Haystack 4.0 Design Review

Brian Frank

2019 Haystack Connect
Background

• WG 551 since Oct 2017
• Collaboration with ASHRAE AP-WG and Brick
• Iterated several major prototype designs
• Public review
• https://project-haystack.dev
• Design is fluid
Design Scope

- Maintain fundamental design: entities are Dicts, flat map of name/value pairs
- Allow breaking tag changes if required, but avoid it
- Focus on the *definition* side of tag names
- Instance data vs definition data
Problem Scope

• Machine readable ontology to drive docs and tools
• Which tags are used on a given type?
• What sub-components/points should be used on a given equip?
• Improve query abstractions
• RDF as first class export
Symbolic Def Design
Symbols

• We introduce new first class Symbol scalar kind
• Just like Ref, but **not** opaque
• Prefixed with ^ instead of @
• Refs: vendor specific namespace for instances
• Symbols: public namespace for definitions
• Dual identity: can have both Ref and Symbol id
Symbol Types

- **Tag**: \(^\text{site}, ^\text{equip}, ^\text{ahu}\)
- **Conjunct**: \(^\text{elec-meter}, ^\text{hot-water}\)
- **Feature key**: \(^\text{lib:ph}, ^\text{filetype:zinc}\)
Anatomy of an Instance

- *Entities* are modeled as Dict
- Dict is a hashmap of tags (name/value pairs)
- The *id* tag is primary key with *Ref* value
- Entity relationships are tags with a *Ref* value

```
id:@s, dis:"Bldg", site
id:@e, dis:"AHU-1", equip, ahu, siteRef:@s
```
Anatomy of a Def

- *Defs* are modeled as Dict
- Dict is a hashmap of tags (name/value pairs)
- The *def* tag is primary key with *Symbol* value
- Def relationships are tags with a *Symbol* value

```
def:^air, is:^gas
def:^duct, is:^conduit, conveys:^air
```
Def Examples

def: ^number
is: ^scalar
doc: "Floating point number annotated with an optional unit"

---
def: ^equip
is: ^entity
mandatory
doc: "Equipment asset"
Defas as Data

• Defs are normal Haystack data
• Each Def is a Dict
• Flatten to Grid
• Symbolic references form a graph
• Defined in Trio (YAML)
•Encoded using any format (Zinc, JSON, CSV, etc)
Libs - Modularity

- Defs are always declared within a lib
- Libs are packaged as a zip file of Trio files
- Special lib/lib.trio file declares lib meta
- Library meta is a feature key def: ^lib:foo
- Project Haystack defines three standard libs: ph, phScience, phIoT
def: ^lib:phIoT

doc: "Project Haystack defs for Internet of Things"

version: "3.9.4"

baseUri: `https://project-haystack.org/def/phIoT/`

includes: [^lib:ph, ^lib:phScience]
Namespace

- Namespace: hashmap of symbols to defs in scope
- Defined by a list of libs and their symbolic defs
- Lib namespace is based on includes (and itself)
- Project namespace is vendor specific
- Works like import/using/include in Java/C#/C
- Always use simple, unqualified names in our defs and as tag names
Subtyping

• Taxonomy organization
• Subtype is specialization of a more general term
• Set theory: A is a subtype of B if all instances of A are instances of B
• Subtyping is transitive: if A is a subtype of B and B is a subtype of C, then A is a subtype of C (tree)
• Inverse of a subtype is called a supertype
Subtyping Usage

• Defs declare one or more supertypes via "is" tag
• Root defs: marker, val, feature, aspect
• Conjuncts must have explicit "is" tag
• Feature keys implicitly subtype their feature
• Kinds and values now use subtyping:
  
  \[ \text{def: } ^\text{area}, \text{ is: } ^\text{number} \]
Inheritance

• Mechanism of reuse through subtyping
• Inherit each tag from your supertypes if not declared locally (recursively processed)
• Can mark def tags notInherited
• Declared vs normalized
Normalization

- Effective representation a def as dict
- Compiler: declared dicts \(\rightarrow\) normalized dicts
- Tags inherited from supertypes
- Def extensions: defx: ^foo, addMe (late binding)
- The lib tag for def's library (never declared)
- Implicit supertype for feature keys
- Docs and exports are normalized representation
Aspects

- Ontology organization
- Def tag with symbol value (symbolic relationship)
- Design not complete
- TagOn/Tags
- MixinOn/Mixins
- Choices
- Misc: contains, quantityOn, equipFunctions
TagOn

- Associates value tags with an entity marker
- Computed inverse is tags (never declared)

def: ^area
is: ^number
tagOn: ^space

----
defx: tz
tagOn: ^point
MixInOn

- Models an optional type extension
- Secondary dimension from subtype tree
- Tools would expose as "checkbox" option
- cur-point mixinOn point
- vfd mixinOn motor
- zone-space mixinOn room
Choice

- Models an enumerated option in a definition
- May be narrowed in a subtype
- May be narrowed in an instance via markers
- Two components: aspect tag itself and its choice marker enumeration
Point Function

def: ^point
pointFunction:
^pointFunctionType

---
def: ^pointFunction
is: ^aspect

---
def: ^pointFunctionType
is: ^marker

----
def: ^sensor
is: ^pointFunctionType

---
def: ^cmd
is: ^pointFunctionType

---
def: ^sp
is: ^pointFunctionType
Equip Function

def: ^equipFunction
is: ^aspect
of: ^phenomenon
---
def: ^heats
is: ^equipFunction
of: ^substance
---
def: ^ahu
heats: ^air
---
def: ^boiler
heats: ^fluid
---
def: ^hot-water-boiler
heats: ^hot-water
Reflection

- Reflection: dict instance -> defs
- Implementation: defs -> dict instance
- Goal to avoid disruptive changes to instance data
- Defs is optional feature (for near term)
- Do not require inference (ahu infers equip, but we are still going to require equip tag)
Implementation

• Def symbol becomes tag name
• Conjunct parts become individual markers
• Any supertype marked mandatory becomes marker tags
• Shown in "usage" section of docs
Other Stuff

- Proposed enhancements to Filter to access def data – question mark operator
- RDF export (discussed in next session)
- Unresolved equip/point explosion
Documentation

• How to use the HTML documentation
• Review of ontology
2019 Haystack Connect

The place for the Project Haystack community to network, share, create synergy, and generate business opportunities.

May 13-15, 2019 / San Diego, California