Interoperability and…
The Internet of Whatever

Alistair Munro, Airbus

A Personal View

This is not an Airbus opinion!

The work was done to support the innovateUK IoT-Bay project.

Some History

- Go back a few 1000 years – things that fight
- Then skip forward a few 1000…
- For example
  - Railways
  - Telephone networks, fixed, mobile
  - Variations of TCP
  - **Home and building systems (HBES)**
  - Things that fly
- Lessons
  - Different systems can coexist
  - Solve problems, don't create them

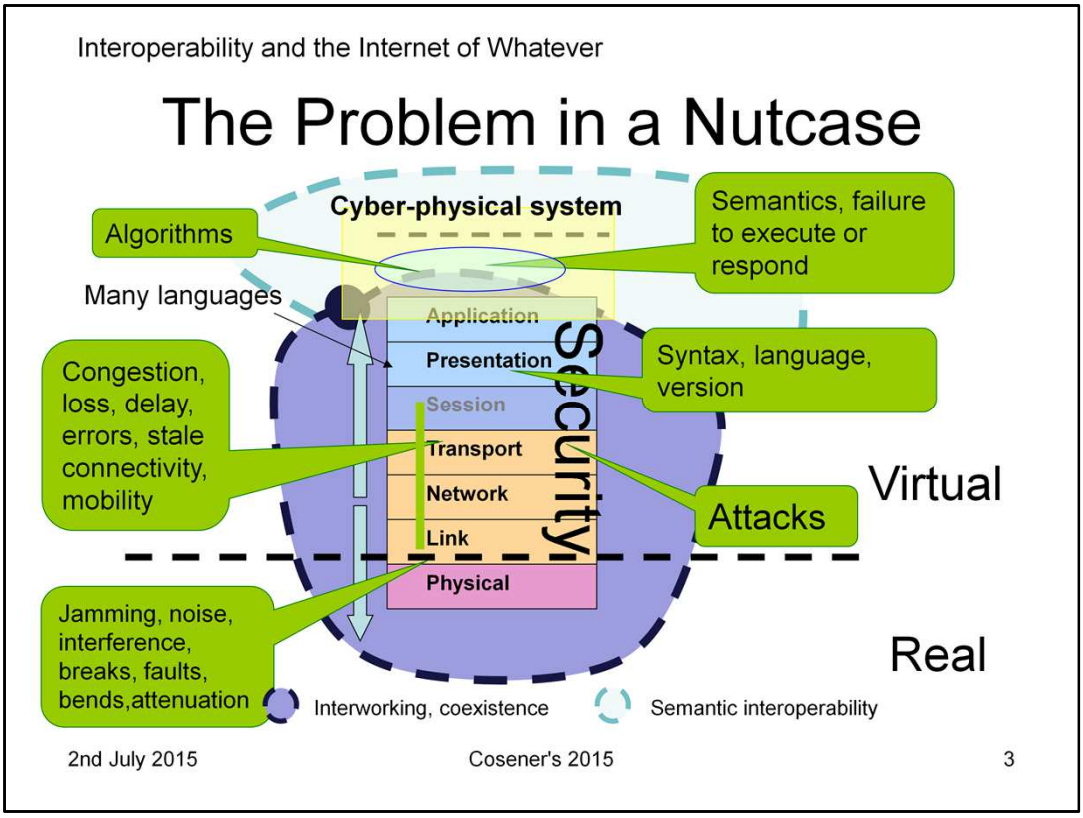
The ancient Greeks understood interoperabiity because they used complex signalling protocols to manage battles.

This persists today, e.g. NATO's 3 levels of interoperability to stop troops killing each other.

It never goes away: railways in different countries use different signalling systems – they never need to interoperate but drivers must understand them for each country – user-level semantic understanding; telephone networks interoperate very well but they are different everywhere – coexist and interwork, except mobile communications; TCP variants can talk to each other but "speak" TCP in different ways process semantic mis-understanding; more about HBES later – mixture of coexistence, interworking and interoperability, some or none; aircraft and aircraft systems must interoperate completely throughout their life - everything.

So interoperability has 3 dimensions: coexistence – don't destroy each other; interworking – messages can be transferred/translated between different systems; [semantic] interoperability – can interact and use information correctly. Reality is a continuum in all 3 dimensions.

The major lessons: (1) interoperability is neither necessary nor sufficient, it can create silos and exclusions unnecessarily as well as breaking down boundaries. We can live with separate and different systems; (2) trying to model human intent is probably futile with current tools – interesting research but does not make things work, for example, aviation safety is built on holes in the ground and widows' compensation – interoperability has been a win but long and hard. Avoid taking a path that adds another level of indirection – you will argue about your models, spend all your time trying to validate them, and forget what you were trying to achieve in the first place.



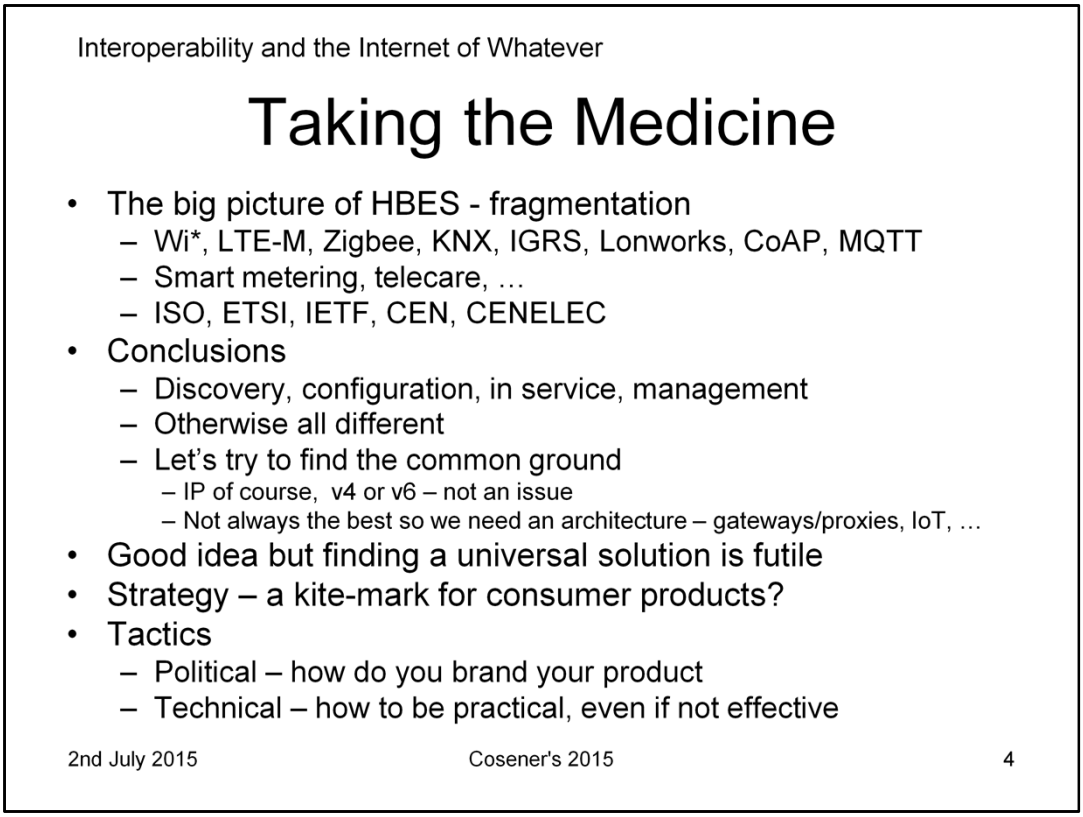
This says it all…

Each layer has its own coexistence, interworking and interoperability problems. Each is an interoperability domain in its own right: if you don't have a common understanding of naming, addressing and administrative scope, of quality, of security, of managing multiple traffic flows, and of routing then it will be difficult to build a working system from the bottom up to the end-to-end at least. The same issues apply in the end systems to a greater or lesser extent right up to the individual application and beyond to its interaction with the real world and with other enterprises.

Like QoS, it has to be there [coexist, interwork or interoperate] from the ground up. Failures at lower layers are possibly cruder and there are more of them. Higher layer failures are more subtle, abstract, and difficult to address – they may span many interactions.

The green boxes illustrate things that go wrong. They may be faults. They may be attacks. They may be interoperability problems: many security controls are sensitive to time delays and [even normal] variations may alarm them.

Security controls and mechanisms and countermeasures should [must?] be implemented but it must be done properly and in a way that the weakest element [the human user] cannot subvert them. They may appear at any layer. They must be verified to be effective. They must be maintained through life.



Let's look at Home and Building Electronic Systems [ISO term], HBES.

It is wildly diverse and fragmented – many technologies, many applications, many standards bodies claiming ownership. I show a limited selection – apologies if your favourite is not there.

All claim interoperability as a property that can be tested and certified.

We surveyed the scene, as others have done, to find out what is common – if anything. If we stand back a bit, we can see four types of function:

- Discovery – who is out there and what are they? Some systems are bounded by a physical medium (wireless coverage, range of a power line – more than you thnk or want). It is easy to build in a gateway to make local devices visible outside by proxy – not a good solution but necessary in many cases;
- Configuration – making/breaking associations with the entities [objects, devices, services, data, …] that will participate in the application, [maybe] authenticating them and securing the end-to-end relationships, sharing parameters. Entities may be shared, or not;
- Operation (in-service) – exchange of "application semantics" and acting on the information. The application may tolerate more-than-once, at-most-once or exactly-once semantics;
- Management – not always easy to distinguish from other function types, e.g. maintaining a time reference – probably part of the application in-service functions. Probably any function that is outside the others: building an overall map of a locality, monitoring traffic and device health, blacklisting entities, forcing disconnection.

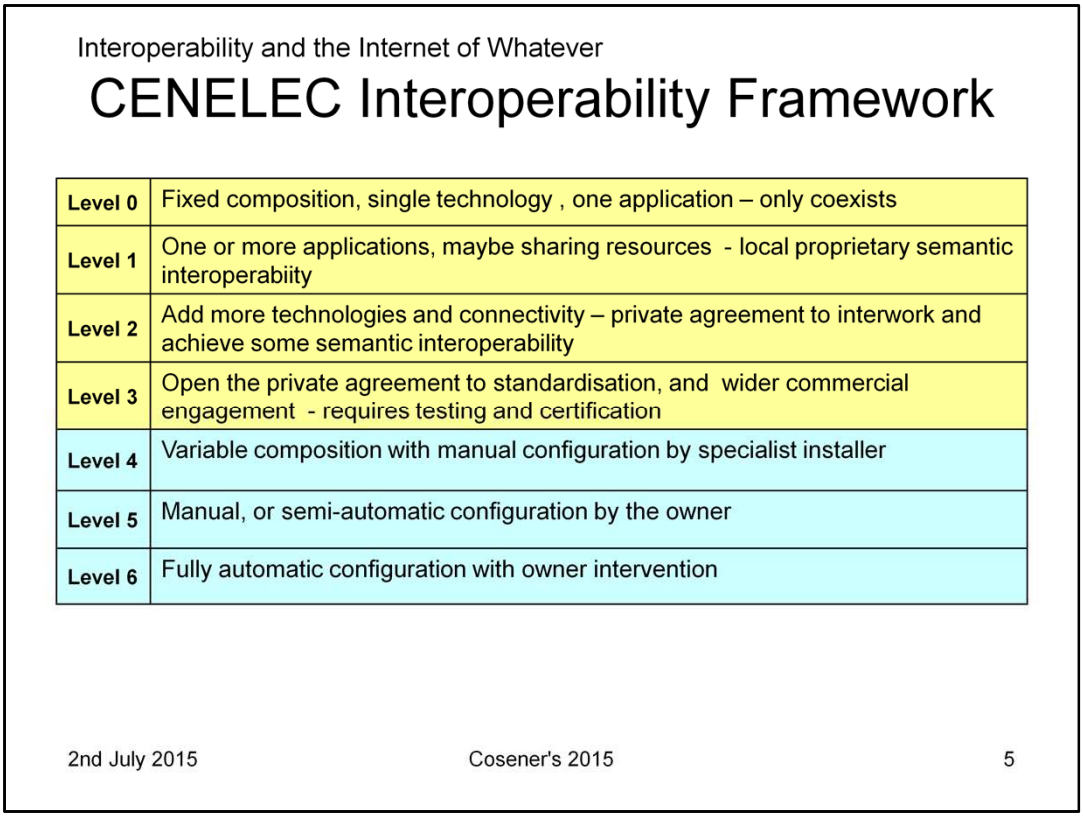
Nothing fits this model perfectly. But from a distance it works and it gives a top-level breakdown of measuring the potential to interoperate and to document it. No all specifications do everything: some, such as uPnP, are facilitators for discovery and configuration.

We must be practical so let's exploit something that is inherent in all the technologies – some sort of IP platform. The right-hand side is easy because CoAP, MQTT are native IP application layers that can be used for any application. Moving left, systems like Lonworks, IGRS and KNX offer IP connectivity in various ways: a local adapter that maps the devices into a HTTP[S] server or an IP adaptation (hardware – a dongle or special gateway) that is functional enough to connect at link level into a switch or router.

There's no need to waste effort on a universal solution. This will emerge if the problems are important enough to require it. Meanwhile, how do we nudge it along?

We need a strategic outcome that will engage manufacturers, service providers, app[lication] developers, certifiers and regulators (who will be important for safety-of-life systems and services), and also appeal to the public who will buy the products – branding of some kind, a kitemark.

So how to get there? There are two tactics: the politics of branding; and taking a lead in making it happen.



The political part first. This is an adaptation of the CENELEC Interoperability Framework for HBES.

It embodies technical step-changes that incumbents would have to concede, even if they didn't like them: no exclusivity – people will buy different systems; no duplication – people have limited tolerance for buying separate disconnected systems and will expect resources to be shared; no hassle – people expect products to "just work" – they may need to authorise the configuration of a new "thing" but it should build itself into the system(s) on its own.

It recognises that this doesn't happen at all but it gives everybody a chance and it reflects reality.

Level 0 systems are very common. They are purpose-built proprietary products that you can buy, e.g. from Maplins or Velux, for a single purpose. The most you can expect is that they don't interfere with each other. You can buy them, plug them in and go. What's not to like? Replication of everything is what's not to like.

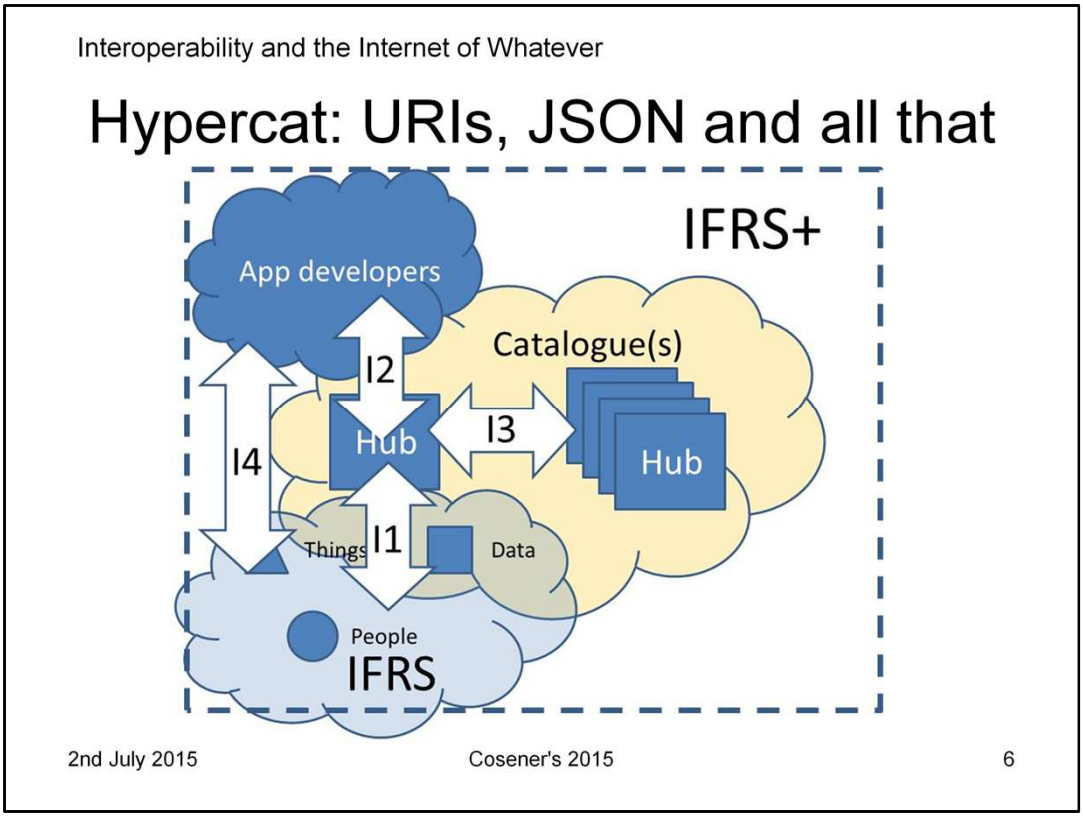
There are many Level 0 packaged products and they may be quite flexible and sophisticated.

Levels 1, 2, and 3 expand this: the products you buy can use what you already have, maybe you just buy a new application; then you can connect to another company's products in a proprietary way, maybe inside your home, often outside to a "value-added" service; and then you can buy and use anything that complies with a standard, which may work or not. However all these opportunities are only realised because manufacturers agree to cooperate and they require specialised equipment, e.g. gateways, and usually an "expert" arrives to install the systems. This is where the innovation is happening now but it is constrained in many ways.

There is a lot of commercial activity at the Levels.

Levels 4, 5 and 6 pose the challenge to future systems. They aim to deliver HBES systems, products and services as a commodity. You buy a pack of thermostats, put them where you like around your house (or site), let them negotiate to join your existing environmental management system, then you decide which of them you will use, … It's under your control at Level 6 in a comprehensible way. You need an installer or specialist at Levels 4 and 5, but the intent of 6 is to eliminate the need for such intervention.

To be useful as a standard, the framework comes with a set of artefacts, the Interoperability Implementation Conformance Statements (IICS). If you remember PICS, PIXITs, and TTCN from the old OSI days, (they are alive and well in ETSI), then you will know that this is extremely tiresome. It is the only way we propose to standards bodies but we look for a practical way to make this automated, usable and useful.



How do we make the CENELEC IICS concept palatable?

Recall the lessons learned: we need something that is sufficient but no more and as simple as possible, allowing things to coexist if that is enough and to work together if they want; and don't make a formal understanding of how people think central to the concept.

We need an architecture, functions, and protocols to make this happen. There are four hurdles: how to name things, how to address/locate them, how to describe them, and how to make them interact with each other. Of course, we have this already: URIs, search engines, HTTP, JSON, web servers and databases.

An information model and a means to store the information and exchange it. This is not novel or a surprise, so we have to see what others have done and if there is an approach that has some take up by developers, IoT enterprises and end-users.

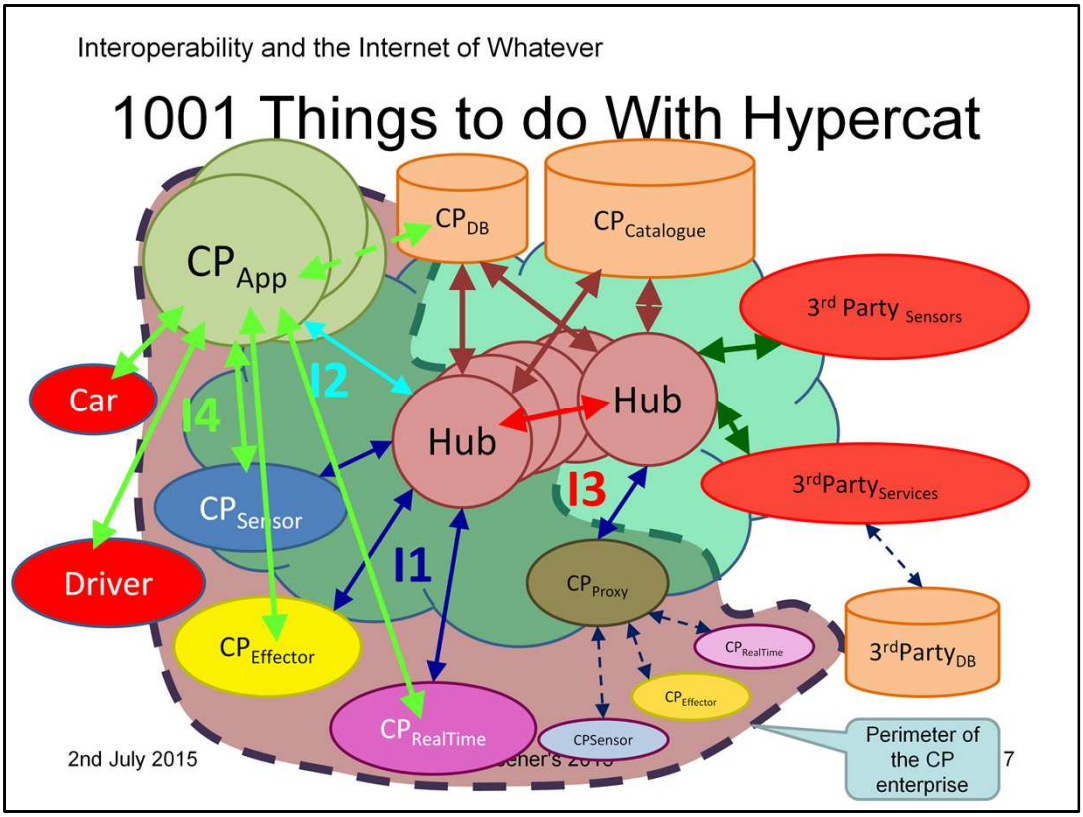
Hypercat provides a starting point. But only a starting point. It defines a catalogue that is accessed through hubs. A loosely defined JSON schema allows you to encode information and metadata about things. You use URIs to address the catalogue and HTTP[S] to access it.

The Hypercat protocol is designed to enable "App developers" to connect with hubs (interface I2) and discover objects. Objects can register themselves in the catalogue (interface I1). Hubs can connect with each other (interface I3).

This provides support for the IFRS discovery phase but not for application semantics supporting configuration, operation and management, where the application interacts with the objects directlly. Therefore we added interface I4 to do this. We do not require any specific protocol to be used over I4: it could be MTTP, COAP or anything appropriate to the individual application.

The Hypercat protocol is not defined sufficiently precisely to enable interoperability even by the limited objective that we have set. However it is trivial although laborious, to encode the IFRS IICS as metadata and use it to extend the information model.

The interaction between hubs over I3 is also unclear. However it is obvious that catalogue access requests could be forwarded or redirected, so a DNS-like approach could be specified.



The picture shows how Hypercat could be used to support a car-parking application (CP).

So what is the car-parking enterprise? It is about real-estate, filling space with cars in an efficient way, collecting payments, offering incentives, and business models. So it is an ideal Internet-of-Things application.

Cars, parking-spaces, payments, people (drivers, CP providers and operators), sensors (number-plate scanners, proximity/presence sensors, secuirty cameras), effectors (entry/exit gates), databases are “things” that participate. Actors and assets of the financial part of the enterprise are not shown.

The perimeter of the application is shown. There will be many things that are outside it: other hubs and catalogues, legacy 3rd party sensors and services. The CP database, the car and the driver are treated as being half-in/half-out. The CP estate is not shown explicitly – it participates passively via the CP sensors and effectors.

Given the number of interfaces, actors, and assets, security and its governance must be a major concern. However time is too short to cover this analysis.

Taking each interface in turn:

I1: sensors and effectors advertise their existence and capabilities via a local hub;

I2: instances of the CP app operated by a CP provider discover sensors and effectors and configure them if appropriate and permitted;

I3: provides a referral and redirection capability between hubs and catalogues – things within the scope of a CP app instance may be widely distributed;

I4: Once the CP app is configured, Hypercat recedes into the background and application protocols take over.

The main message to take away is that the number of hubs and catalogues could be very large. They could be owned by anybody and operated as part of a home or part of an enterprise. They will probably be shared among several applications but could be operated just for the purposes of the CP system. They could be big or small in logical terms.

The I3 protocol should allow referral at least; redirection would also be useful. Thus Hypercat can provide a DNS-like service.

But this is all very obvious…

Thoughts – What Next?

- Political is more important than technical
- Solve problems, don’t create them
- Recognise the diversity and fragmentation
- Hypercat – a Domain Interoperability Service?
- Don’t forget the legacy or the law

Some (Possibly) Useful Links

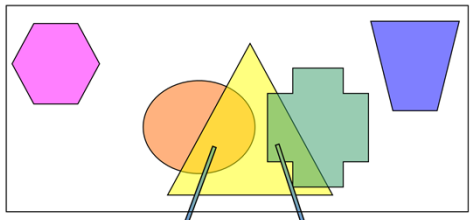
- There are several papers on DNS for IoT – should check them to check this approach for sanity
- Search for “Hypercat”, “CENELEC IFRS” – this will take you directly to useful links.
  - No need to pay
- If anybody knows better, please let me know!

Thank You ?

Backup

Interoperability – 3 Issues

Co-existence

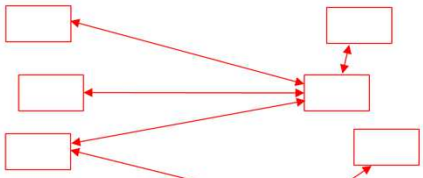


Systems

Interworking

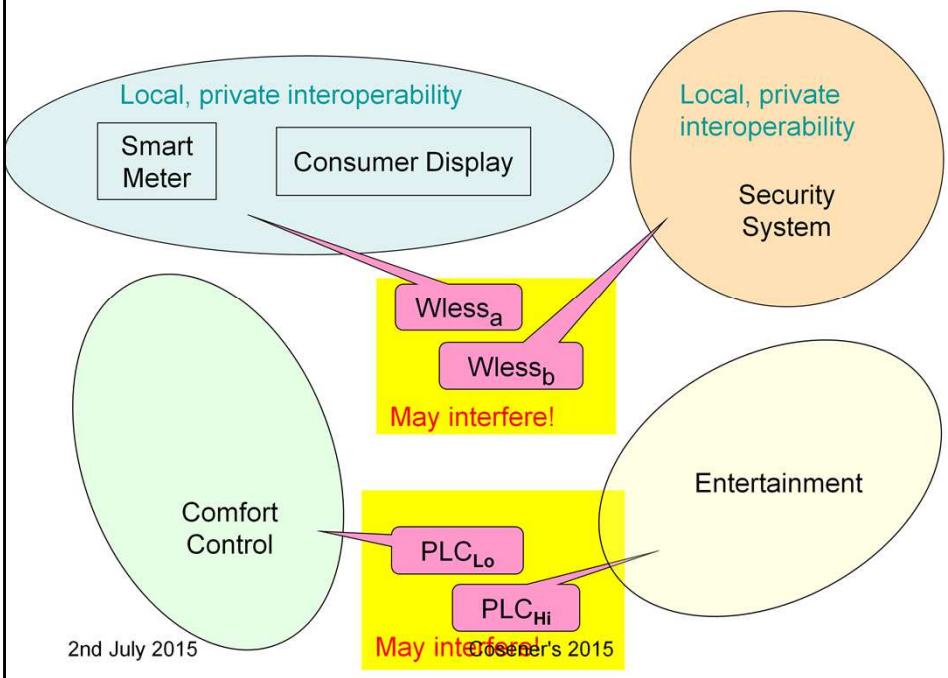
Technologies

Interoperability

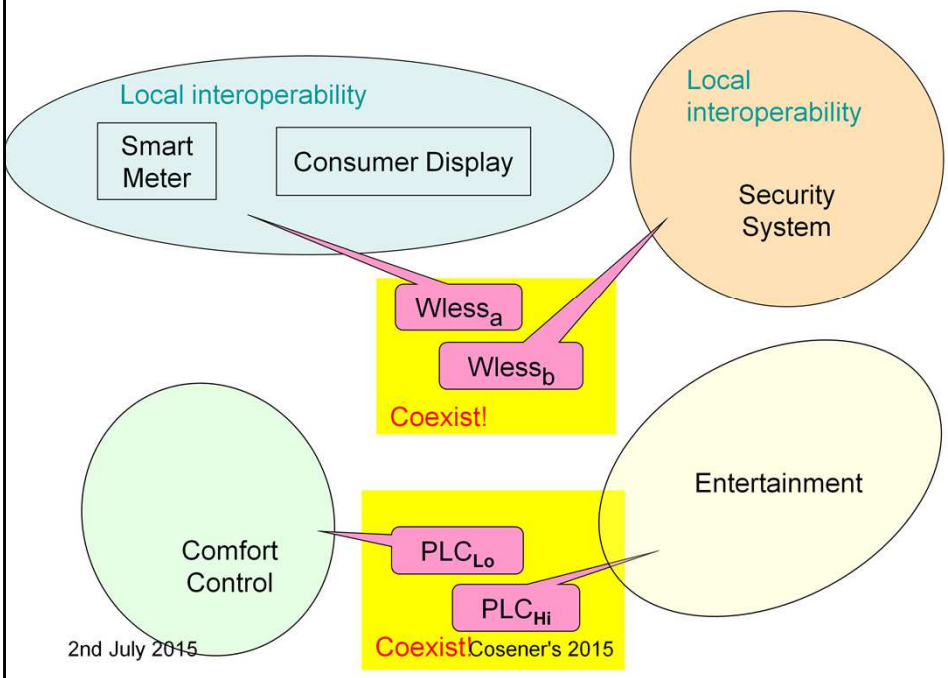


Functions

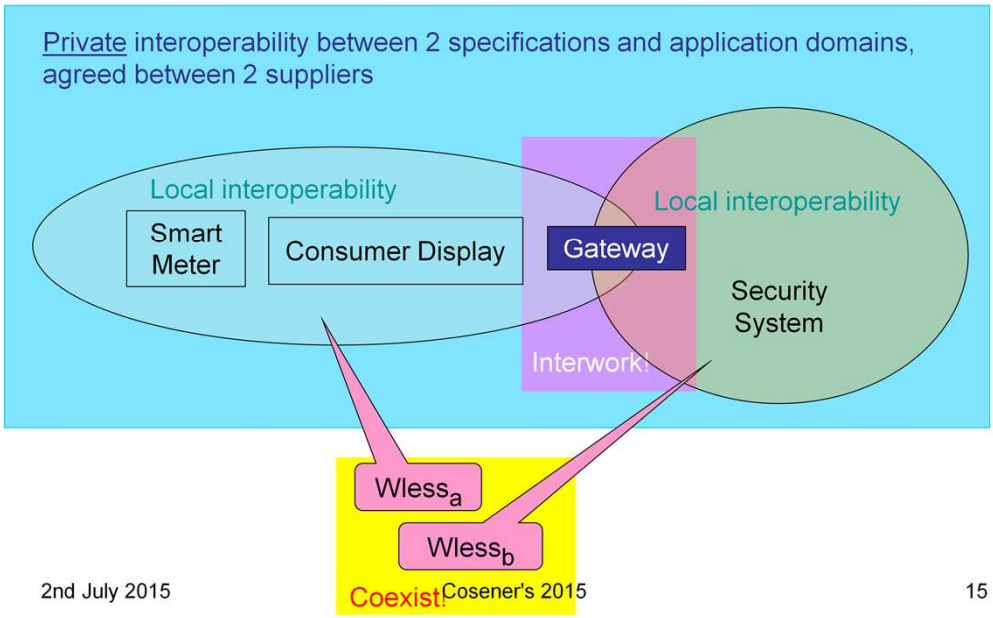
Level 0 – Isolated Systems



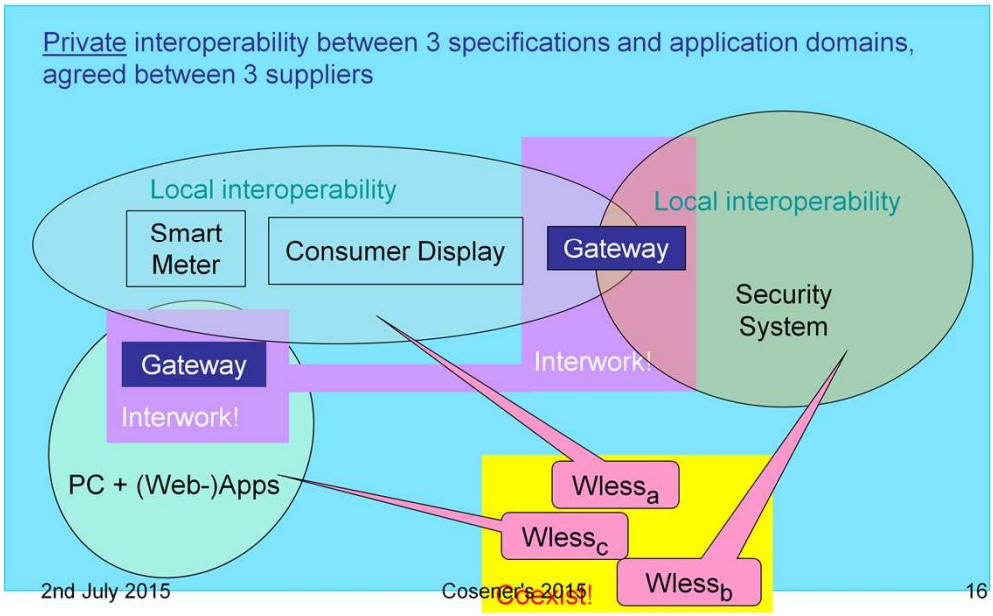
Level 1 – Coexistence



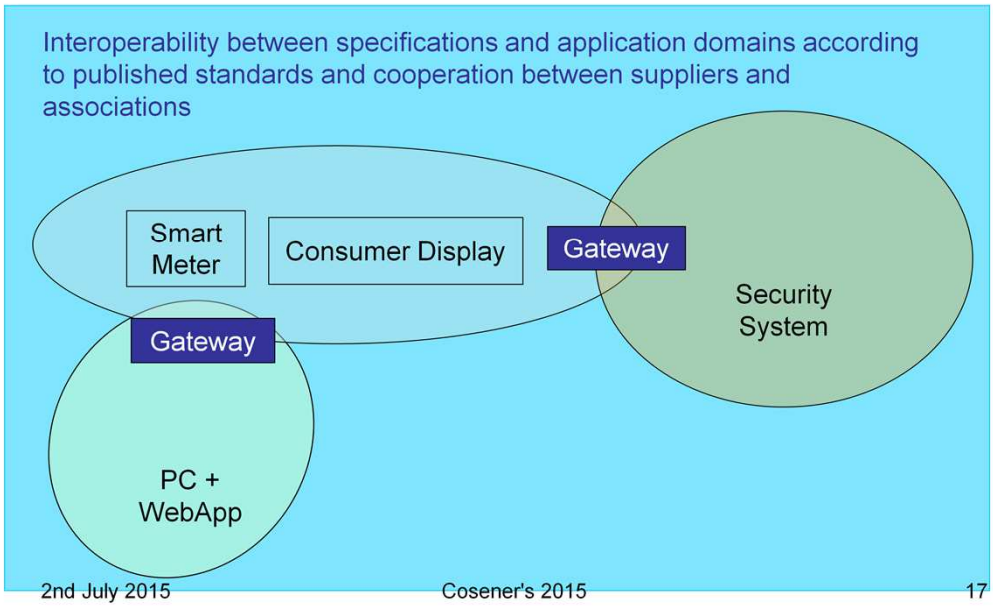
Level 2 – Interworking (1:1)



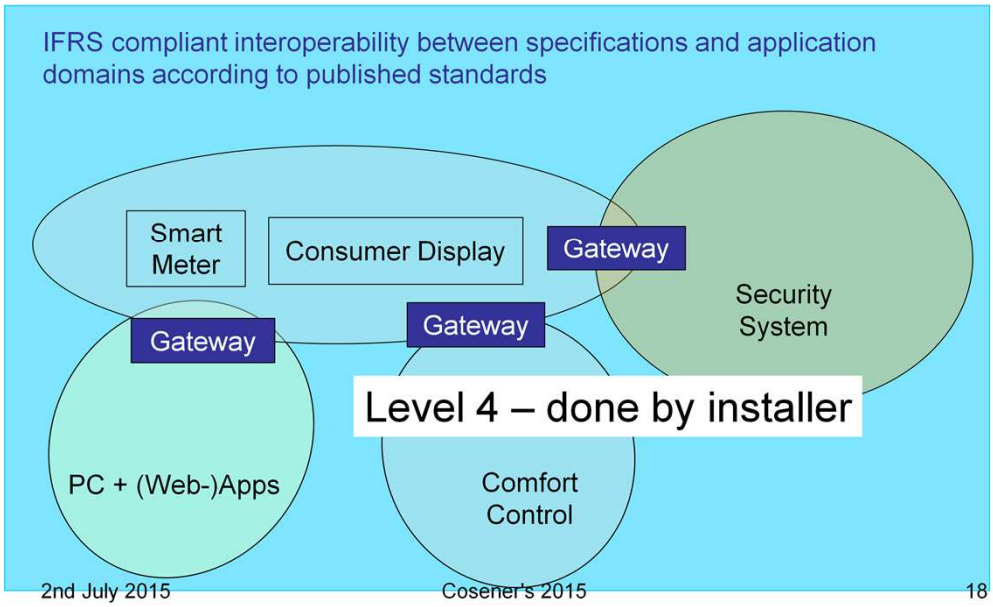
Level 2 – Interworking (> 1:1)



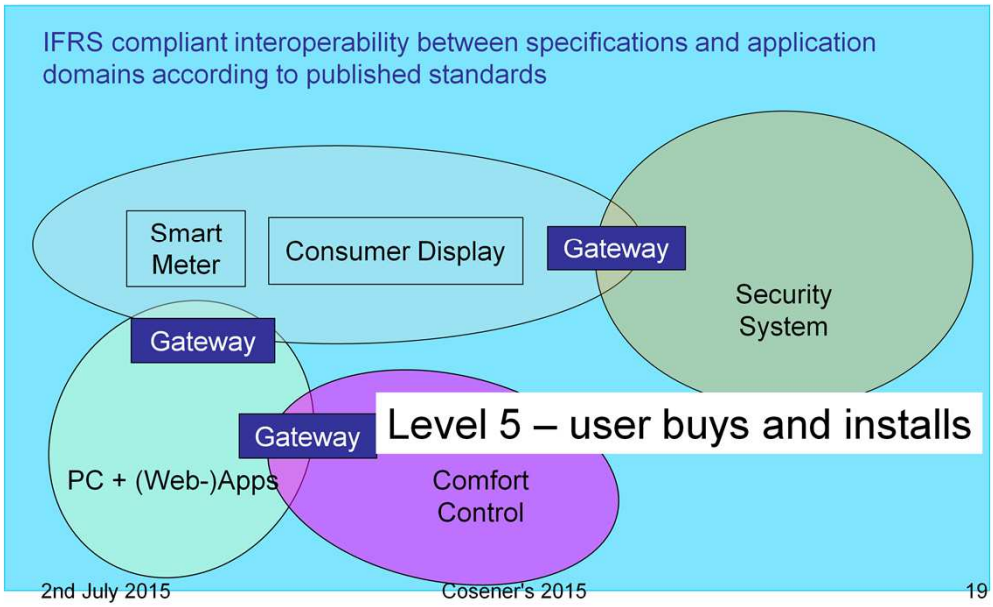
Level 3 – Interoperability



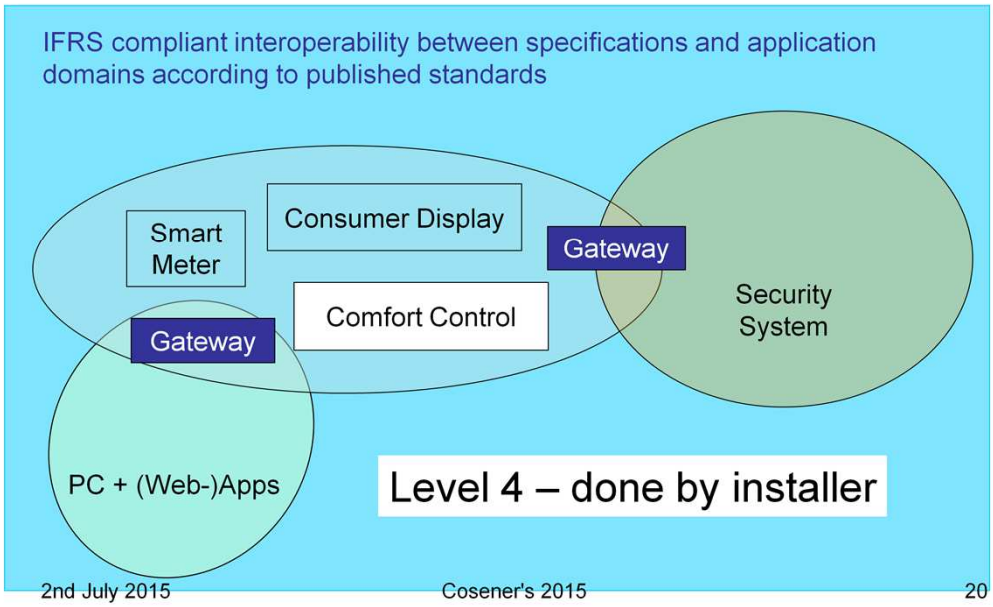
Adding a New System (1)



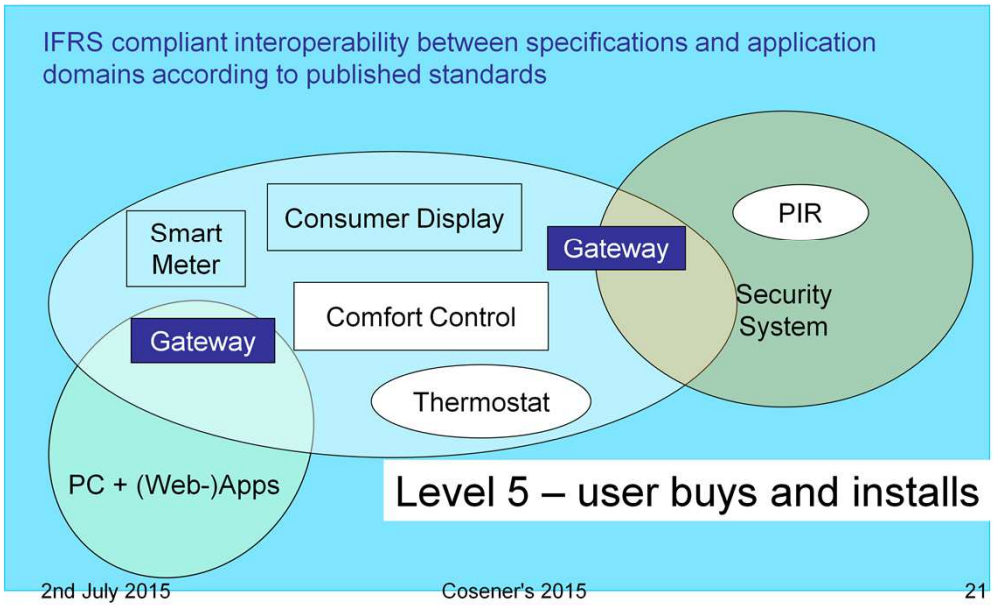
Adding a New System (2)



Adding a New Application



Adding New Devices



Service Provider Access