personal information leaked through the use of those APIs. Trusted applications are able to get richer access to the device and personal data, but this trust needs to be justified, e.g. through external review, and the reputation of the application developers.

Standards for the Web of Things will need to carefully consider security and privacy, and this has implications for how identity, authentication, provenance and access control are managed.
8 W3C Web of Things Community Group

A W3C Community Group is an open forum, without fees, where Web developers and other stakeholders develop specifications, hold discussions, develop test suites, and connect with W3C's international community of Web experts.

The mission of the Web of Things Community Group (CG) is to accelerate the adoption of Web technologies as a basis for enabling services for the Internet of Things. The Web of Things CG was launched as a means to foster discussion across different stakeholders. In order to achieve this mission, the group will bring representatives of key stakeholders together to:

- Collect use cases as a basis for identifying requirements
- Develop materials describing an architecture for the Web of Things
- Review existing standards and their applicability
- Identify gaps where new standards would be appropriate
- Develop proposals for new standards as needed
- Identify opportunities for creating broader awareness of the Web of Things
- Engage with the developer community to gather implementation experience

The roadmap set out in the charter calls for:

- Initial analysis by the end of October 2013
- Draft proposals for new standards by the end of October 2014
- Revised proposals and charters for their standardization by end October 2015

Standardization should follow implementation experience and a consensus on core use cases. In some cases, proposals could be submitted to existing W3C Working Groups, and in others, new Working Groups could be proposed. The Community Group may find it appropriate to recommend work in other standards development organizations such as the IETF. Any work that is intended for use in W3C standards specifications will be subject to the Contributor License Agreement (CLA).

Further information can be found at:

- [http://www.w3.org/community/wot/](http://www.w3.org/community/wot/)
9 W3C Web of Things Standardization Workshop

W3C Workshops provide a venue for bringing people together for intense discussions relating to the timeliness and scope of potential standards work, and are frequently a precursor to initiating new Working Groups at W3C. Workshops are free of charge. All participants are required to submit position statements which are evaluated by the program committee who make recommendations for which participants to invite to give presentations or to appear in panel sessions.

W3C plans to hold a standardization workshop on the Web of Things during 2014 at a European location. This will be an opportunity to share ideas with other groups and to get valuable feedback on technical directions. The date and venue have yet to be determined. A Call for Papers will be announced once the chairs, program committee, venue and date have been decided.

Note that W3C holds several such workshops each year. See:

- [http://www.w3.org/2003/08/Workshops/](http://www.w3.org/2003/08/Workshops/)
10 Conclusions

As this is the initial standardization report for the Compose project, we are still at an early stage in understanding the opportunities for new standards arising out of the innovation that the project creates. Compose covers a broad range of requirements, and we are seeking to build as much as we practically can on existing work. This includes:

• HTTP, Web Sockets and JSON for accessing services and the marketplace itself
• Server-side scripting for defining behaviour, e.g. JavaScript and Node.js
• Linked data and existing vocabularies for use in metadata
• Current approaches to identity, authentication and security
• Existing connection technologies for IoT devices, e.g. ZigBee and Bluetooth

As we identify gaps, we will have an opportunity to contribute to future standards work. This will be the focus for the second standardization report D9.1.2. The ability for Compose to influence the future will depend upon how effectively we engage with other groups during the course of the project. We have launched the W3C Web of Things Community Group to foster that engagement, and will also exploit the opportunities for face to face discussion at the standards workshop, that W3C plans to hold in 2014.
11 Related Work

This is a collection of links to related work including research projects. See also Communications technologies for the Web of Things.

- **IERC European Research Cluster on the Internet of Things**
- **Internet of Things activities in the ITU**
- **Smart Cities Expo World Congress** -- annual trade show in Barcelona
- **W3C Semantic Sensors Network Community Group**
- **Compose Project** -- enabling open markets of services for the Internet of Things
- **Webinos project** -- web platform for securely sharing access to personal devices
- **IoT-A project** -- architectural reference models for the IoT
- **iCore project** -- virtualisation framework for services for the Internet of Things
- **Hub of All Things (HAT digital vault)** -- UK research project on enabling a market of services based upon data collected on personal behaviour, where all data generated by the individual is owned by that individual
- **IoT.est project** -- composition/orchestration of services, auto configuration and testing
- **ProbeIT** -- supporting exploitation of European research advances in IoT deployments
- **Agora** -- French consortium focusing on smart homes
- **Home Gateway Initiative (HGi)** -- forum for discussion of specifications and standards for devices serving as gateways between residential networks and wide area networks
- **ASHRAE BACnet** -- protocol for building automation and control networks
- **ZigBee Alliance** -- low power wireless communications protocol for battery operated devices, and note that the Zigbee gateway provides support for SOAP, REST and GRIP protocols for remote access to ZigBee based devices.
- **Open Standard For The Smart Homes Of The Future** -- ABB, Bosch, Cisco, and LG are setting up a consortium to provide a software platform for smart homes. (announced 28 October 2013)
- **OpenRemote** -- open source middleware for the Internet of Things
- **Early talk on the Web of Things** from September 2007

OneM2M is a global initiative for machine to machine M2M standardization. This seems focus on the role of mobile network operators in enabling data transfer over cellular networks.

- **oneM2M website**
11.1 Webinos project

Webinos is an EU funded FP7 project aiming to deliver a platform for web applications across mobile, PC, home media (TV) and in-car devices. Webinos demos include using a phone to control a TV, home automation and home healthcare.


A key aspect of Webinos is the concept of a *Personal Zone* as illustrated below:

Each of us has an increasing number of personal devices along with the apps and services that run on them. The *Personal Zone* is basis for people to manage their personal devices. All Webinos devices include an agent (the *Personal Zone Proxy*) that integrates the device into the zone. Webinos applications run within web browsers. The Personal Zone is exposed to trusted application with a suite of Webinos APIs. These include the means for applications to discover and bind to services on devices within the zone.

The Zone is visible on the Internet via the *Personal Zone Hub* which can be implemented on a personal device or as part of the cloud. You can install apps and services onto the Hub for inter zone applications. Intra zone messaging is possible either peer to peer or indirectly through the Personal Zone Hub. Zone APIs are subject to access control policies. You can thus determine just which APIs and services are available to your friends. The [webinos project](http://www.webinos.org/) implements Personal Zones with an embedded agent (a Personal Zone Proxy) using node.js. Public key certificates issued
by the Personal Zone Hub are used for authentication of devices. Internet of Things
devices either embed a Personal Zone Proxy or are connected to a device that is.
Webinos restricts devices to belonging to a single zone. For shared devices like TVs,
the zone owner needs to set the policies for sharing the device with other people.

Personal Zones support peer to peer applications, and give users back direct control
over their personal data, avoiding the need for handing over personal data to
centralized databases and services hosted by Internet giants such as Google and
Facebook. Personal Zones thus correspond to Personal Data Vaults. The URI for your
Personal Zone Hub can be discovered from your email address using the Web Finger
protocol.

11.2 Internet of Things activities in the ITU

The International Telecommunication Union (ITU) is the United Nations specialized
agency for information and communication technologies.

11.2.1 Joint Coordination Activity on Internet of Things (JCA-IoT)

JCA-IoT is a forum for the discussion of the Internet of Things, and open to ITU
Members and designated representatives of relevant SDOs.

- JCA-IoT

The goal of JCA-IoT is as follows;

1. To coordinate the ITU-T work on network aspects of identification systems
   initially focusing on the work in the SGs on terms and definitions, the output
   of the correspondence groups and the technology watch report on USN.
2. To collect and analyze the list of standardization items and associated
   roadmap;
3. To forward urgent standardization issues to relevant SGs and other SDOs as
   appropriate;
4. To act as a single point of contact within ITU-T with other SDOs in order to
   avoid duplication of work;
5. To assist coordination between ITU-T Study Groups and other ITU groups
   and to include their representatives; and
6. To carry out an external collaboration role with other relevant recognized
   SDOs and to invite their representatives.

Many draft Recommendations refer to W3C Recommendations, and it is highly
related to W3C activity. It is worthwhile to advertise IoT related W3C technologies to
ITU-T and other SDOs.

The International Telecommunications Union's Recommendation ITU-T Y.2063
provides a functional framework for the Web of things as a basis for unified access to
devices through a web-based service environment, as well as through legacy
telecommunications.
11.3 Compose Project

The Compose project is an EU FP7 research project that seeks to enable an open marketplace of services for the Internet of Things.

- Compose project home page [http://www.compose-project.eu/](http://www.compose-project.eu/)

The following illustrates the architecture as given in the project proposal:

![Diagram of Compose Marketplace](image)

Note: the following is subject to change as the project proceeds.

The Compose Marketplace is a cloud based platform implemented using [Cloud Foundry](https://www.cloudfoundry.org/). The starting point are the sensors and actuators shown on the right hand side. These are registered as *Compose Service Objects* within the Compose platform. Sensors and actuators are required to support HTTP, and a gateway is assumed if the devices don't support HTTP directly. Service objects are essentially transparent proxies for sensors and actuators.

The services layer allows developers to create services with customizable behaviour, using Java or perhaps JavaScript. A *Compose Service* can be bound to service objects or other services. Compose Services are exposed via HTTP. Developers can provide applications that exploit these services.