

Interoperable Web Services for Building Automation – Integrating KNX

KNX Scientific Conference 2006

Matthias Neugschwandtner,
Georg Neugschwandtner, Wolfgang Kastner



Automation Systems Group
Institute of Automation
Vienna University of Technology



www.auto.tuwien.ac.at/knx

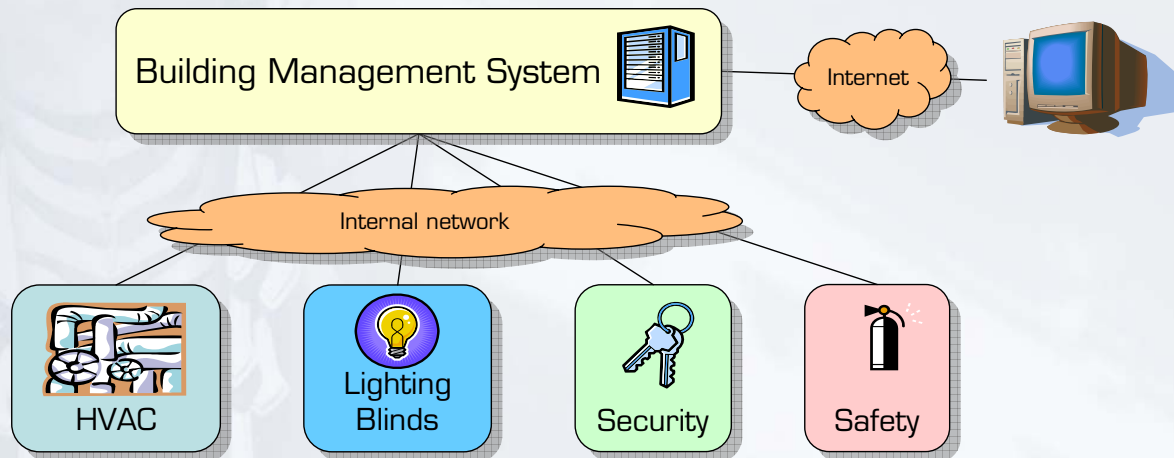
Outline

- Motivation
 - Why use Web Services?
 - Where are they appropriate?
- Relevant standards
 - BACnet/WS vs. oBIX
- Exposing a KNX system via oBIX
 - Mapping of data model and services
 - Discovery
- Prototype implementation



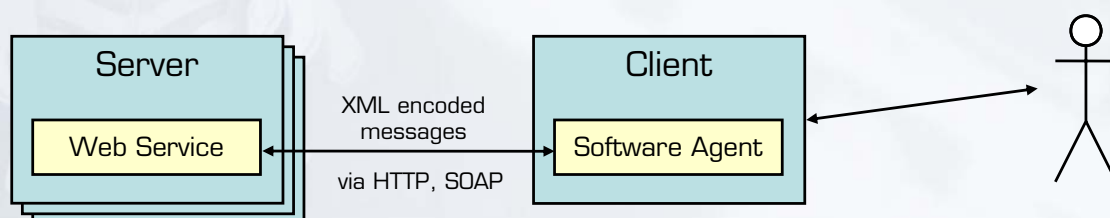
Task

- Integrate datapoints of subsystems at the management level
- Possibly via an internal network



Web Services

- Integration challenges
 - Open standard for exposing subsystem data
 - Today: OPC popular – but only for Windows
- New technology: Web Services
 - Machine to machine communication
 - XML-messages over plain HTTP / SOAP
 - Total platform independence



Web Services

- Modular
 - Off-the-shelf standards for transmission, eventing, discovery, security, ...
- Support service oriented architectures
 - Self-contained, loose coupling
 - Fine grained services can be flexibly arranged into complex applications (Orchestration)
 - Today: Interoperable access to datapoints
 - Tomorrow: High-level business services (load management, ...)
- Drawback: additional overhead
 - XML encoding, no server push
 - But less an issue at the management level

Web Services in building automation

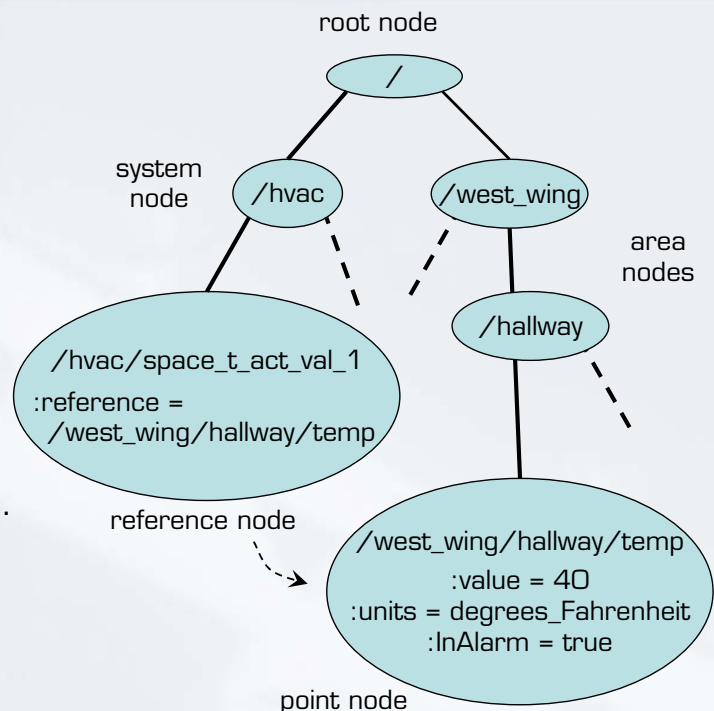
- The established standard: OPC
 - Initially (and still widely)
 - Based on OLE/DCOM
 - Multiple separate servers: Data Access, Historical Data Access, Alarms and Events, ...
 - Moving towards
 - XML/Web Services
 - Unified Architecture (OPC UA)
 - Only XML DA already available to the public
- The challengers: BACnet/WS and oBIX
 - Specialized on building automation
 - Already include histories, alarms, ...
 - Drafts freely available from ASHRAE and OASIS oBIX TC

BACnet/WS data model

- Not limited to BACnet as underlying network
- Nodes
 - Arranged in tree structure
 - Hold data in attributes
- Attributes
 - Primitive value types: Boolean, Integer, Real, String, ...
 - Enumerations and arrays
 - Can be localized
- Services
 - Operate on attributes
 - Retrieval and manipulation of primitive values
 - Retrieval of entire arrays, value history, locale information

BACnet/WS data model

- Naming via paths
 - According to tree structure
 - URL-styled:
/.../boiler/temp:value
- Node types
 - Required attribute
 - Describe semantics of node or sub-tree
 - HVAC system, Point, ...
 - Determine mandatory attributes
 - e.g., "value" for Points



oBIX data model

- Full-blown object oriented data model
 - Much like object-oriented software
 - Highly extensible: inheritance, custom classes, ...
- Everything is an Object
 - Including classes and even method signatures!
 - Just objects composed of other objects
- Standard library
 - Base object types (classes)
 - Including primitive values (Boolean, Integer, ...)
 - Special purpose classes
 - Server functionality: Watch, history, batch operations, ...

oBIX data model

- RESTful approach
 - Resource centric architectural style for WS
 - Highly restricted set of operations
 - Resources share a uniform interface
 - Mimics how the Web works
 - only GET, PUT, POST – but countless kinds of pages
- oBIX services
 - Only three network request types
 - invoke (operation), read/write (any other object)
 - Still, any concept can be described
 - Custom operations, just like any other object

oBIX data model: Object naming

- Name
 - Identifies a (sub-)object within its encompassing object
 - Internal, programmatic identifier
 - E.g., for overriding inherited sub-objects
- URI reference
 - Identifies an object globally
 - E.g., for referring to the class of an object
 - For exposing objects for access from outside (client)
 - No higher-level semantics associated with URI namespace
 - Unlike BACnet/WS tree structure!

Feature comparison

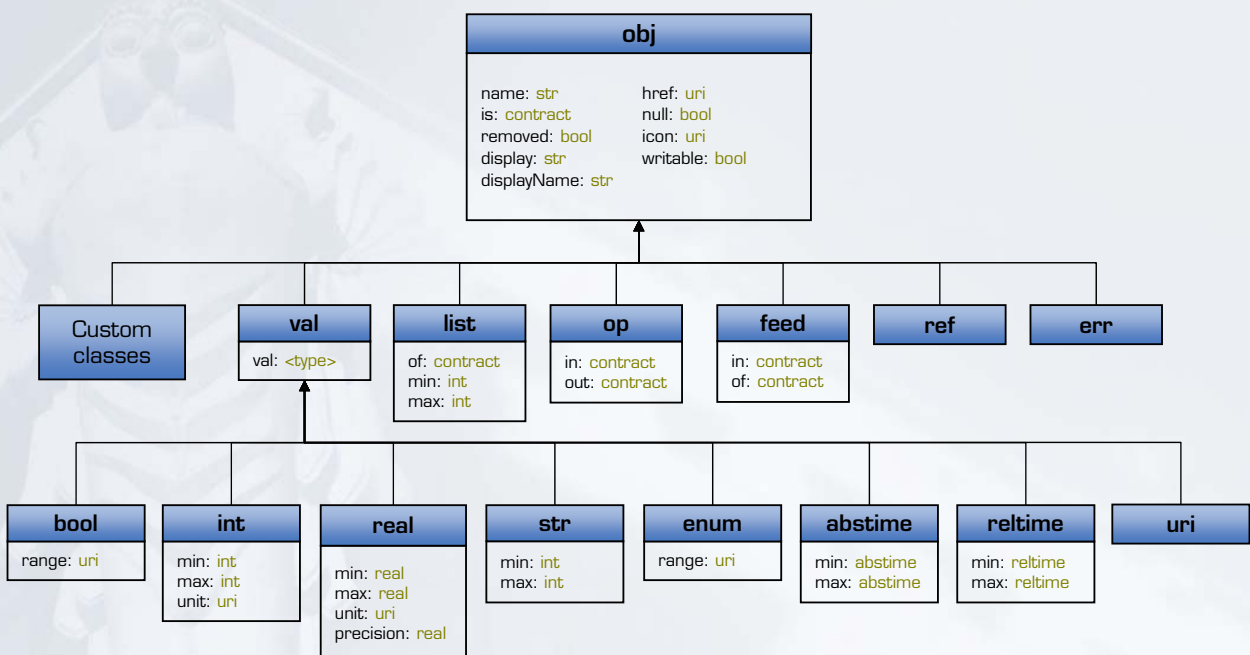
	BACnet/WS	oBIX
Extensibility	Static data model	Open, extensible data model
Data point representation	Node with value attribute, "Normalized points"	Any object, semantic marker for points
Services / operations	Predefined: attribute access	Individually definable
Unit system	Predefined collection of strings	Combination of SI base units
Protocol bindings	SOAP	SOAP and plain HTTP
Localization	Support of multiple locales simultaneously	Left to the HTTP binding
Security	Standard Web Service security (HTTP over SSL/TLS, WS-Security)	In addition: permission based degradation

Mapping KNX to oBIX: Goals

- Provide a Web Service for datapoint access
- Allow plain oBIX client to
 - Monitor and influence the process
 - Set device parameters
- Mapping of
 - Datapoint types
 - Services for process and management data
- Leverage native oBIX language element semantics
 - Event feed for push-style communication, ...
- Allow discovery



oBIX base object types



oBIX and XML

- Object model representation in XML
 - Base object types: <int/>, <str/>, <op/>, ...
 - "Class" specified as individual XML elements
 - Other objects: <obj/>
 - "Class" specified in the "is" XML attribute ("contract")
 - Attributes: XML attributes ("facets") and sub-objects
 - Methods: Network request types and sub-objects
- Contracts: Template objects
 - Inheritance: Sub-objects of the contract are present in the derived object
 - Contracts can be empty (only describe semantics)
 - Objects can fulfill multiple contracts

oBIX XML example

Contract

```
<obj href="def:furnace">
  <bool name="burnerOn" />
  <real name="curTemp" is="obix:Point" />
  <real name="setTemp" val="50.0"
    is="obix:WriteablePoint" />
</obj>
```

Object

```
<obj name="furnace" href="myhouse/heating/furnace"
  is="def:furnace">
  <bool name="burnerOn" val="true" />
  <real name="curTemp" val="45.3" />
  <real name="setTemp" val="50.0" />
</obj>
```

oBIX: predefined classes

- Lobby
 - Well known entry point, watch service, batch operation
- Points
 - Read only: marker contract for base object types
 - Read/write: contract adds write operation
- Historical trends
 - History record: time stamped point value
 - History object: records and query methods
 - Filters, rollup calculation (e.g. average)
- Alarms
 - Normalized model to query, watch and acknowledge alarms
 - Support for stateful alarms (e.g. boiler temperature)

KNX interworking: necessary facts

- Functional block
 - Part of a device
 - Consists of datapoints (and behavioural specification)
- Datapoints
 - Group objects
 - Interface object properties
- Group communication
 - Based on group addressing
 - Push-style (spontaneous) vs. pull-style (request based)
- Datapoint types
 - Data type: format and encoding
 - Dimension: range and unit

Mapping: datapoint types

- Data type mapped to
 - Value object types
 - If necessary, with object containment
- Dimension mapped to
 - Facets: min/max, displayName
 - Range and unit objects

DPT_B1 (Boolean)	<bool/>
DPT_B1U3 (Control_Dimming)	<obj> <bool/> <int max="7"/> </obj>

Mapping: example DPT contracts

Data type

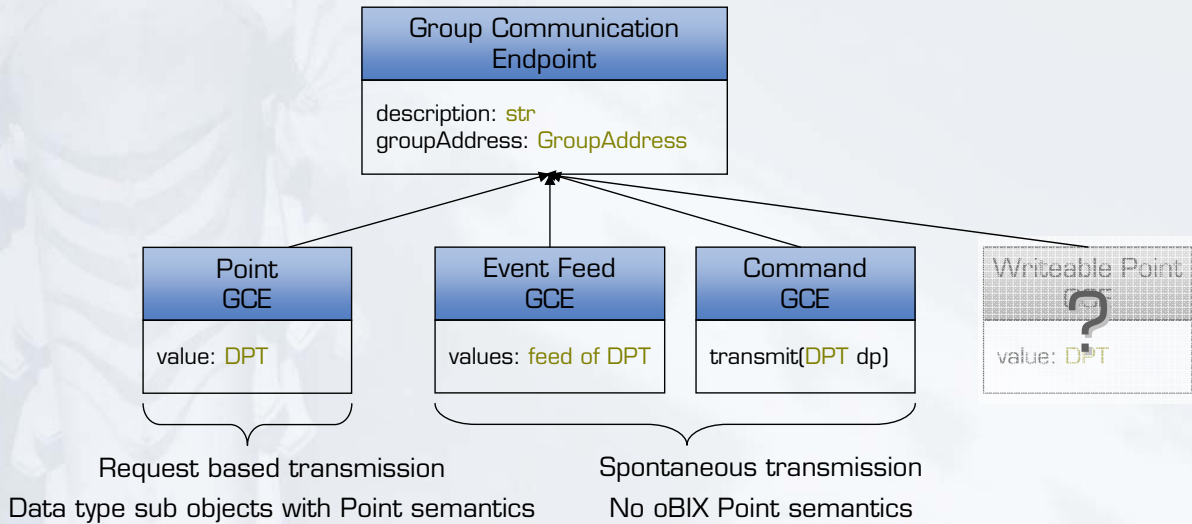
```
<obj href="knx:DPT_B1U3" is="knx:DPT">  
  <bool name="B1" val="false"/>  
  <int name="U3" min="0" max="7" val="0"/>  
</obj>
```

Dimension

```
<obj href="knx:DPT_Control_Dimming" is="knx:DPT_B1U3">  
  <bool name="B1" displayName="brightness"  
    range="knx:range/incDec" />  
  <int name="U3" displayName="stepcode" />  
</obj>  
<list href="knx:range/incDec" is="obix:Range">  
  <obj name="true" displayName="increase" />  
  <obj name="false" displayName="decrease" />  
</list>
```

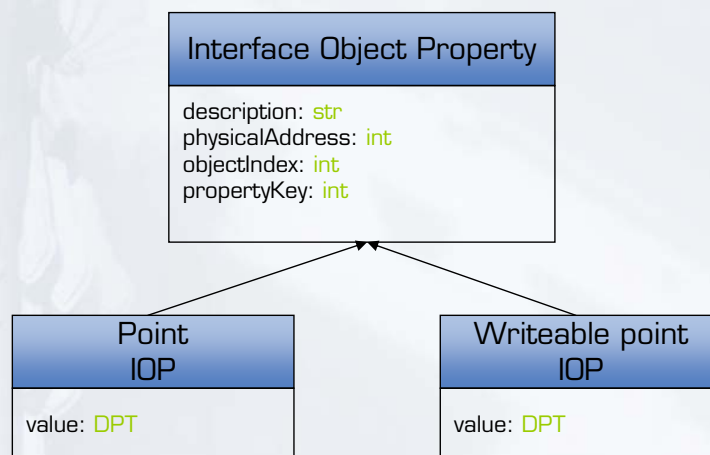
Mapping: process data

- KNX process communication:
via group objects and associated group addresses
- Object hierarchy based on group object interaction type



Mapping: management data

- KNX management communication:
via interface object properties and physical addressing
- Straightforward mapping: oBIX Point container objects



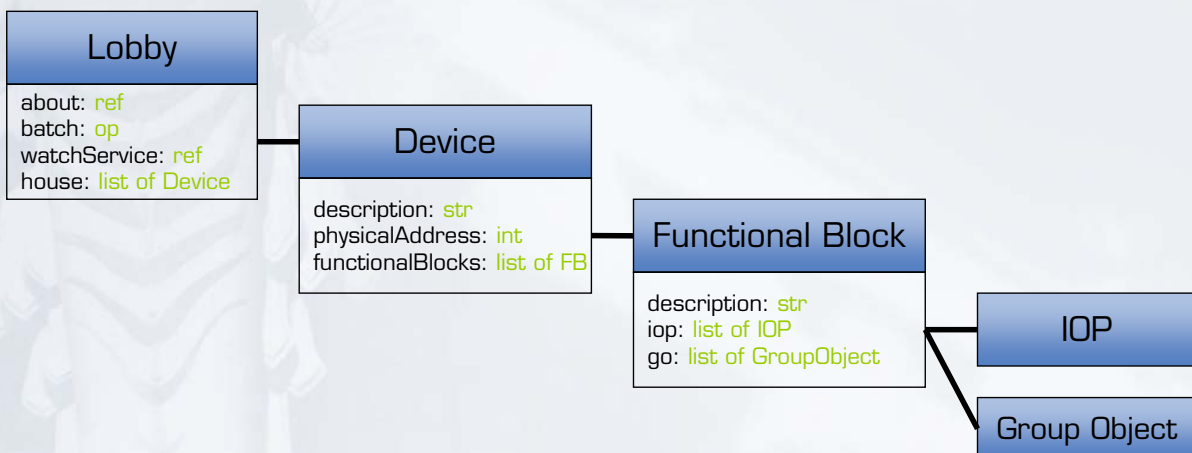
Discovery

- Which GCEs/IOPs are available on the server?
- Not “built in” as in BACnet/WS
- Group by ... ?
- Two complementary approaches
 - Device centric
 - Group address centric



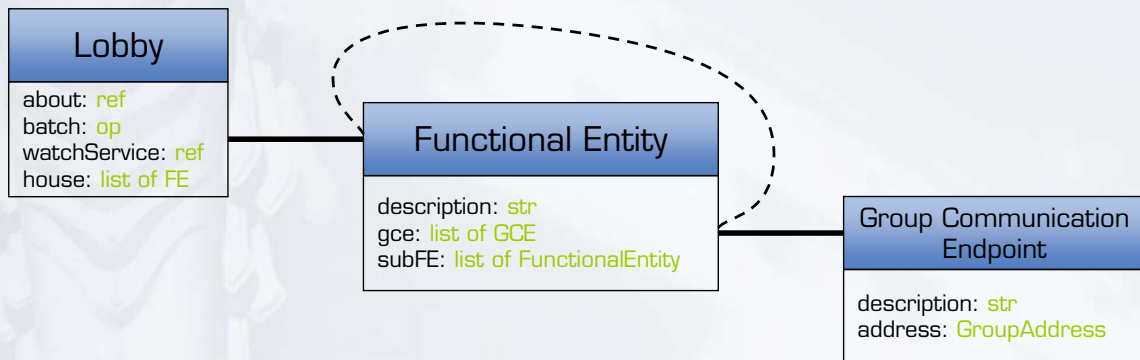
Discovery – device centric

- Structure based on devices and functional blocks
- Access parameters & diagnostic data via IOPs:
“Management view”



Discovery – group address centric

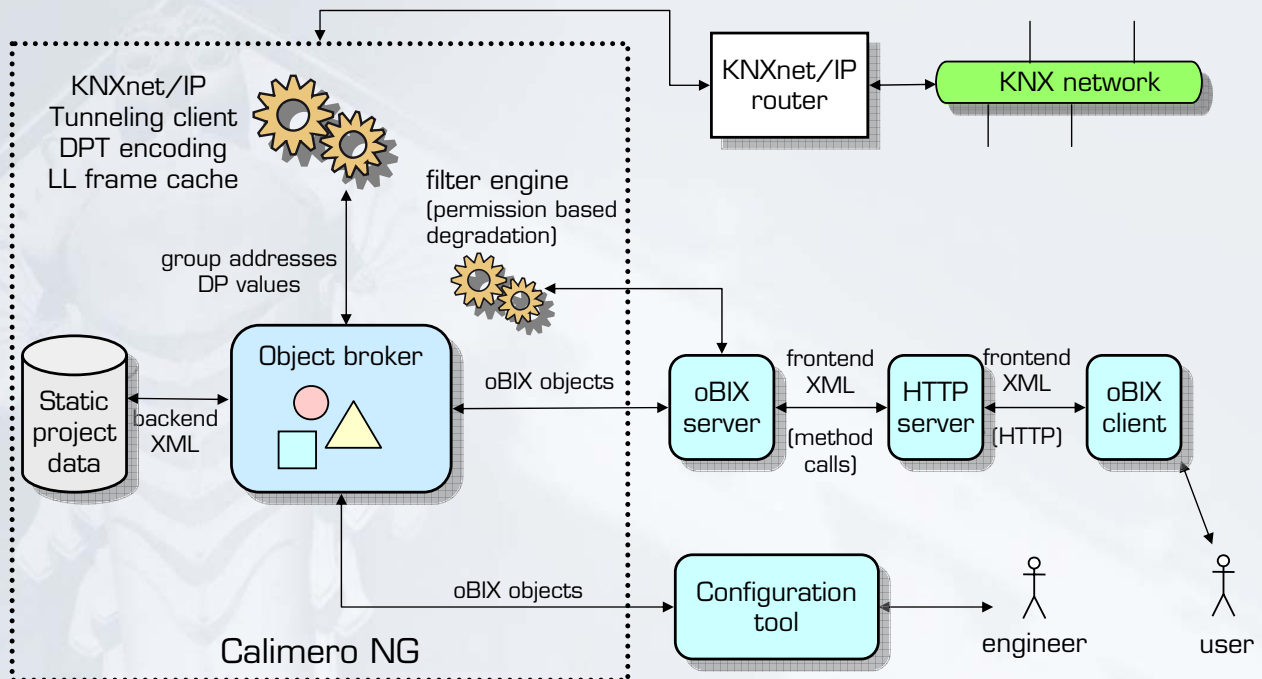
- Focus on group communication
- Structure based on functional entities
 - Directory object, groups GCEs by location or purpose
- Well suited to access process data: “Process view”



Implementation

- Basic process view
 - Group communication only
 - Most popular DPTs
- Plain HTTP binding
- Basic oBIX object access
 - No watches, alarms, batching, history yet
- Configuration data supplied manually
- Leverage available open source software
 - oBIX toolkit on SourceForge
 - Calimero for network access and DPT transcoding

Server structure



Outlook

- Configuration import from ETS
 - XML export format would allow straightforward conversion
 - Functional blocks?
- Keep an eye on related standards
 - OPC UA, BACnet/WS
- Develop higher abstractions
 - “Enterprise-level” Web Services
 - oBIX V2 abstractions