AllSeen Simplified API Project Proposal

Prepared for: AllSeen Core Workgroup

Prepared by:

Kristof Martens - AllSeen Software Development Lead - Technicolor

Dominique Chanet - AllSeen Lead Software Architect - Technicolor

PROJECT DEFINITION

Project Description

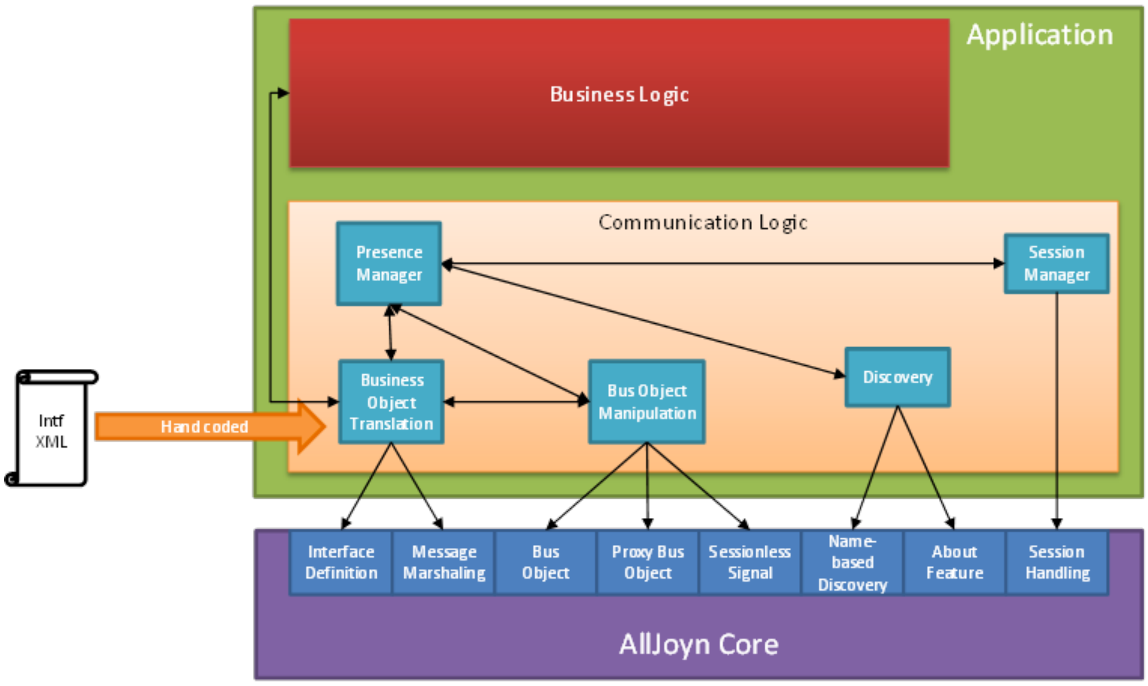
AllJoyn is a toolbox that offers a lot of fairly low-level functionality. With this toolbox, developers can create tailor-made solutions for their particular distributed application use case. This approach works great if you're building one-off, ad hoc distributed applications. In the Internet of Things context, however, there is a lot of value if devices and applications can transcend this silo mentality, and offer their services and information in a way that is standardised and reusable. Therefore, it no longer makes sense to have each device or application craft its own discovery and session management system out of the building blocks provided by AllJoyn. Building further on the theme of reusable services, we propose some extensions to the current AllJoyn type system. The goal of these extensions is to increase the expressiveness of the interface definition language in such a way that AllJoyn interfaces become much more self-describing. By describing an application's (or a device's) services in a formal definition language, and using a code generator to produce language bindings for this description, we can drastically improve the chances for interoperable implementations, while also increasing developer convenience. The proposed extensions to the type system have been described on the Alliance wiki (https://wiki.allseenalliance.org/core/extensions_to_the_type_system_proposal), and have been discussed on the AllSeen Core mailing list.

In addition, the distributed many-to-many communication scenarios that arise with the advent of the Internet of Things map very well on the Publish-Subscribe information exchange paradigm. We'd like to introduce this paradigm as a first-class member of the AllJoyn APIs. Publish-Subscribe could then be used for the more data-oriented use cases (discovery, sensor readings, allowing a device to gather information about its surroundings, ...), while the traditional RPC model can play an important role in the configuration and control type of use cases.

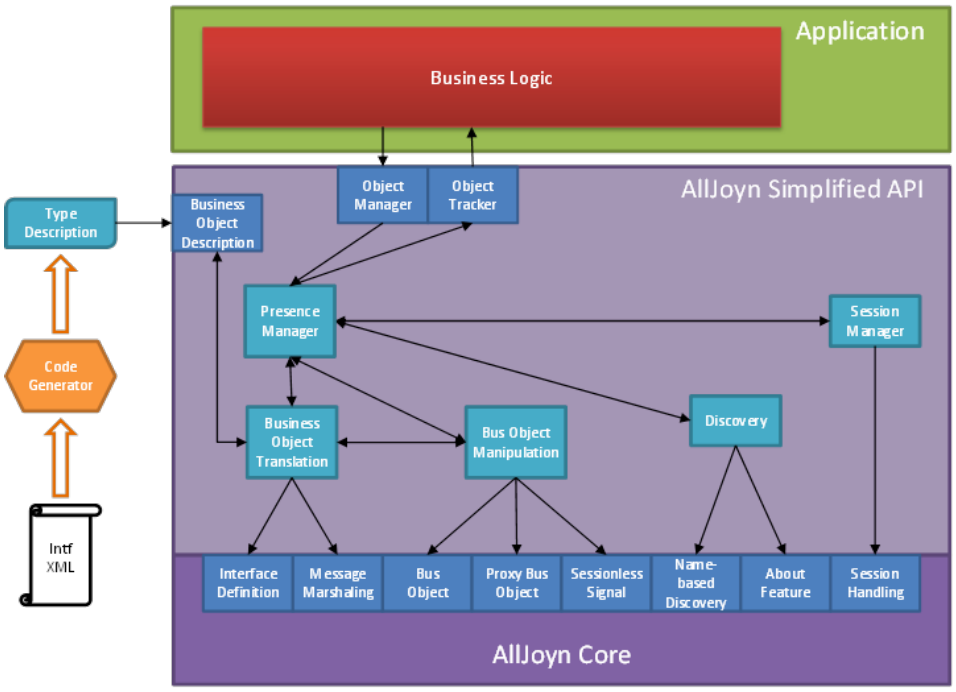
In summary, we propose to create a simplified AllJoyn API that operates at a higher conceptual level than the current API. We aim to achieve this goal by:

- shielding application developers from the minutiae of discovery, connectivity and session management. Underneath the simplified API, there will be a common methodology for discovery and session setup.
- extending the AllJoyn type system, and leveraging a code generation tool, to allow developers to express their application logic more directly in terms of business objects.
- introducing the Publish-Subscribe paradigm for data-oriented use cases
- overall reducing the number of distinct API concepts a developer has to understand when programming for AllJoyn. This will reduce the steepness of the learning curve, and attract more developers to the AllSeen ecosystem.

Architectural diagrams



High-level architecture of an application built on top of the current AllJoyn API.



This picture shows how the proposed simplified API implements a lot of the application's communication logic in a standardised way, and exposes a more limited set of API concepts to the application developer.

PROJECT SCOPE

Project name proposal

We propose the name of this project to be simplified_API.
We also propose to start working on a separate GIT repository part of the core working group to get things started. It needs to be discussed within the workgroup whether this new API will be part of the AllJoyn GIT repository at some point.

Scope

- Publish/Subscribe based mechanism
- Generalised interface discovery and session setup
- Simplified API and boilerplate code reduction
- Code generator updates taking into account the boiler plate reduction
- Extended type system and code generator to allow for:
    - Optional fields
    - Enums
    - Named structs
    - …
- Sample implementation showcasing the usage of the new API's and code generator improvements
- Full documentation (Developer Guide, Reference Manual, Best Practices, ...) for the simplified API

Completion criteria

- Consensus on the shape of the simplified API
- Simplified API integrated into AllJoyn Core Framework
- Language bindings available for all supported platforms

Project dependencies

- AllSeen Core Workgroup: This project will be part of the AllSeen Core working group and will obviously depend on the work being done in this workgroup.
- Developer Tools Workgroup: There will also be a dependency on the Developer Tools working group for the proposed changes to the code generator to make use of the proposed improved marshaling mechanism.

PROJECT DETAILS

Initial contribution

Our initial contribution will consist of:

- Taking part in discussions between all parties to define a simplified API
- Writing working code and tests for C++
- Sample implementation showcasing the simplified API proposal
- At a later phase we will also contribute more extensive tests and documentation on how to use the new API's

Proposed release schedule

Our goal is to make deliveries that will fit within the AllSeen release schedule

AllSeen 14.6:

- Propose a working POC simplified API including Pub/Sub support in C++
- Code generator for C++ POC based on extended marshalling support
- Sample implementation making use of the new API's and code generator in C++

AllSeen 14.10:

- Optimise core functionality to new API's
- Add additional language bindings
- Integrate new API's as intrinsic part of core
- Write documentation about new API's and how to use them effectively

Project contributors

Maintainer:

Dominique Chanet - AllSeen Lead Software Architect Technicolor

Committers:

Dominique Chanet - Software Architect
Steven Aerts - Software Engineer
Paul Praet - Software Engineer
As'ad Salkham - Software Engineer
Ronny Vanhelmont - Software Engineer
Kris Verbeeck - Software Engineer
Kristof Martens - Team Lead