

HES Product Interoperability

-

Principles and Examples

Dritan Kaleshi
Ron Ambrosio

19 September 2006



Interoperability

- Interoperability
 - The ability of two or more entities to communicate and co-operate despite differences in the implementation language, the execution environment and/or the model abstraction.
- Interoperability, in general, needs to be defined at:
 - Protocol level (the ordering between the exchanged messages and the blocking conditions)
 - Syntactic level (names and signatures of operations; encoding)
 - Semantic level (the “meaning” of the operations; the behaviour of the entities due to the exchanged messages)
- In addition, in our context, the interaction model level
 - In a multi-spec interoperability framework one must ensure that the translation process meets the individual system interaction model assumptions and requirements



Interoperability Specifications - Existing

- Within the realm of a given specification/standard specify application models, device profiles and all the necessary messages (format and content) that manufacturers should comply with to be certified as “spec/standard interoperable”
- Every existing system has well-defined processes and profiles for this: KNX, EchoNet, CEBus, LonWorks, etc.



WG1 Interoperability Project

- How to bridge between semantically similar but syntactically different existing (and possibly future) home electronic systems?
 - Define a generic list of objects *and* operations on them that would allow any HES to realise interworking functions either directly or through a 3rd layer (2-tier system) solution
- Approaches open to WG1 Interop Project
 - Adopt a single system as the agreed-upon universal standard
 - This is not possible due to market pressure and politics, including standardisation politics
 - Or develop system-to-system translations between all target systems
 - Technically not acceptable.
 - Or define a framework and specify the process of translation in multi-system installations without *the need to specify* specific system-to-system protocol translations



Principles of the Interoperability Framework (1/2)

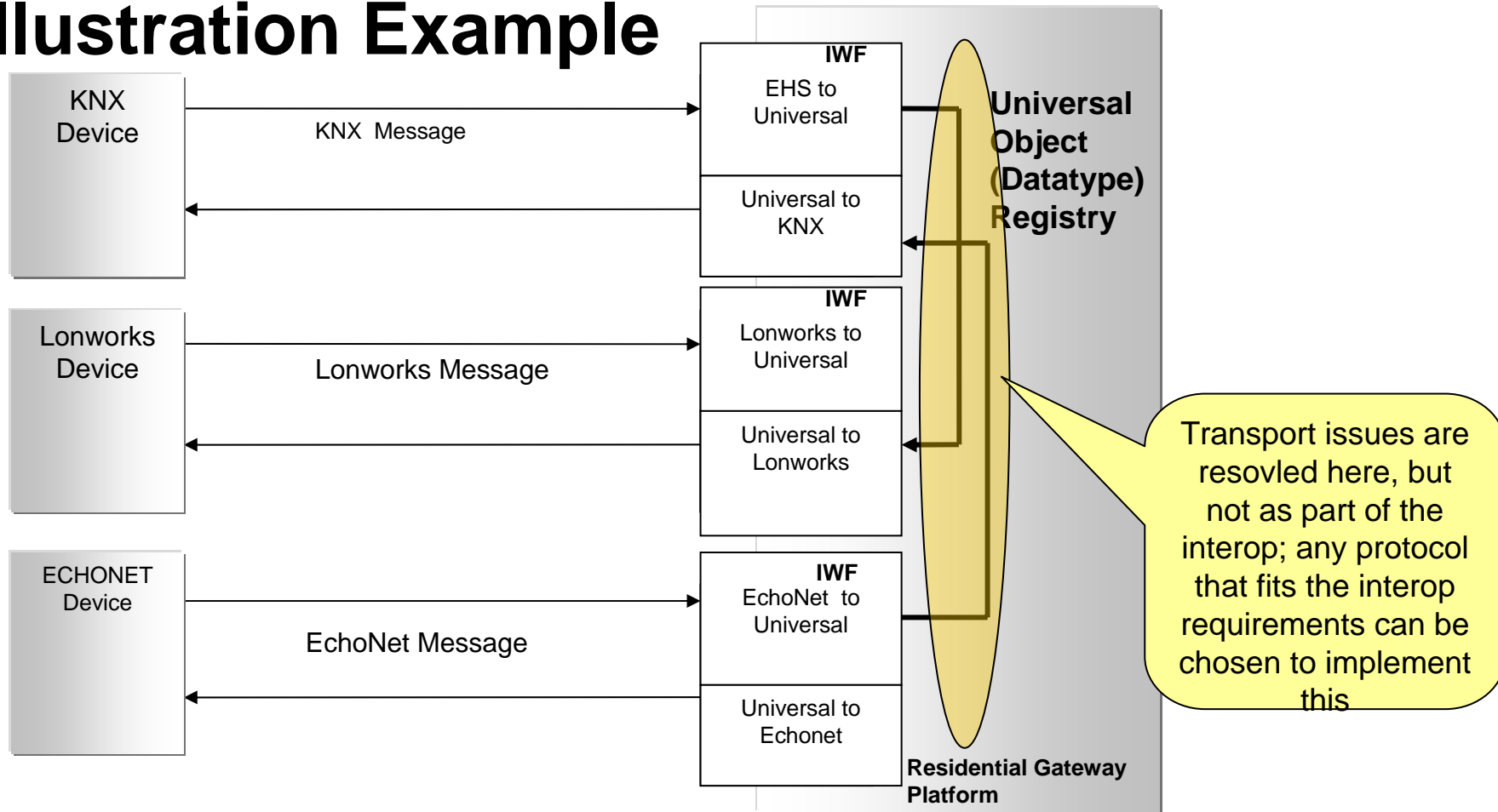
- Interoperability through 1-to-N translation
- Capture functionality, not transport rules
 - Data representation and semantics is more important than the transport syntax and rules
 - The manufacturers know their protocols better!
 - And they are not giving them up – they are trying to standardise them wherever they can!
 - Which means entrenched multiple standards will exist.
- Communication interoperability is given
 - Either in a single gateway device or several gateways (distributed home gateway platform)
 - Note that we have accepted as given the existence of (at least) one gateway device in the home.
 - Hence we can safely assume that communication interoperability is provided for any process running on the gateway(s)



Principles of the Interoperability Framework (2/2)

- Differentiate between product interoperability and application interoperability
 - Product interoperability means ensuring common data representation between different entities (syntax at application data representation)
 - We have many systems specifications but the same scenarios, same application set, same actions and same (expected) behaviour
 - Application Interoperability
 - Built on top of above, but includes further specifications for the interaction model and application operation and management
- Specify requirements for the transport protocol to be used within the interoperability framework
 - The transport protocol will need to be standardised at some point, but not as part of the interoperability project. Several candidates do exist, and this is open for discussion

Illustration Example



Example : HVAC System (1/3)

•HVAC Controller (Sys_A)

- Current Temperature
 - Degrees Celsius
 - Encoding: unsigned char
- Fan Speed
 - Off, Low, Medium, High
 - Encoding: unsigned short {0,1 ,2, 3}
- ...

•Fan (Sys_B)

- Fan Speed
 - Off, Low, Medium, High
 - Encoding: unsigned short {0, 1, 128, 255}
- Fan Status
- ...

•Temperature Sensor (Sys_C)

- Current Temperature
 - Degrees Fahrenheit
 - Encoding: unsigned short integer (2 bytes)
-

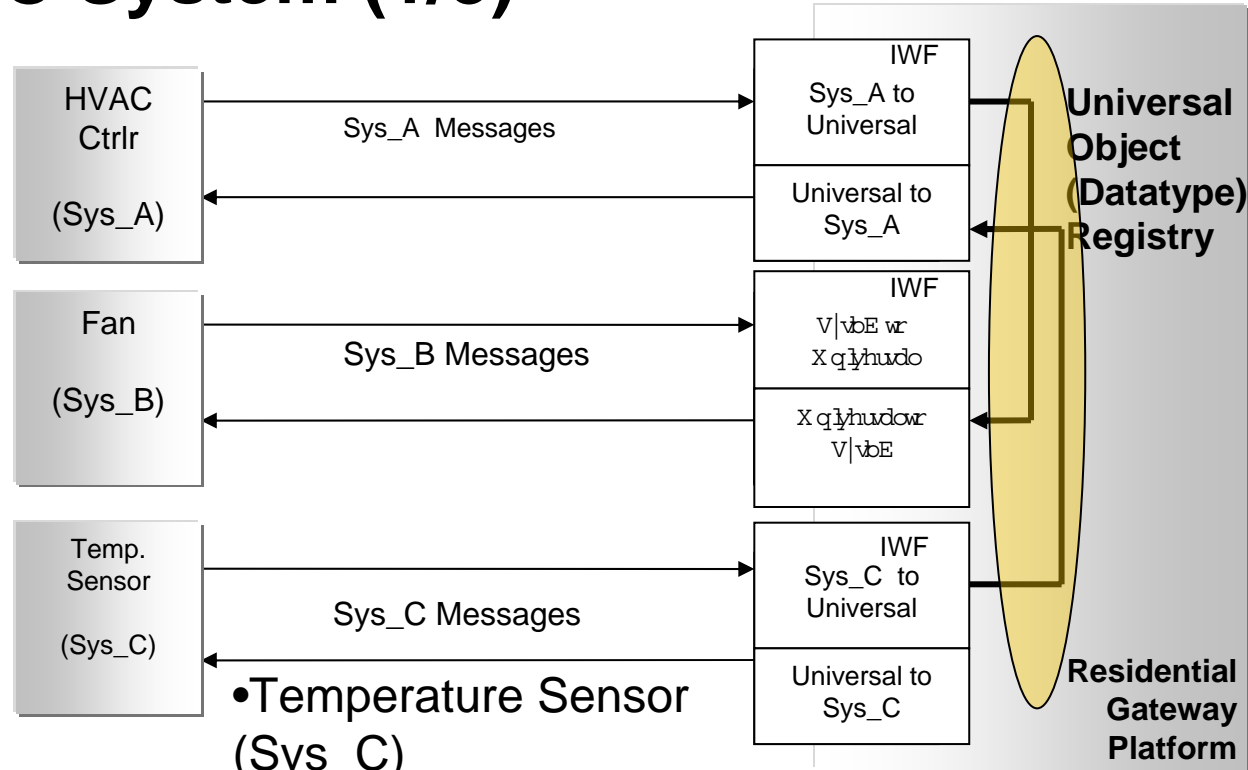
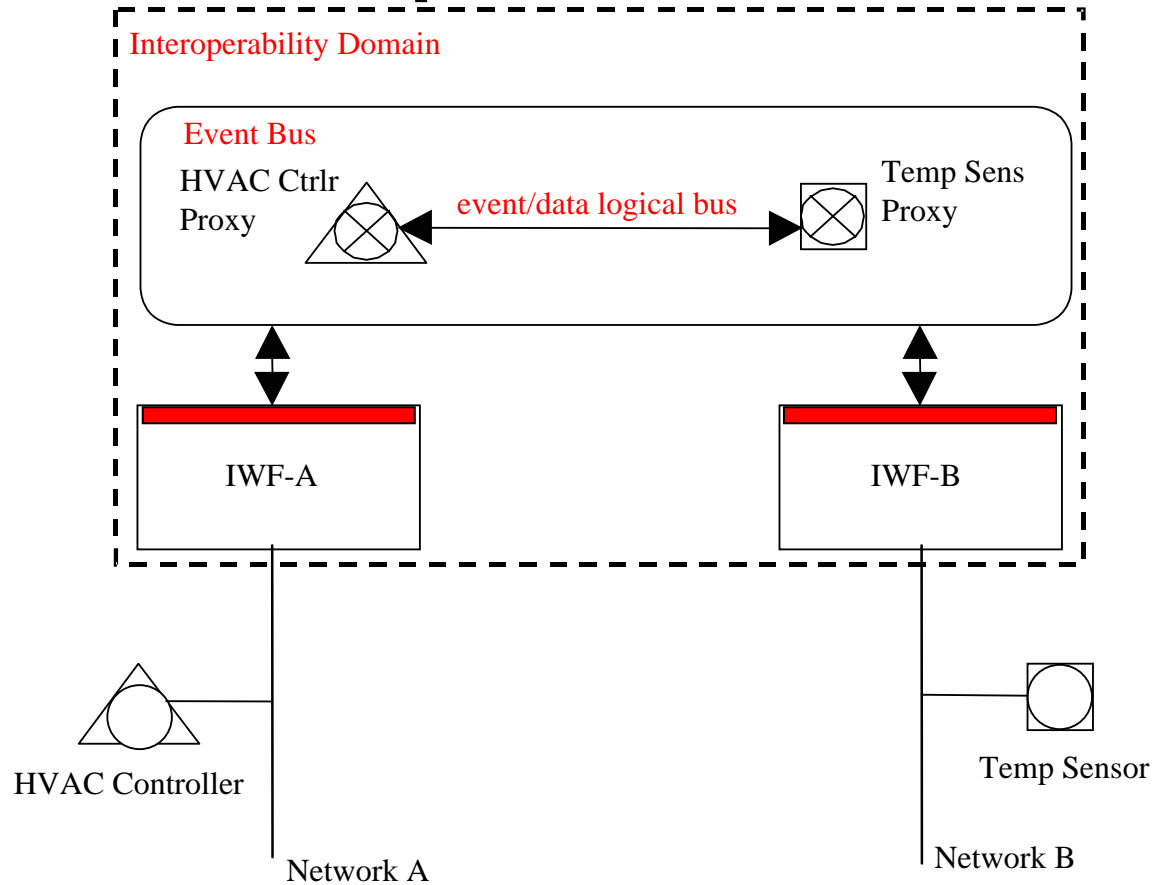
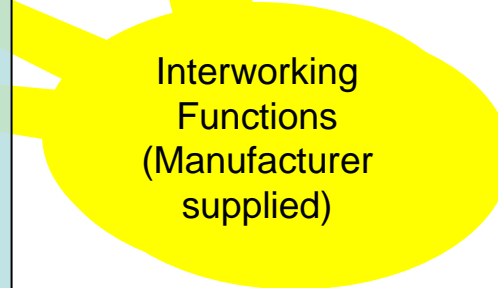
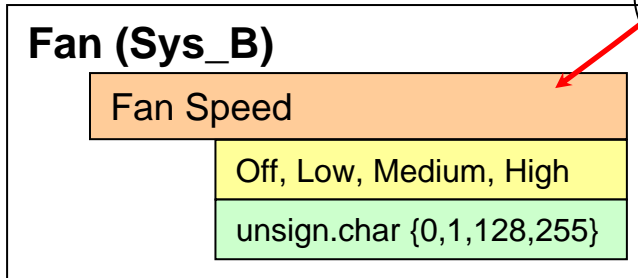
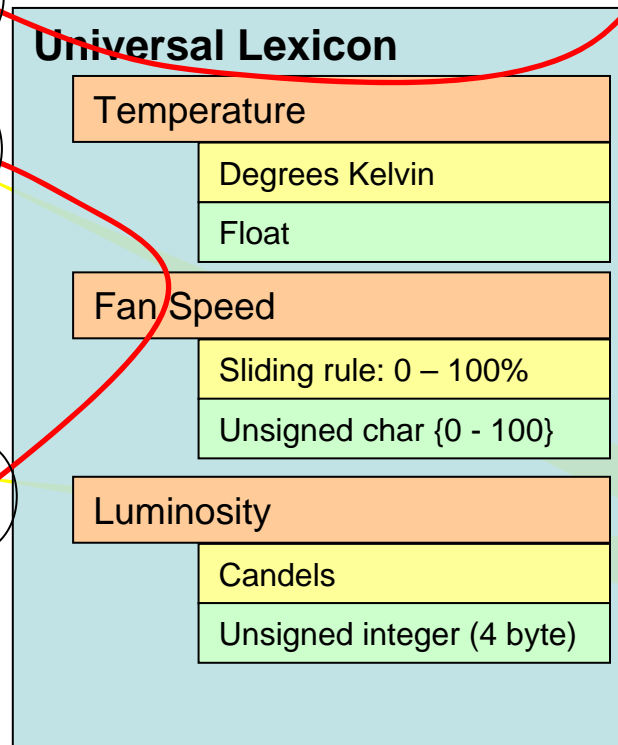
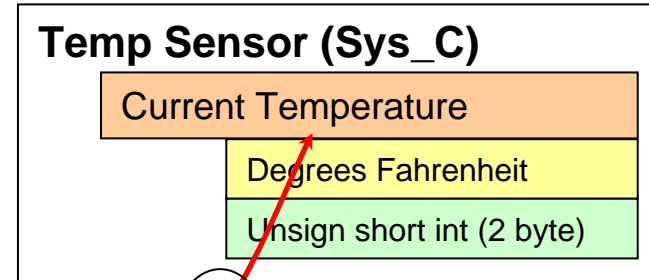
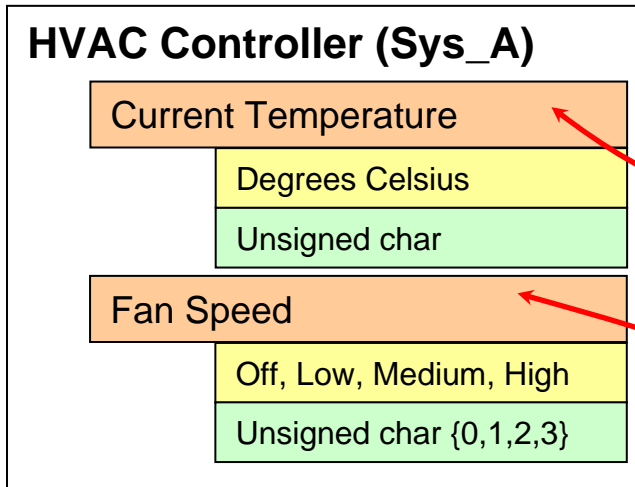


Illustration Example

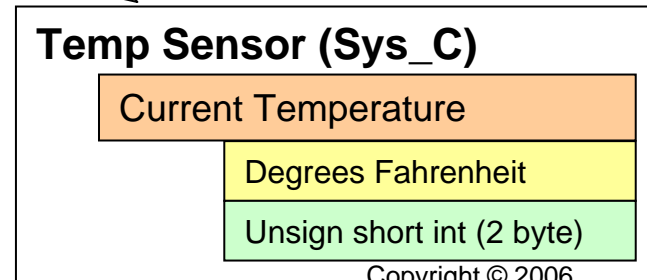
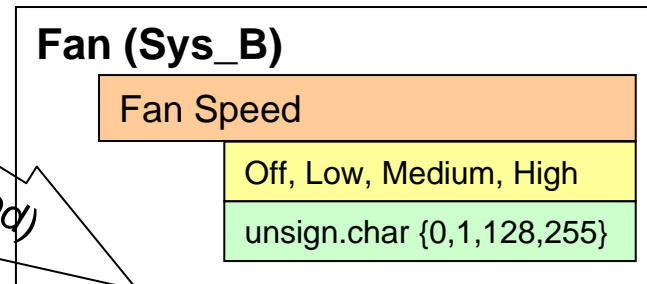
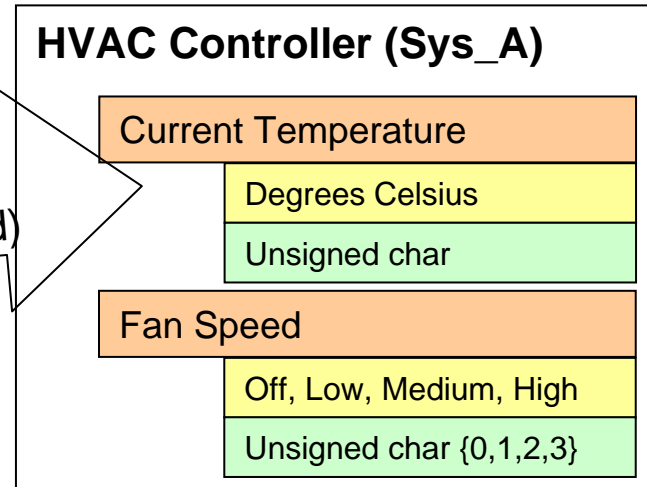
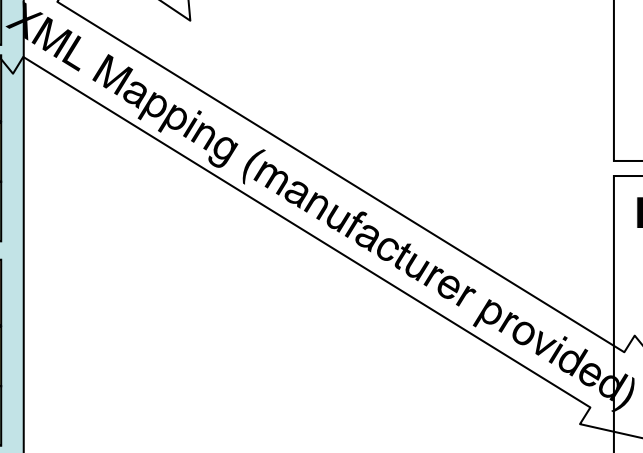
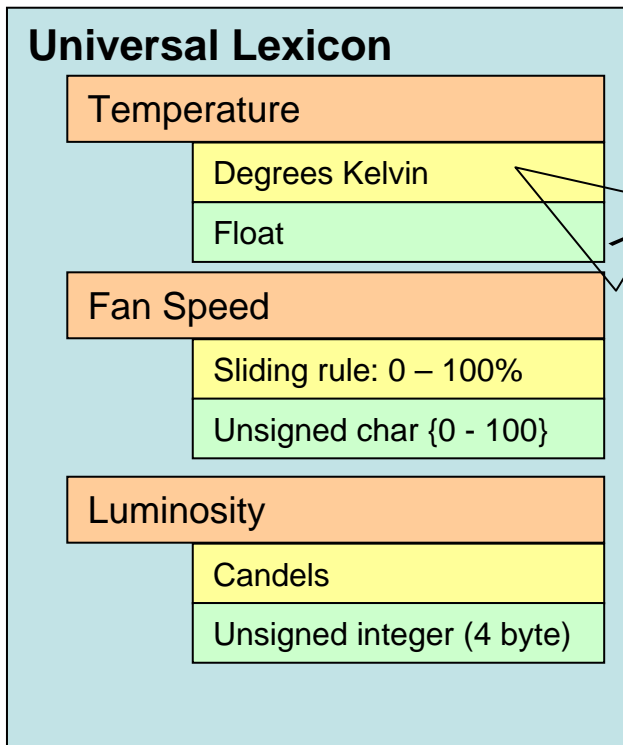


Example : HVAC System (2/3)



Example : HVAC System (3/3)

Manufacturer-provided
Device Descriptions





Demonstrators

- Like all standards/specifications, it should be shown to work
- UoB Interoperability Demonstrator
 - Proof-of-concept of the translation process
 - One possible way to implement it, but not the only one
 - Note that *implementation* is not (and should not be) defined in the standard
 - Application emulator, and uses an event bus implementation as well
 - Application-level operation
 - Application configuration is manual, through configuration files
 - Application emulator, but uses a logical event bus to communicate between the proxy objects
 - Will be improved to use parsing and validation of the XML submitted file



1st pass :: Tag Translation (message parsing)

System_A Generic Message Structure

```
<?xml version="1.0"?>  
  <EHS_command>  
    <service>  
  </service>  
    <object>  
  </object>  
    <data>  
  </data>  
  </EHS_command>
```

Universal Generic Message Structure

```
<?xml version="1.0"?>  
  <universal_command>  
    <action>  
  </action>  
    <property_name>  
  </property_name>  
    <value>  
  </value>  
  </universal_command>
```

System_A to Universal Translation Map (Tags)

```
<?xml version="1.0"?>  
  <EHS_command>  
    universal_command  
    <service>  
      action  
    </service>  
    <object>  
      property_name  
    </object>  
    <data>  
      value  
    </data>  
  </EHS_command>
```

<!--this translates only the tags, there should be a second level translation of content-->

<!--second level (content) translation would require a parser -->

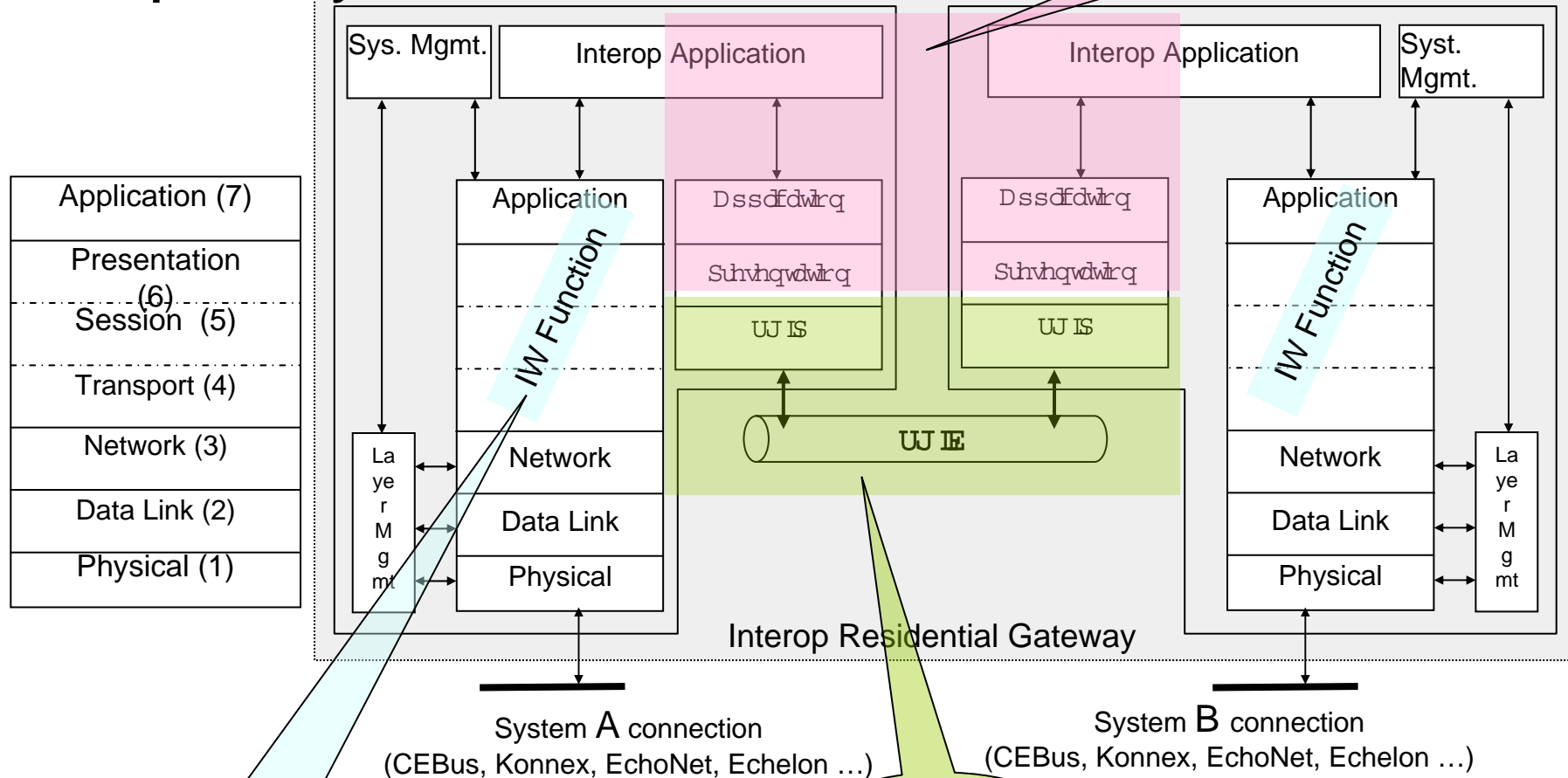
System_A to Universal Content Map

```
<?xml version="1.0"?>
<!--Sys_A temperature sensor-->
<!-- An already to-universal-translated XML document would use-->
<!-- this for a 2nd-level translation of the contents -->
<!--universal to Sys_A-->
<universal_command>
  <action>
    <write>
      write
    </write>
    <read>
      read
    </read>
  </action>
  <property_name>
    <current_temp>
      room_temp
      <codeBase>
        celsius_to_Kelvin
      </codeBase>
    </current_temp>
    <device_name>
      name
    </device_name>
    <date>
      date
    </date>
  </property_name>
  <value></value>
</universal_command>
```

2nd pass
:: data mapping

Interoperability
specification domain

Interoperability Framework



Manufacturer-provided
interworking Function

HomeGate
specification
domain



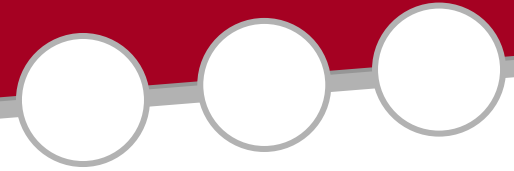
Overall Summary (including WG1 N1236)

- The Interoperability Framework addresses:
 - Product interoperability through the application data mapping/translation in a 1-to-N way
 - Application interoperability
 - Application Models
 - Eventing over an event bus as the lowest common denominator for an interaction model for the existing systems
- The Interoperability Framework can be run on a single or distributed platforms (not only in a gateway, but very likely to be such a device)
- The approach adopted is the best one under the current and future existence of multiple HES standards.



Further work

- Complete the Lexicon in two phases:
 - Establish the lexicon editing process through examples (as shown, but complete)
 - Complete a lexicon established application models/profiles
 - E.g. A/V profile, with control plane interoperability and the possibility of adaptation at streaming level, as/if required.
- Develop full demonstrators (ideally)
 - Requires not insignificant effort – volunteers?
- Complete application management processes
- Any other suggestion????



Thank you...

and any further questions ???