ALLSEEN ALLIANCE

Designing an AllJoyn Interface

Dave Thaler
Interface Review Board (IRB) Member

Partner Software Engineer, Microsoft

Agenda

1. **Background**
2. Design Guidelines
3. Process for Authoring

What is the Interface Review Board?

- Creates guidelines for authoring AllJoyn interfaces
- Reviews any interfaces submitted to it for review
- Provides recommendations to the Technical Steering Committee regarding readiness for standardization of reviewed interfaces

- If you have additional suggestions for design advice, let us know!

What does it mean to define an interface?

- An interface has a set of properties, methods, and/or asynchronous signals
- An AllJoyn interface is defined independent of language binding
- The formal language for defining an AllJoyn interface is “introspection XML”
- Fully qualified interface name indicates whether it’s standard or proprietary:
  - Standard: **org.alljoyn**.Widget
  - Proprietary: **com.example**.Widget

Use Cases

- **Developers** need interface definitions for code generation
- **AllSeen Alliance** needs interface definitions to standardize
- **End-users** need descriptions of interfaces on a given device for creating If-This-Then-That (IFTTT) style rules with Events & Actions
- **Admins** need descriptions of interfaces on a given device for creating security policies

Introspection XML Formats

Historically there have been 3 variations of AllJoyn introspection XML

Format:	D-Bus style Introspect()	IntrospectWith Description()	Extended Introspection XML	(Future) Converged Introspect()
Remotely queryable	✓	✓	✕	✓
D-Bus tool compatible	✓	Some	✕	✓
Localizable user descriptions	✕	✓	✓	✓
Developer descriptions	✕	✕	✓	✓
Named types	✕	✕	✓	✓
Enumerations	✕	✕	✓	✓

Does it matter which format version?

- Convergence will happen, so at a high level: No
- But existing specs/tools vary in which format version(s) they use
- They should all evolve to use the converged version

- The guidance discussed today is independent of format version

Design Guidelines

Factoring Interfaces Appropriately

Goal #1: predictability and ease of use by a consumer app

Goal #2: ease of implementation of a producer app

- Don’t duplicate functionality already available in some other interface
- Split the functionality into logical blocks, with one interface per block
- Put optional functionality into separate interfaces
  - Implementation granularity should be at the interface level
- If some functionality is intended to be permitted only to a specific role, put it in a separate interface
  - Security granularity should be at the interface level

Designing for Evolution

- Never remove or change members of an interface
- Do not add proprietary extensions to a standardized interface
  - Define an auxiliary interface instead
- Every new version of an interface must be a strict superset of the previous version of that interface

Designing Interface Members

- Make objects type-safe
  - Avoid the “variant” type
  - Avoid alternative marshaling schemes such as structured data inside a string or binary blob (XML, JSON, CBOR, etc.)
- Use properties, methods, and signals appropriately
  - Use properties to represent externally observable state
  - Don’t use write-only properties – use methods instead
  - Only make a property read/write if makes sense to set in isolation
  - Don’t use custom signals to distribute property change information

Properties natively support “changed” signals

Specify EmitsChangedSignal behavior explicitly in XML:

- “true”: emits signal with the new value
  - Most common behavior
  - Allows caching value for efficiency
- “invalidates”: emits signal without value
  - Used if size of value is large
  - Still allows caching value for efficiency
- “false”: no signal
  - Never cached
  - Used if changes very frequently enough to negate any benefit of caching
- “const”: value will never change (not supported until AllJoyn 16.04)
  - Allows caching
  - Use “true” until “const” is supported

Human-readable descriptions

- **End-user** descriptions go in <description> elements (or AllJoyn DocString annotations in the future) associated with a language tag

```
<property name="IsOpen" type="b" access="read">
   <description language="en">
     True if door is open, false if closed.
   </description>
</property>
```

- Additional **developer**-only text, if any, goes in XML comments

Process for Authoring

Overview of authoring mechanics

1. Create introspection XML using an XML editing tool
2. Run **ajxmlcop** tool to check it against IRB design guidelines
3. Generate code
4. Submit for IRB review and update based on feedback
   - Required for standard interfaces, but optional for proprietary

Creating XML

Can use whatever tool you want for creating XML

(e.g., Visual Studio, Xerces, Notepad++, XML Notepad 2007, etc.)

- For more discussion of tools see here

Validate against XML schema:

- D-Bus style Introspect() XSD: https://www.allseenalliance.org/schemas/introspect.xsd
- Extended Introspection XML XSD: https://wiki.allseenalliance.org/irb/extended_introspection_xml#schema_definition

ajxmlcop tool

Accepts any current XML format in 15.09

Three levels of output messages:

- **Errors:** things that are invalid XML. That is, code would fail somehow.
- **Warnings:** most IRB guidelines are in this category. Stuff that isn't enforced by code per se, just policy and which the IRB might say is ok in certain cases, such as in legacy interfaces.
- **Informational messages:** a few guidelines are in the "consider" category or have a condition that can't be checked by code, but which is useful to call to a reviewer's attention so can verify whether it's ok

Running ajxmlcop

Pass XML filename as argument:

```
c:\temp\xml>ajxmlcop Hae-v1.xml

WARNING-24: interface 'org.alljoyn.Hae' is missing annotation org.alljoyn.Bus.Secure="true"

WARNING-29: interface 'org.alljoyn.Hae' missing description element

WARNING-30: method 'org.alljoyn.Hae.SetLocation' missing description element

WARNING-25: property 'org.alljoyn.Hae.Location' missing EmitsChangedSignal annotation

INFO-27: property 'org.alljoyn.Hae.Location' is readwrite, only appropriate if independent
of all other properties

WARNING-32: property 'org.alljoyn.Hae.Location' missing description element

==========================================================

0 errors, 5 warnings, 1 informational messages
```

Generating code

- You can write code by hand, but easier to auto-generate if you can
- Microsoft’s code generator for Standard Client apps on Windows 10
  - Supports Introspect() and IntrospectWithDescription() formats
  - http://go.microsoft.com/fwlink/?LinkID=623246
- Alliance’s code generator for Thin Client and Android Java apps
  - Supports all three current XML format variants
  - https://wiki.allseenalliance.org/devtools/code_generator
- D-Bus C code generator also exists but *doesn’t* have AllJoyn support
  - Supports D-Bus Introspect() format
  - https://developer.gnome.org/gio/stable/gdbus-codegen.html

Submitting for IRB review

Instructions on IRB page on Alliance website and in repository's README.md:

1. Clone the interfaces git repository
2. Create a textual description of the interface using the template provided
3. Create a formal XML description of the interface
4. Add both to the appropriate location in the repository
5. Submit the change via Gerrit infrastructure, adding "IRB" as code reviewer

IRB will provide feedback through Gerrit within two weeks or less

Summary

Interfaces are defined in XML

Previous separate formats are converging

The same XML is used for many purposes

Design Interfaces for Ease of Consumption

Follow approved IRB guidelines

Use ajxmlcop to check

Alliance Interfaces go through IRB review

Custom ones are also welcome

Send us suggestions!

allseen-irb@allseenalliance.org

Thank you

Follow us on

For more information on AllSeen Alliance, visit us at:
allseenalliance.org & allseenalliance.org/news/blogs