Gateway Connector Service API Guide

(Linux)

February 3, 2015

Contents

1 Introduction 3

1.1 Purpose 3

1.2 References 3

1.3 Acronyms and terms 4

2 Overview 5

2.1 Reference code 5

2.1.1 Source code 5

2.2 Obtain the Gateway Connector Service 5

2.3 Build a Gateway Connector Application 5

2.4 Setting up the AllJoyn framework and About feature 5

3 Steps to create Connector App 6

3.1 Initialize the AllJoyn framework 6

3.1.1 Create bus attachment and add authentication 6

3.2 Start Listening to Announcements 6

3.2.1 Register an Announcement Listener 6

3.3 Extend the Gateway Connector base class 6

3.3.1 Initialize the service 7

3.4 Update connection status 7

3.5 Retrieve Connector App ACLs 7

1 Introduction

1.1 Purpose

This document demonstrates how to use the AllJoyn™ Gateway Connector service to create Gateway Connector Application.

This document is intended for software engineers and assumes familiarity with the AllJoyn SDK.

1.2 References

The following are reference documents.

- *AllJoyn™ Framework Tutorial*
- *Introduction to AllJoyn™ Framework*
- *Gateway Service Framework Interface Definition*
- *Getting Started with the AllJoyn™ Service Framework for Java*
- *About Feature API Guide (Java)*

1.3 Acronyms and terms

Term	Definition
AllJoyn framework	Open source peer-to-peer framework that allows for abstraction of low-level network concepts and APIs
AllJoyn Device	A device existing on the network and using Alljoyn framework to communicate with another AllJoyn devices
ACL	Access Control List. Lists the set of devices/apps/objects/interfaces for which the user would like to enable the remote access.
GM App	Gateway Management Application. Enables end user to define and manage ACL(s) via the Controller app.
Controller App	An application which exists on the same network as the GM App and communicates with it using the Controller Framework to control Connector Apps and configure its ACL(s)
Connector App	Connector Application. An application that is controlled by the Controller App. via the GM App. and provides capabilities of remote access to the AllJoyn devices (Remoted Apps) and Exposed Services
Exposed services	Part of an ACL. Lists the AllJoyn services provided by the Connector App, that are being exposed for the AllJoyn Devices.
Remoted Apps	Part of an ACL. List of AllJoyn applications with remote access permission.
Manifest	Manifest file provides metadata about the Connector Apps as well as the Exposed Services that the Connector App exposes and the object path/interfaces the Connector App is ready to remote.

2 Overview

2.1 Reference code

The reference code consists of a Gateway Connector Service and the Gateway Connector sample application.

2.1.1 Source code

https://git.allseenalliance.org/cgit/gateway/gwagent.git/tree/cpp/GatewayConnector lists the repositories used to build the Gateway Connector sample application.

Table 1. Gateway Controller Framework components

Package	Description
Alljoyn	The Standard Client Alljoyn code.
ControllerService	Gateway Controller Service
Services Common	Code that is common to the AllJoyn service frameworks.
Sample Apps	Code that is common to the AllJoyn service framework sample applications.

2.2 Obtain the Gateway Connector Service

See the *Getting Started with the AllJoyn™ Gateway Connector Service (Linux)* section for instructions on compiling the *Gateway Connector Service*.

2.3 Build a Gateway Connector Application

The following steps provide the high-level process to build a Gateway Connector application.

1. Create the base for the AllJoyn application.
2. Implement the security authentication mechanism.
3. Implement the Announcement Listener
4. Extend the Gateway Connector base class and implement the callbacks logic
5. Implement the logic of connecting the application to the cloud.

2.4 Setting up the AllJoyn framework and About feature

The steps required for this service framework are universal to all applications that use the AllJoyn framework and for any application using one or more AllJoyn service frameworks. Prior to use of the Gateway Connector Service, the About feature must be implemented.

Complete the procedures in the following sections to guide you in this process:

- *Getting Started with the AllJoyn™ Service Framework (Linux)*
- *AllJoyn™ About Feature API Guide (Linux)*

3 Steps to create Connector App

3.1 Initialize the AllJoyn framework

See the *Getting Started with the AllJoyn™ Service Framework (Linux)* section for instructions to set up the AllJoyn framework.

3.1.1 Create bus attachment and add authentication

```
BusAttachment* bus = new BusAttachment("ConnectorApp", true);
bus->Start();
bus->Connect();
bus->EnablePeerSecurity("ALLJOYN_PIN_KEYX ALLJOYN_SRP_KEYX ALLJOYN_ECDHE_PSK",
authListener);
```

3.2 Start Listening to Announcements

3.2.1 Register an Announcement Listener

```
AnnounceHandler* announceHandler = new MyAnnouncementHandler(onAnnouncement);
AnnouncementRegistrar::RegisterAnnounceHandler(*bus, *announceHandler, NULL,
                                                     0);
```

3.3 Extend the Gateway Connector base class

Extend GatewayConnector base class to start using the service and implement the relevant abstract methods.

```
class MyApp : public GatewayConnector {

protected:

        virtual void shutdown() {…}

        virtual void mergedAclUpdated() {…}

        void receiveGetMergedAclAsync(QStatus unmarshalStatus,

                GatewayMergedAcl* response) {…}
}
```

1. Implement `shutdown` callback to execute GM App instruction to stop this Connector App
2. Implement `mergedAclUpdated` callback to be notified when a Controller App changed one of this Connector App's ACLs
3. Implement `receiveGetMergedAclAsync` callback to receive `GatewayMergedAcl` response if the method `getMergedAclAsync` was previously invoked.

3.3.1 Initialize the service

```
qcc::String wellknownName = "MyApp1";

MyApp myApp(bus, wellknownName.c_str());
```

Provide the constructor with the `BusAttachment` and the name of the Connector application. This name is used to construct the Gateway Connector Well-Known-Name.

```
QStatus status = myApp.init();

if (ER_OK != status) {

       //Failed to initialize the service

}
```

3.4 Update connection status

Connector App is requested to update the GM App about its connection status to the cloud.

Use `GatewayEnums.h` provided by the GM App to use the `ConnectionStatus` enum.

```
QStatus status = myApp.updateConnectionStatus(ConnectionStatus::GW_CS_CONNECTED);
```

3.5 Retrieve Connector App ACLs

There are two ways to retrieve ACLs of this Connector App. First is to retrieve it synchronously from the GM App:

```
GatewayMergedAcl mergedAcl;

QStatus status = myApp.getMergedAcl(&mergedAcl);
```

The second way is to retrieve it asynchronously. The mergedAcl response will arrive in the `receiveGetMergedAclAsync` as explained in the section 3.3.

```
GatewayMergedAcl mergedAcl;

QStatus status = myApp.getMergedAclAsync(&mergedAcl);
```