

AllJoyn™ Gateway Agent

High-Level Design

Rev 1 Update 1

June 13, 2014

This work is licensed under a Creative Commons Attribution 4.0 International License.

<http://creativecommons.org/licenses/by/4.0/>

Any and all source code included in this work is licensed under the ISC License per the AllSeen Alliance IP Policy.

<https://allseenalliance.org/allseen/ip-policy>

AllJoyn is a trademark of Qualcomm Innovation Center, Inc. AllJoyn is used here with permission to identify unmodified materials originating in the AllJoyn open source project.

Other products and brand names may be trademarks or registered trademarks of their respective owners.

Contents

1 Introduction	7
1.1 Document scope	7
1.2 Document organization	7
1.3 Reference documents	7
1.4 Revision History	7
1.5 Acronyms and terms	8
1.6 Product requirements	8
1.7 Design scope	8
1.7.1 Design assumptions	8
1.7.2 Current dependencies	8
1.7.3 Ongoing dependencies	9
1.7.4 Out of scope	9
2 Gateway Agent Design	10
2.1 Overview	10
2.2 Gateway concepts	11
2.2.1 Service provider	11
2.2.2 Remote profile	12
2.2.3 Remoted AllJoyn device/app/interface	12
2.3 System architecture	12
2.3.1 Gateway Management application	13
2.3.2 Connector app	14
2.3.3 AllJoyn Router	16
2.3.4 Control app	16
2.3.5 Entity relationship	17
2.3.6 Accessing secure interfaces	18
2.3.7 Secure Gateway Management app interfaces	19
2.4 Call flows	20
2.4.1 Gateway Agent discovery	20
2.4.2 Profile management	21
2.4.3 Notification	28
2.4.4 AllJoyn device access	30
2.5 Functional design	33
2.5.1 Profile management interface	33
2.5.2 App Access interface	39
2.5.3 Fields	43
2.5.4 Config file policy enforcement	43
2.6 Versioning	45

3 App Management.....	46
3.1 Architecture	46
3.1.1 App management at the Gateway Management app	47
3.1.2 Package Manager app.....	48
3.1.3 App management at the Control app	49
3.2 Call flows	49
3.2.1 Connector app install	50
3.2.2 Connector app upgrade	51
3.2.3 Connector app uninstall	52
3.2.4 Connector app restart	53
3.2.5 Add a new device/app.....	54
3.2.6 Remove a device	55
3.3 Functional design	55
3.3.1 App Management interface.....	55
3.3.2 Package Manager app functions	59
3.3.3 AllJoyn package format.....	60
3.3.4 Fields	61
4 Performance	62
5 Requirements	63
5.1 Overall requirements	63
5.2 Gateway discovery	64
5.3 Profile management	64
5.4 App access for Connector apps	65
5.5 Config file policy	65
5.6 App management	66
6 Configuration Parameters.....	68
Appendix A Gateway Agent Architecture with Security 2.0	69
A.1 Accessing secure interfaces with Security 2.0	69

Figures

Figure 2-1 Gateway Agent context architecture	11
Figure 2-2: Gateway Agent architecture	13
Figure 2-3 Gateway Agent Entity Relationship	18
Figure 2-4 Gateway Agent Security 1.0	19
Figure 2-5 Gateway Agent discovery	21
Figure 2-6 Create profile	22
Figure 2-7 Activate profile	23
Figure 2-8 Connector app startup	24
Figure 2-9 Update profile	25
Figure 2-10 Delete profile	26
Figure 2-12 Deactivate profile	27
Figure 2-14 Outgoing notification	28
Figure 2-15 Incoming notification	29
Figure 2-16 Method invocation	30
Figure 2-17 Get property	31
Figure 2-18 Set property	32
Figure 2-19 Remoting session-based signals	33
Figure 3-1 Gateway Agent app management architecture	47
Figure 3-2 Connector app install	50
Figure 3-3 Connector app upgrade	51
Figure 3-4 Connector app uninstall	52
Figure 3-5 Connector app restart	53
Figure 3-6 Add a new device/app	54
Figure 7-1 Gateway Agent Security 2.0	69

Tables

Table 1-1 Gateway Agent design dependencies	8
Table 1-2 Out of scope items	9
Table 2-1 Profile management interface functions	34
Table 2-2 Create profile method	34
Table 2-3 Activate profile method	36

Table 2-4 Get profile method	36
Table 2-5 Get profile status method	37
Table 2-6 Get profile list method	37
Table 2-7 Update profile method	38
Table 2-8 Delete profile method	39
Table 2-9 Deactivate profile method	39
Table 2-10 App Access interface functions	40
Table 2-11 Get profile method	40
Table 2-12 Update connection status method	42
Table 2-13 Profile updated signal	42
Table 2-14 Profile deleted signal	42
Table 2-15 Shutdown app signal	43
Table 2-16 Profile status	43
Table 2-17 Config file allow rules	44
Table 3-1 App details maintained at the Gateway Management app	48
Table 3-2 App Management interface functions	55
Table 3-3 Install App method	56
Table 3-4 Uninstall App method	56
Table 3-5 Restart App method	57
Table 3-6 Get App Status method	57
Table 3-7 Get Installed App method	57
Table 3-8 Get Manifest Data Method	58
Table 3-9 Get Manifest Interfaces method	58
Table 3-9 Get Manifest Interfaces method	58
Table 3-11 Package Manager app functions	59
Table 3-11 InstallApp function	59
Table 3-13 UninstallApp function	60
Table 3-14 Manifest file content	60
Table 3-15 Connection status	61
Table 3-16 Install status	61
Table 3-17 Operational status	62
Table 5-1 Gateway Management application requirements	63
Table 5-2 Gateway discovery requirements	64
Table 5-3 Profile management requirements	64
Table 5-4 App access for Connector apps requirements	65
Table 5-5 Config file policy requirements	65

Table 5-6 App management requirements66

Table 6-1 Gateway Management app config parameters.....68

1 Introduction

1.1 Document scope

This document captures system-level design information for the AllJoyn Gateway Agent. Related interfaces and API design is captured at a functional level. Actual definition for interfaces and APIs is outside the scope of this document.

1.2 Document organization

This document is organized in following chapters:

Chapter 1 – Introduction: Provides introductory information including scope, document organization, definitions and acronyms, references, document revision history and design scope.

Chapter 2 – Gateway Agent Design: Captures system design for the AllJoyn Gateway Agent including system architecture, call flows and functional design.

Chapter 3 – App Management: Captures application management design for the Connector Apps installed on the Gateway Agent.

Chapter 4 – Performance: Captures performance requirement for the Gateway Agent.

Chapter 5 – Requirements: Captures system requirements for the AllJoyn Gateway Agent.

Chapter 6 – Configuration Parameters: Captures configuration parameters for the Gateway Management application.

1.3 Reference documents

Reference Number	Document Name	Document Path
1	Gateway Agent 1.0 PRD	

1.4 Revision History

Revision	Date	Change Log
Rev 1	April 22, 2014	First released version in Alliance
Rev 1 Update 1	June 13, 2014	Updates to reflect component name changes.

1.5 Acronyms and terms

Acronym/term	Description
AllJoyn framework	Open source peer-to-peer framework that allows for abstraction of low-level network concepts and APIs.
AllJoyn service framework	A service framework provided over the AllJoyn platform.
Consumer	AllJoyn application consuming services over the AllJoyn platform.
GM App	Gateway Management App
GUID	Globally Unique Identifier. A 128 bit identifier generated randomly in a way that the probability of collision is negligible.
GW	Gateway
IoE	Internet of Everything
PM App	Package Manager App
Producer	AllJoyn application providing services over the AllJoyn platform.
SLS	Sessionless Signal
WKN	Well-Known Name

1.6 Product requirements

This HLD focuses only on UC1 from the Gateway Agent 1.0 PRD. Main design deliverables are the Gateway Management application and policy enforcement at the AllJoyn Router. Any product requirements on the Connector application are considered out of scope.

1.7 Design scope

1.7.1 Design assumptions

The Gateway Management application and Connector app are running as separate applications on the Gateway Agent and communicating with the same AllJoyn router. This architecture enables these two applications to be run as separate processes with different privileges, namely, the Gateway Management application running with higher privileges. Since the Connector app is provided by the service provider (or another third-party entity) and since it connects with a cloud end point, it is considered untrusted entity.

1.7.2 Current dependencies

The Gateway Agent design assumes dependency on the following dependencies.

Table 1-1 Gateway Agent design dependencies

Dependency	Version
Security feature	1.0

Notification service framework	1.1
About feature	1.0

1.7.3 Ongoing dependencies

There are a number of ongoing dependencies (listed below) that will likely impact Gateway Agent design. The HLD will be enhanced as needed once the design is matured and related product requirements are defined.

- Security 2.0

1.7.4 Out of scope

The following items are considered out of scope of the current HLD.

Table 1-2 Out of scope items

Out of scope item	Resolution
Upgrade for the Gateway Management app	The app install/upgrade design will cover the install/upgrade/uninstall for the Connector applications. The architecture is designed to be flexible to accommodate Gateway Management app upgrade in future.
Error response codes for the Gateway Management app-related interfaces	This will be addressed as part of actual interface definition.
Interface design between the Gateway Management app and the Package Manger app	This will be addressed as part of implementation phase.

2 Gateway Agent Design

2.1 Overview

In a proximal AllJoyn IoE network, producer devices expose functionality like notification and control to other consumer devices on the proximal network. This enables consumer devices to receive notifications about events or state changes on producer devices and display them to the user. This also enables user-initiated or machine-initiated control actions to be performed on producer devices e.g. turning on/off the device, updating device settings etc. It is desirable to have a mechanism to *remote* such device functionality so that users can have the seamless experience of receiving notifications and/or controlling devices while away from the proximal network. This is also desirable for home automation use cases. Gateway Agent is designed to provide such a mechanism.

Figure 2-1 shows context architecture for enabling remote access to services provided by devices in the proximal network.

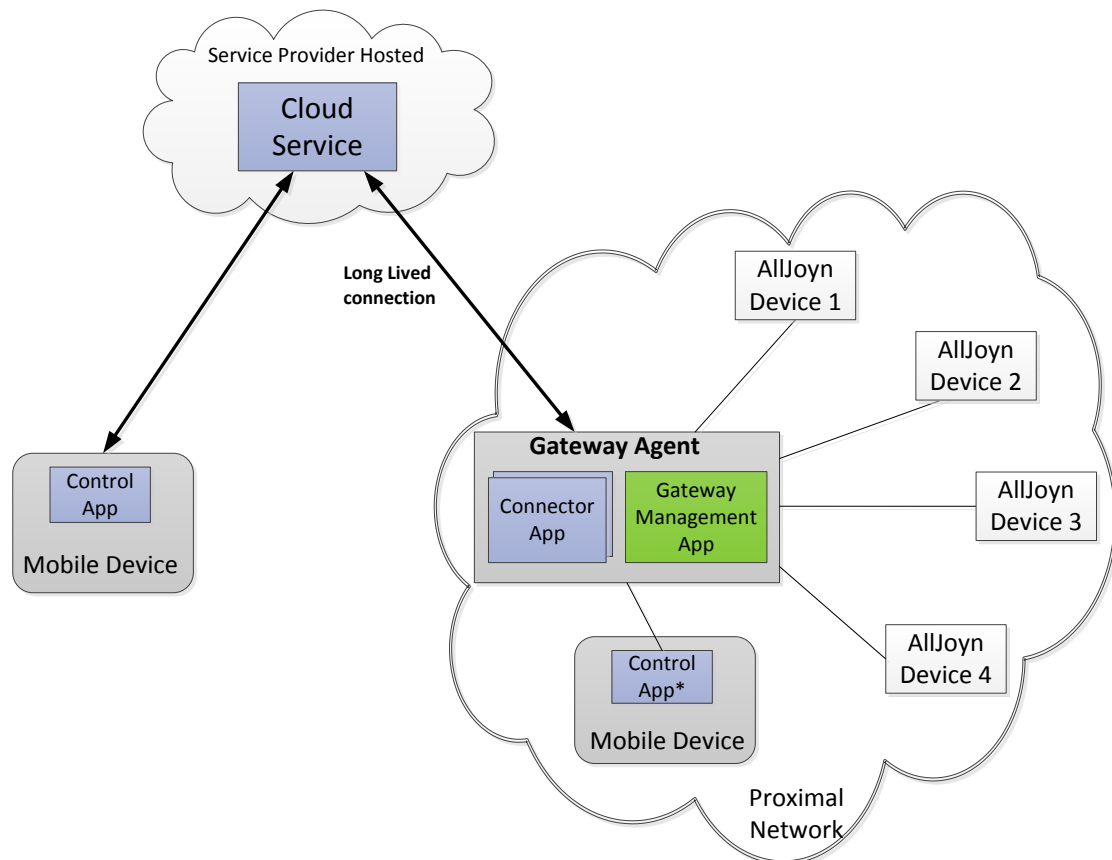


Figure 2-1 Gateway Agent context architecture

User would sign up with a service provider for accessing AllJoyn device services remotely. Service provider would enable remote services access via a Control app that can be used to control devices or send/receive notifications. When in the proximal network, the Control app will likely interact with devices over AllJoyn peer-to-peer communication. When outside the proximal network, the Control app will interact with a cloud service hosted by the service provider.

A new Gateway Agent component is added in the proximal network for enabling remote access. Gateway Agent has a single Gateway Management app (GM app) and one or more Connector apps developed by 3rd party or service provider. Connector app enables a Control App situated outside of the proximal network to connect with IoT devices in the proximal network via the cloud service. The GM App and Connector apps are AllJoyn enabled and interact with a single preinstalled AllJoyn Router on the Gateway Agent. The Gateway Management app enables user to define and manage remote profiles via the Control app. A remote profile lists the set of devices/apps/interfaces for which the user would like to enable the remote access. The remote access to AllJoyn devices is controlled via the config file at the AllJoyn Router. The Gateway Management app is responsible to update the config file appropriately to only allow access to remoted AllJoyn devices. Remote profile information is exposed to Connector apps via an AllJoyn interface provided by the Gateway Management app.

The Connector app maintains a long-lived connection with the service provider cloud service. This persistent connection is used for receiving control actions initiated by user remotely on the Control app or by the cloud service itself (e.g., in case of home automation). The communication between the Connector app and the cloud service is via the service provider defined web protocol. Within the proximal network, the Connector app interacts with remoted devices/apps/interfaces directly for control actions using the AllJoyn framework via the preinstalled AllJoyn Router. The AllJoyn Router enforces allow rules via the config file that was updated by the Gateway Management app based on the remote profile setting. A Connector app also acts as a proxy for propagating notifications from the proximal network to the cloud service and vice versa.

When outside the proximal network, the Control app will interact with a cloud service hosted by the service provider. The cloud service would then interact with the Connector app on the Gateway Agent. The Connector app will provide protocol conversion from service provider's protocol to AllJoyn-based protocol and invoke interaction with remoted interfaces on the devices as described above.

2.2 Gateway concepts

2.2.1 Service provider

The Service provider provides cloud based services for IoT devices in user's proximal network, e.g., home automation, home security, remote control etc.

2.2.2 Remote profile

The Gateway Management app maintains one or more remote profiles as created by the user via the Control app. A remote profile defines IoE devices/apps/interfaces that need to be remotely accessible. A remote profile is associated with a single service provider.

A remote profile can only be configured by the Control app from within the proximal network. Profile configuration is not allowed remotely for security reasons.

2.2.3 Remoted AllJoyn device/app/interface

An AllJoyn device/app/interface that is listed as part of a remote profile on the Gateway Management app is termed as remoted.

2.3 System architecture

Figure 2-2 shows detailed system architecture for the Gateway Agent. Each Gateway Agent is associated with a single service provider. A service provider could provide one or more Cloud Services for remote access to IoE devices. The Gateway Agent will have a Connector application interacting with a given cloud service. The Gateway Agent can have multiple Connector applications interacting with different cloud services providing remote access for IoE devices. There can also be multiple Gateway Agents deployed in a proximal network each enabling remote access for IoE devices via a different service provider.

To ensure that Connector applications (which can be untrusted) cannot manipulate remote access policies as defined by remote profiles, the Connector app and the Gateway Management app should be installed in two separate user accounts. These two apps will be talking to the same AllJoyn Router preinstalled on the Gateway Agent. Remote access to AllJoyn devices is restricted via the config file policy at the AllJoyn Router. The Gateway Management app updates the config file policy based on the profile activation and deactivation.

The Gateway Management application exposes specific AllJoyn interfaces as depicted in Figure 2-2. These include: :

- Profile Management interface used by the Control app
- App Access interface used by Connector applications.

Access to remoted AllJoyn devices by the Connector apps is controlled via the config file policy enforcement at the AllJoyn Router.

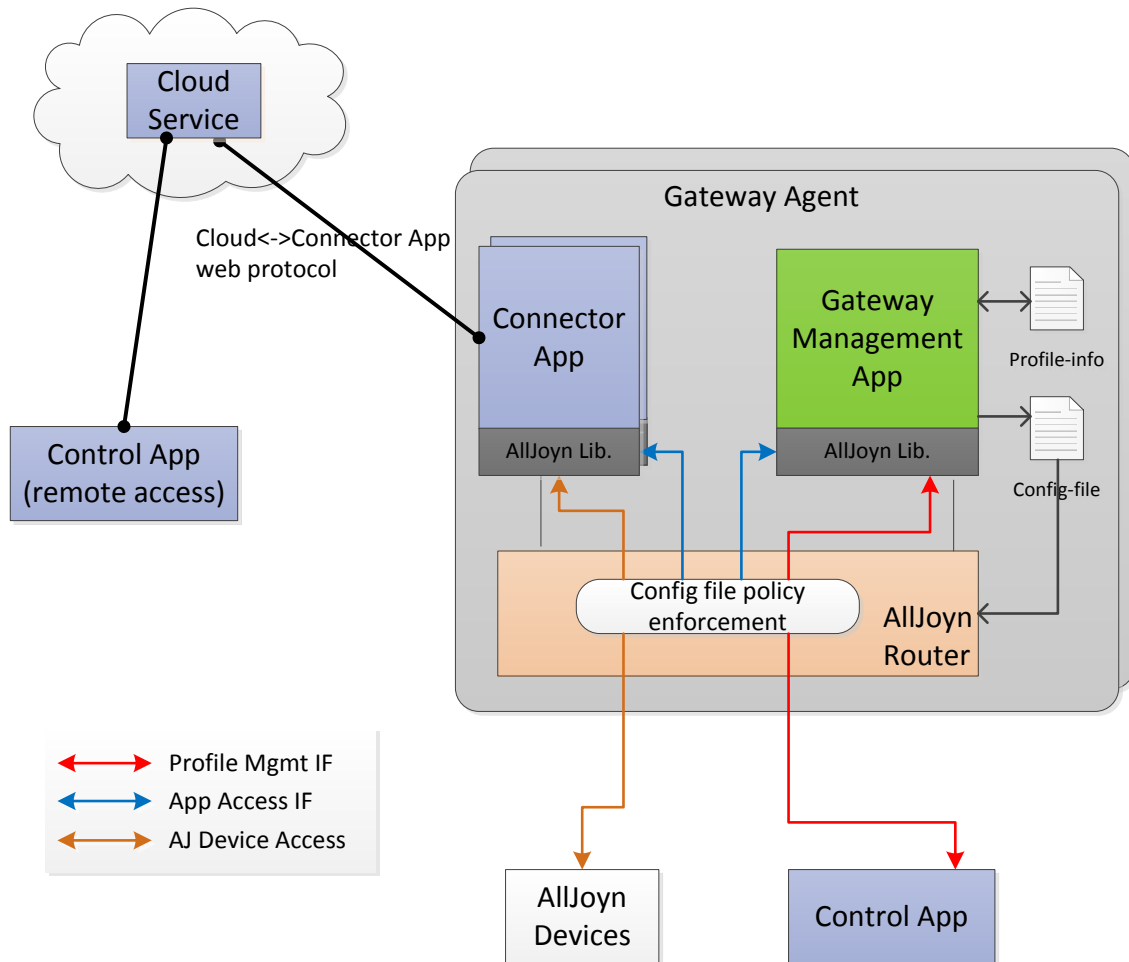


Figure 2-2: Gateway Agent architecture

2.3.1 Gateway Management application

The Gateway Management app provides following main functionality:

- **AllJoyn Profile Management:** Gateway Management app implements a secure AllJoyn Profile Management Interface. This is a generic interface independent of any service provider and exposes profile management functionality. It is advertised in the About announcement signal sent out by the Gateway Management App and is used by the Control app to discover the Gateway Management app. The 'App Name' field in the announcement signal can be set to indicate service provider-specific Gateway Agent, e.g., "Service Provider 1 AllJoyn Gateway App".

The Profile Management interface enables the Control app to configure devices/apps/interfaces for remote access as part of the profile. The Gateway Management app is also responsible for persistently storing profile data. There can be multiple simultaneously active profiles on the Gateway Management app. Upon activation of a profile by the Control app, the Gateway Management app is also

responsible for updating policies in the config-file (used by AllJoyn Router for authorizing device access by the Connector apps).

The Gateway Management app makes use of the UNIX user ID associated with Connector apps for creating config file policy rules. For AllJoyn devices, the Gateway Management app uses their unique name in the config file policy rules. To get the latest unique names for remoted AllJoyn apps, the Gateway Management app listens to AllJoyn announcements from these devices. Whenever unique name changes for remoted device apps, the Gateway Management app is responsible for updating the config file.

NOTE

A given device/app/interface can be configured as part of multiple profiles. In this case, if a service gets invoked on that device/app/interface via two different profiles, the last action will persist.

- **App access for Connector apps:** The Gateway Management app exposes an App Access interface for Connector apps to fetch profile data for active profile provisioned for that Connector app. The use case is to provide details for devices/apps/interfaces that have been configured for remote access to the Connector app and eventually to the cloud service. This interface also supports profile update indication (from Gateway Management app → Connector app) and Connection Status updates (from Connector app → Gateway Management app).

The App Access interface involves two-way communication between Gateway Management app and Connector app. Since the Gateway Management app and Connector app are talking to the same AllJoyn Router, there is no need to establish an AllJoyn session between them for establishing a communication path. These apps can communicate with each other as long as they know each other's end point address (well-known name, or WKN). The Gateway Management app and Connector app register following WKNs with the AllJoyn Router and communicate with each other using these WKNs.

Gateway Management app WKN: "org.alljoyn.GWAgent.GMApp"

Connector app WKN: "org.alljoyn.GWAgent.Connector.<Connector app id>"

For security reasons, a given Connector app should be able to access only profile data associated with itself. This is enforced by maintaining profiles for each Connector app as part of different service objects on the Gateway Management app. The service object has a well known object path of the format /Profiles/<Connector app id>. Each Connector app is then allowed to access only the corresponding service object enforced via the config file policy written by the Gateway Management app.

2.3.2 Connector app

The Connector app is responsible for following functions:

- **Connection management with cloud service:** The Connector app maintains persistent connection with the cloud service based on the protocol supported by the cloud service. It maintains its own set of connection related data e.g. cloud service

URL, transport type (e.g. WebSocket, TCP socket) etc. It also maintains any auth credentials required for connecting with the cloud service.

- **Data exchange with cloud service:** The Connector app implements service provider-specific web protocol to exchange data with the cloud service for incoming and outgoing traffic to/from the proximal AllJoyn network. This protocol can be standard based e.g., HTTP/REST using XML/JSON or can be a proprietary protocol supported by the cloud service.
- **Protocol translation:** The Connector app sits in-between the cloud service and the proximal AllJoyn devices. It performs translation of data it received from cloud service over web-based protocol to the AllJoyn format for invoking device interfaces. In the reverse direction, it performs translation of data it received from AllJoyn devices to web-based protocol format for delivering to cloud service.
- **AllJoyn device access:** The Connector app is responsible for accessing AllJoyn devices over the AllJoyn framework. Note that the Connector app can send/receive AllJoyn messages for a particular device/app/interface *only if* authorized to do so in the config file. This file is written to by the GM App and used by the AllJoyn Router to enforce policies. In the reverse direction, AllJoyn signals emitted from AllJoyn devices are received by Connector apps only if authorized as per config file policy filtering rules at the AllJoyn Router.

As part of this functionality, the Connector app is also responsible for listening to discovery announcement signals for devices/apps that have been configured as part of the profile to have the latest connectivity information for those endpoints.

- **Notification service framework:** The Connector app supports consumer and producer side of the Notification service framework. The Consumer Notification service framework listens for notifications produced by IoE devices/apps in the proximal network. These notifications get filtered based on the config file policy settings at the AllJoyn Router and then get sent to the appropriate Connector apps. A Connector app can also receive incoming notifications from the cloud service to be propagated inside the proximal network. The Connector app will construct the AllJoyn notification message for incoming notification and send it to connected AllJoyn Router. The AllJoyn Router will send it out if allowed per the config file policy settings.

The policy settings in the AllJoyn Router config file for notifications will enable incoming and outgoing notifications filtering based on notification types. The Notification service framework supports different object path for each notification type and notification filtering policy rules in the config file can be set based on that.

When in the proximal network, the Control app may receive notifications from the devices twice – once directly from the device and once from the Connector app as incoming notification via the cloud service. In this case, it is the responsibility of the Control app to perform duplicate elimination for notification message based on the notification message Id. It is recommended that service provider cloud service does not modify the notification message Id.

In a deployment with Super-Agent, the announcement signal from the Super-Agent will be blocked by the AllJoyn Router per config file policy rules and hence will not reach the Consumer Notification service framework at the Connector apps. Hence,

even with Super-Agent deployed, Connector apps will continue to receive notifications directly from the producer devices.

- **App access at the Gateway Management app:** A Connector app can use the App Access interfaces exposed by the Gateway Management app to fetch associated profile data and also receive profile updated/deleted signals. The Gateway Management app maintains profiles for each Connector app as part of separate service objects with a well know object path of the format `/Profiles/<Connector app id>`. When fetching the profile data, the Connector app should use the well-known object path corresponding to itself.

The Connector app provides its connection status with the cloud service to the Gateway Management app via the App Access interface.

- **Exposing services on the proximal network:** Connector apps can also expose one or more services to AllJoyn devices in the proximal network. Such a service can comprise exposing a cloud based service to devices over the AllJoyn framework (e.g., a weather update service). A user must explicitly select which of the supported services of a Connector app should be exposed for access over the proximal AllJoyn network as part of profile information.

The Producer Notification service framework will be one of the exposed services if a Connector app is interested in producing notifications on the proximal network.

2.3.3 AllJoyn Router

The AllJoyn Router is responsible for enforcing access policies for the Connector app via the config file.

Config file policy enforcement: The Connector app directly accesses remoted AllJoyn interfaces on the proximal devices. Access to these interfaces is granted as per the allow policies in the config file. This includes invoking method calls, GET/SET on properties and receiving signals from service interfaces configured as part of the profile. AllJoyn Router also enforces access to only applicable profile data for a Connector app via config file allow policies. As described above, these policies get written by the GM App.

2.3.4 Control app

The Control app is an AllJoyn-enabled application developed by the service provider or an affiliated entity. This app lets user access their IoE devices in the proximal environment as well as remotely. The Control app should support following functions relevant to Gateway Agent.

NOTE

This provides a functional description of Control app how we see it now. Since the Control app is under the realm of service provider, only a subset of these functions may be supported.

- **User authentication with service provider's cloud:** A user will sign up with the service provider for the IoE cloud service. The Control app should provide a way to input user credentials and use that to authenticate the user with the service provider

cloud. This will ensure that only an authenticated user can use the Control app to configure the Gateway Agent.

- **AllJoyn Gateway Agent discovery:** The Control app should use the AllJoyn announcement-based discovery mechanism to discover the Gateway Management app. There can be multiple Gateway Agents deployed in the proximal network. The App Name field in the announcement signal from the Gateway Management app can be set to indicate any service provider-specific Gateway Agent as applicable. The Control app should use the App Name field to discover any service provider-specific Gateway Agent it is looking for.
- **AllJoyn devices discovery and filtering:** The Control app should discover AllJoyn devices/apps in the proximal network using AllJoyn announcement-based discovery. The Control app should filter these devices/apps/interfaces based on the manifest file details for a given Connector app when creating a remote profile for that Connector app. It is assumed that the Control app has access to manifest files either as part of the Connector app install or via separate download for already installed Connector apps. The goal is to only allow those devices/apps/interfaces to get provisioned as part of a remote profile that are supported by the corresponding Connector app.
- **UI for profile functions:** The Control app should provide a UI for a user to select devices/apps/interfaces for profile configuration on the Gateway Agent.
- **Profile configuration:** After discovering a desired Gateway Agent, the Control app invokes profile management method calls provided by the AllJoyn profile management interface to create a remote profile. A profile name is provided as part of profile creation along with other devices/apps/interfaces data. A profile ID is returned in response by the Gateway Management app. The Control app should maintain this profile ID for any future updates to the profile. An explicit call must be invoked by the Control app to activate a previously created profile that enables remote access for associated devices/apps/interfaces.

The Control app can make updates to an existing profile (active or inactive). Updates take effect immediately if it was an already active profile. The Control app can also deactivate an already active profile or completely delete a profile.

2.3.5 Entity relationship

A Gateway Agent has a single Gateway Management app and one or more Connector apps. The Gateway Management app supports one or more remote profiles. Each remote profile gets associated with a single Connector app. The architecture supports a 1:1 relationship between the remote profile and Connector app. Figure 2-3 shows the entity relationship between the Gateway Management app, Connector app, and remote profiles.

Each Connector app connects to a cloud service. Typically, each Connector app would connect to a different cloud service providing different types of remote access. However, a service provider may host a single front end cloud service supporting multiple back-end services. In that case, multiple Connector apps can connect to the same cloud service.

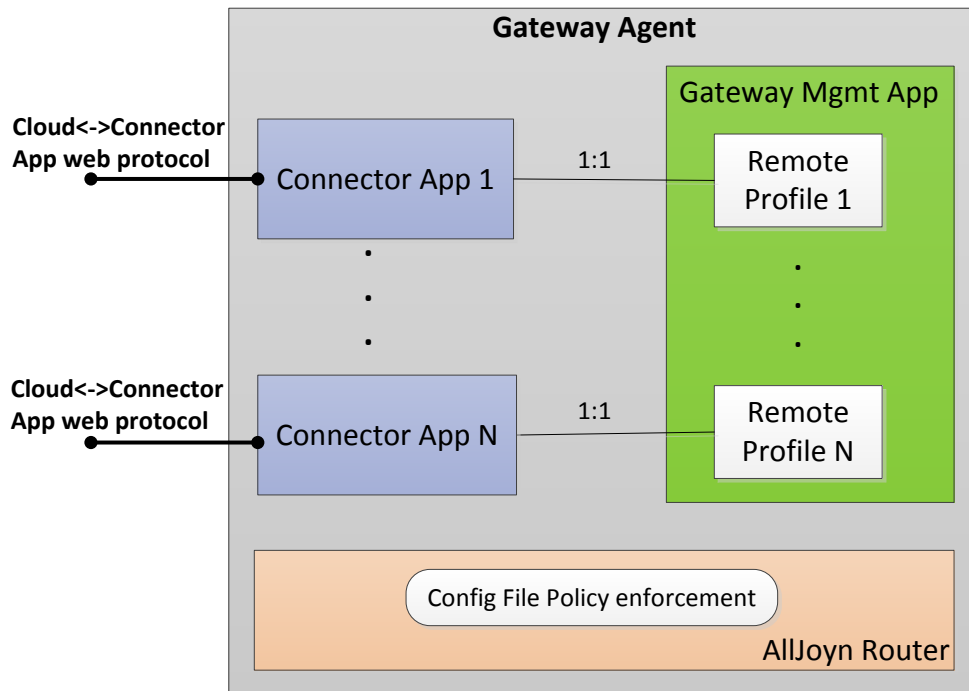


Figure 2-3 Gateway Agent Entity Relationship

2.3.6 Accessing secure interfaces

The AllJoyn interfaces being remoted through the profile would typically be secure interfaces for obvious security reasons. Also, all the AllJoyn interfaces exposed by the Gateway Management app to the Control app will be secure. Access to such secure interfaces is achieved via use of a password, PIN, or other key material (based on the AllJoyn auth mechanism used) in Security 1.0 as described below.

NOTE

In Security 2.0, secure interfaces are accessed via permissions granted by the user. Security 2.0 is currently under design phase and more details will be added later. See Appendix A for a possible architecture on how secure interfaces will get accessed with respect to the Gateway Agent in the Security 2.0 realm.

2.3.6.1 Security 1.0

Security 1.0 uses a password, PIN, or other key material (based on the AllJoyn auth mechanism used) for accessing secure interfaces. The password, PIN, or other key material is provided by the user at the Control app either as part of the profile creation or independently. The Control app queries AllJoyn devices/apps that need to be remoted for the type of authentication mechanism supported. It then asks the user for appropriate key material based on auth type information received from those devices/apps. The Control app is responsible for sending key material along with the auth type to the service provider's cloud service for remoted devices/apps. Every time a device access is

invoked via the cloud service, key material is provided as part of that request. For security reasons, a Connector app should not store the key material for device access.

The Control app needs the key material to access secure Profile Management and App Management interfaces exposed by the Gateway Management app. This key material (e.g., password or PIN) is entered by the user at the the Control app and can be cached by the app for future use.

NOTE

A user can have a single password setting for all the devices/apps in the proximal network or there could be multiple passwords for AllJoyn devices/apps.

Figure 2-4 shows the flow for inputting password and its usage for Security 1.0 as it relates to the Gateway Agent.

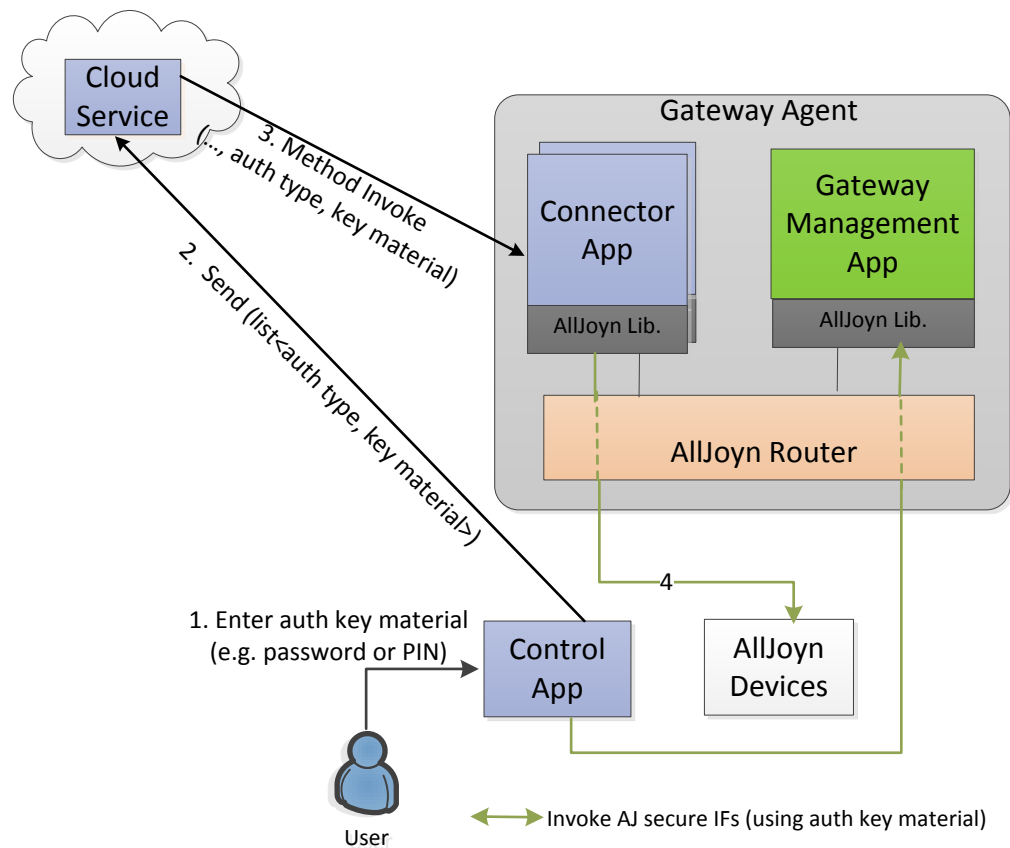


Figure 2-4 Gateway Agent Security 1.0

2.3.7 Secure Gateway Management app interfaces

The Gateway Management app provides secure AllJoyn interfaces for Connector app management and profile management. The Gateway Management app should support the ALLJOYN_SRP_KEYX auth mechanism for these interfaces in the Security 1.0

scope. A default password should be configured on the Gateway Management app and The Configuration service framework should be supported on the Gateway Management app so that the user can change this password via the Control app.

2.4 Call flows

This section captures call flows for various Gateway Agent use cases.

2.4.1 Gateway Agent discovery

The Control app will look for a Gateway Agent that has a Gateway Management app supporting the Gateway app management interface. Figure 2-5 illustrates the following use cases:

- The Control app is looking for a specific Gateway Agent identified by the AppName in the announcement signal from the Gateway Management app, for example, Service Provider 1 GW Agent.
- The Control app wants to discover any Gateway Agent.

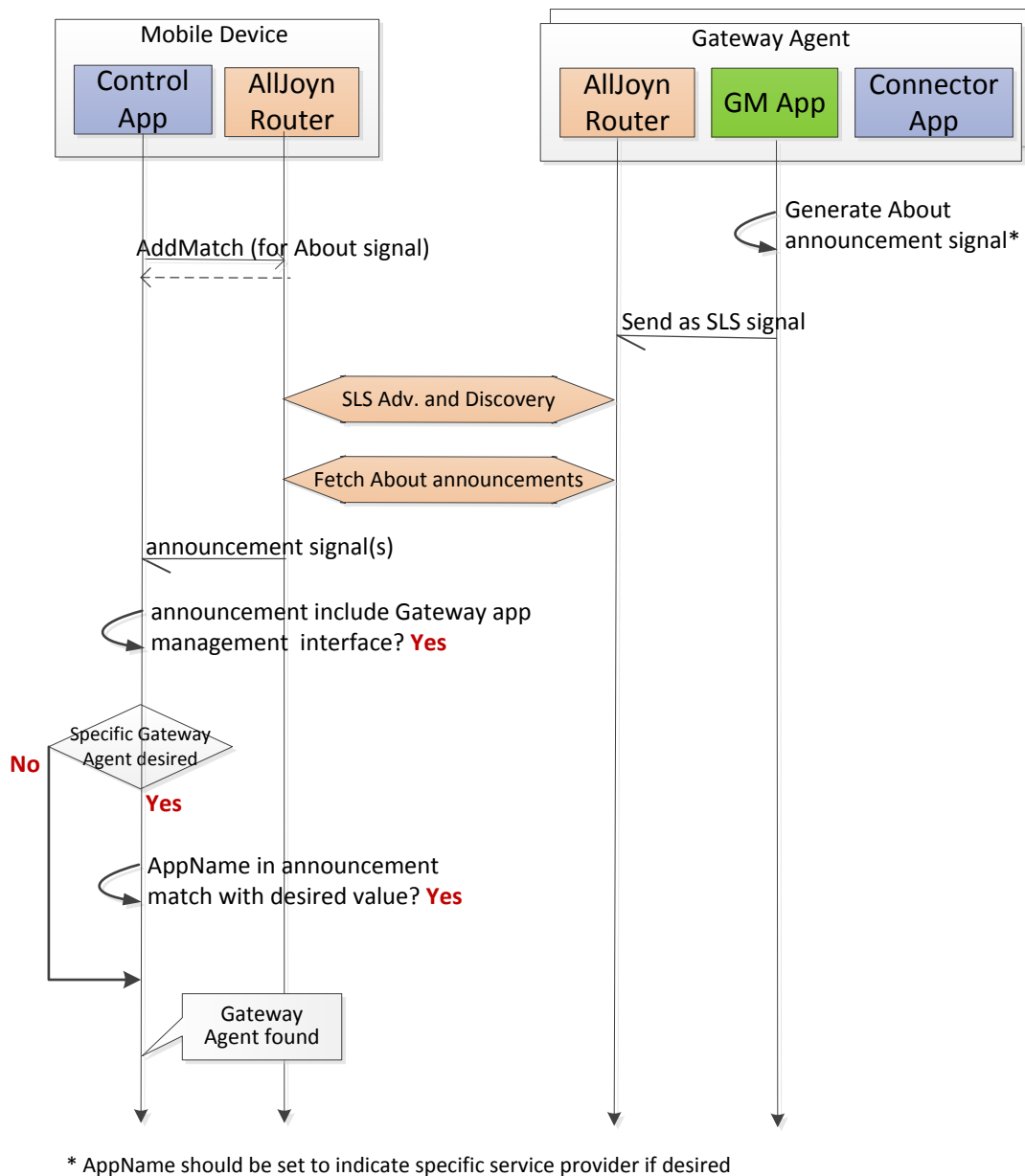


Figure 2-5 Gateway Agent discovery

2.4.2 Profile management

Profile management is provided by the AllJoyn Profile Management interface and supports functionality as captured in Table 2-1.

2.4.2.1 Create profile

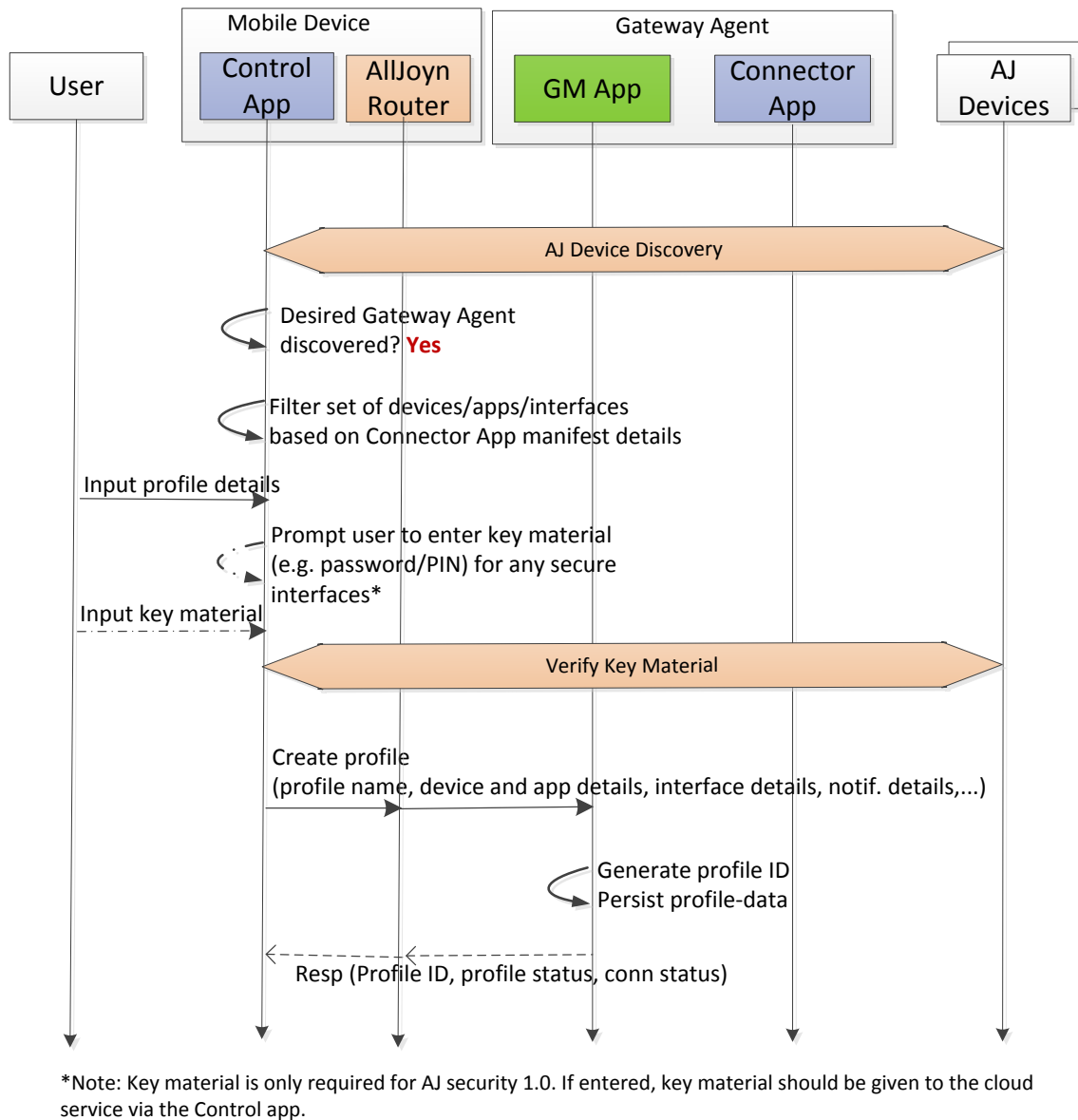


Figure 2-6 Create profile

2.4.2.2 Activate profile

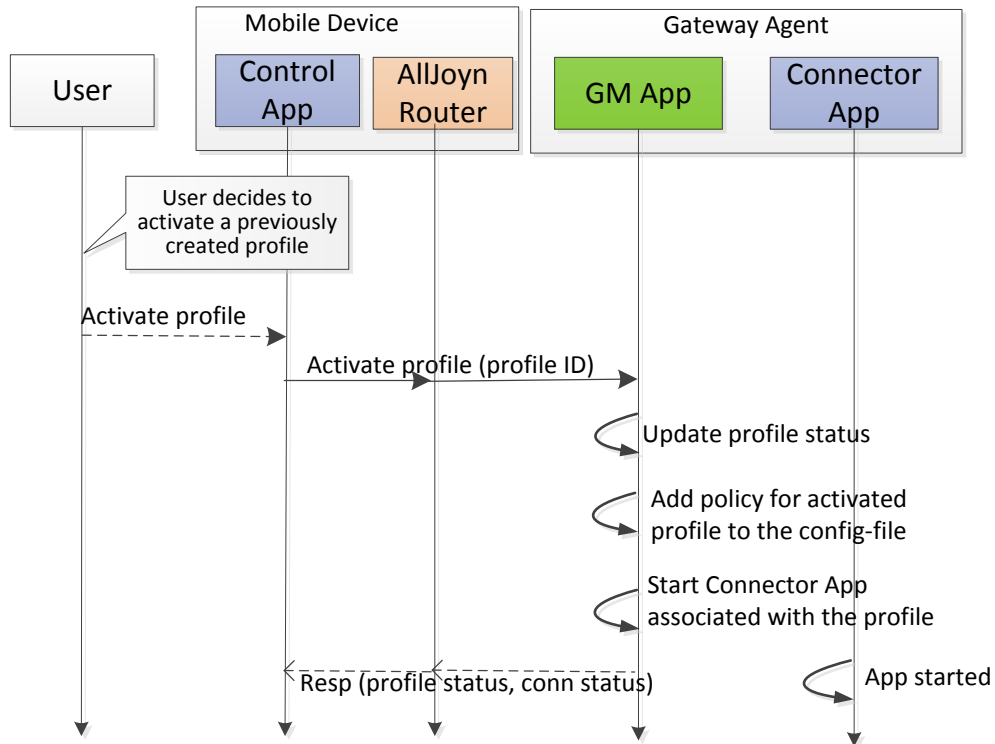


Figure 2-7 Activate profile

2.4.2.3 Connector app startup

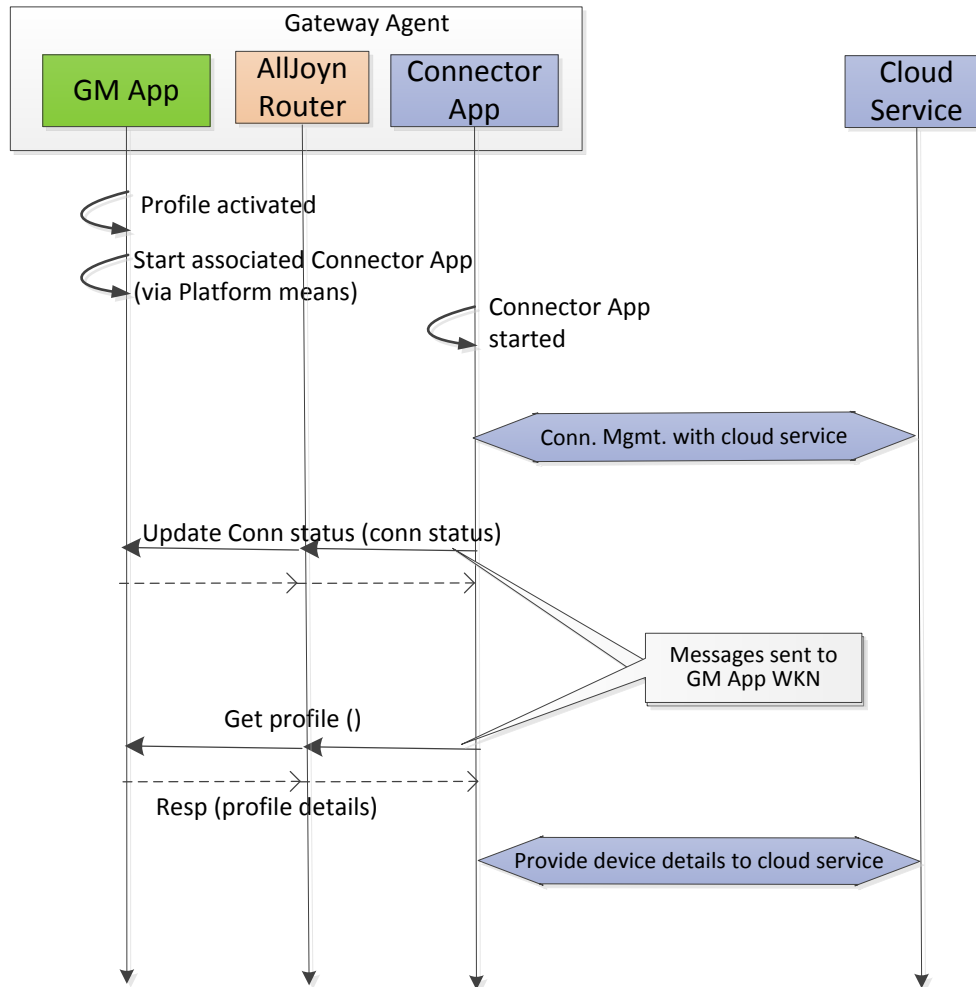


Figure 2-8 Connector app startup

2.4.2.4 Update profile

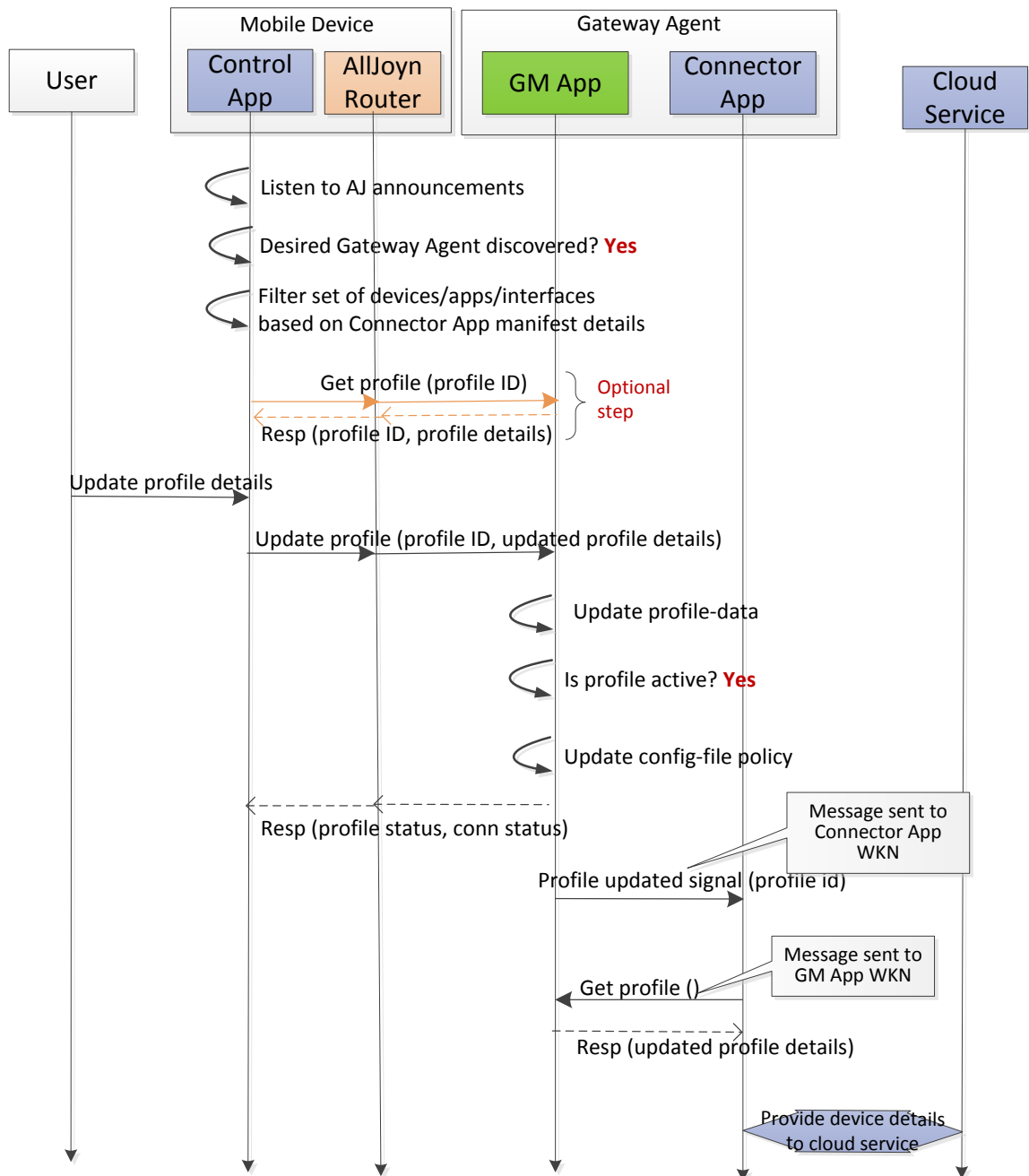


Figure 2-9 Update profile

2.4.2.5 Delete profile

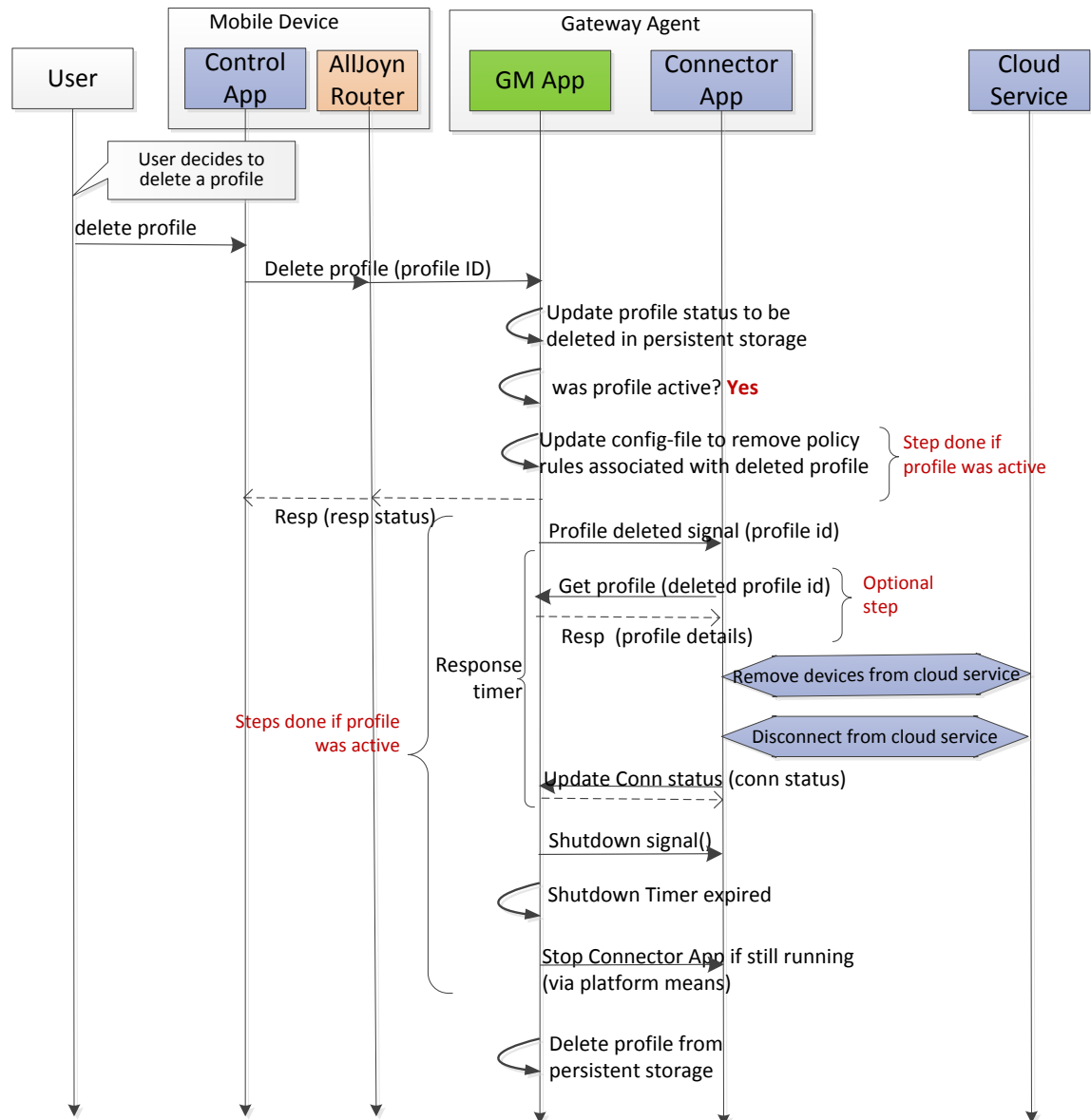


Figure 2-10 Delete profile

2.4.2.6 Delete all profiles

If a user wants to delete all the profiles on the Gateway Management app, the Control app must invoke delete profile on each of the existing profiles. It can get the list of profile ids from the Gateway Management app by invoking the Get profile list function.

2.4.2.7 Deactivate profile

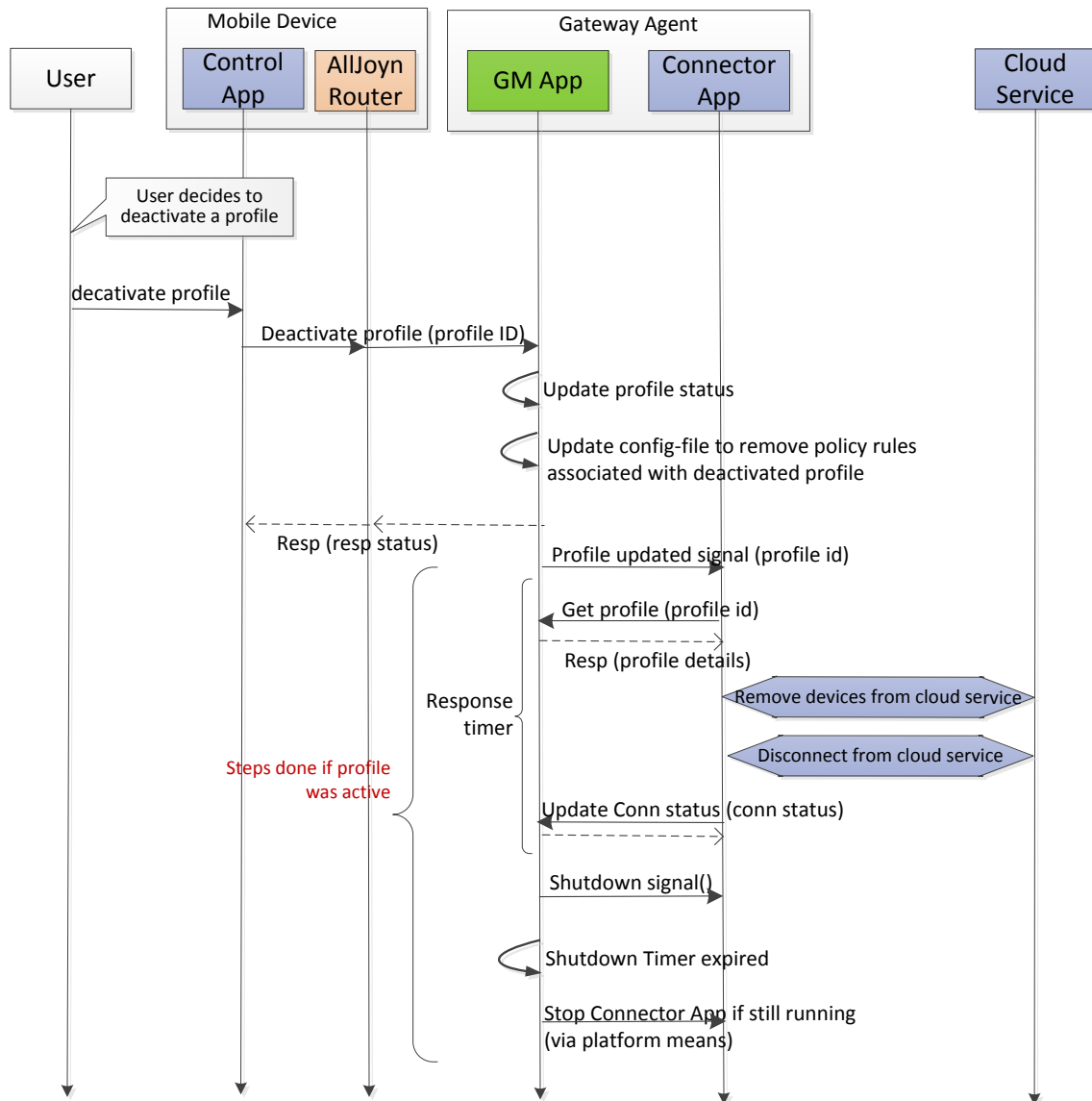


Figure 2-11 Deactivate profile

2.4.2.8 Deactivate all profiles

If a user wants to deactivate all active profiles on the Gateway Management app, the Control app must invoke deactivate profile on each of the existing active profiles. It can get the list of profile ids and their status from the Gateway Management app by invoking the Get profile list function.

2.4.3 Notification

NOTE

The dismissal signal from Notification 1.1 is not removed.

2.4.3.1 Outgoing notification

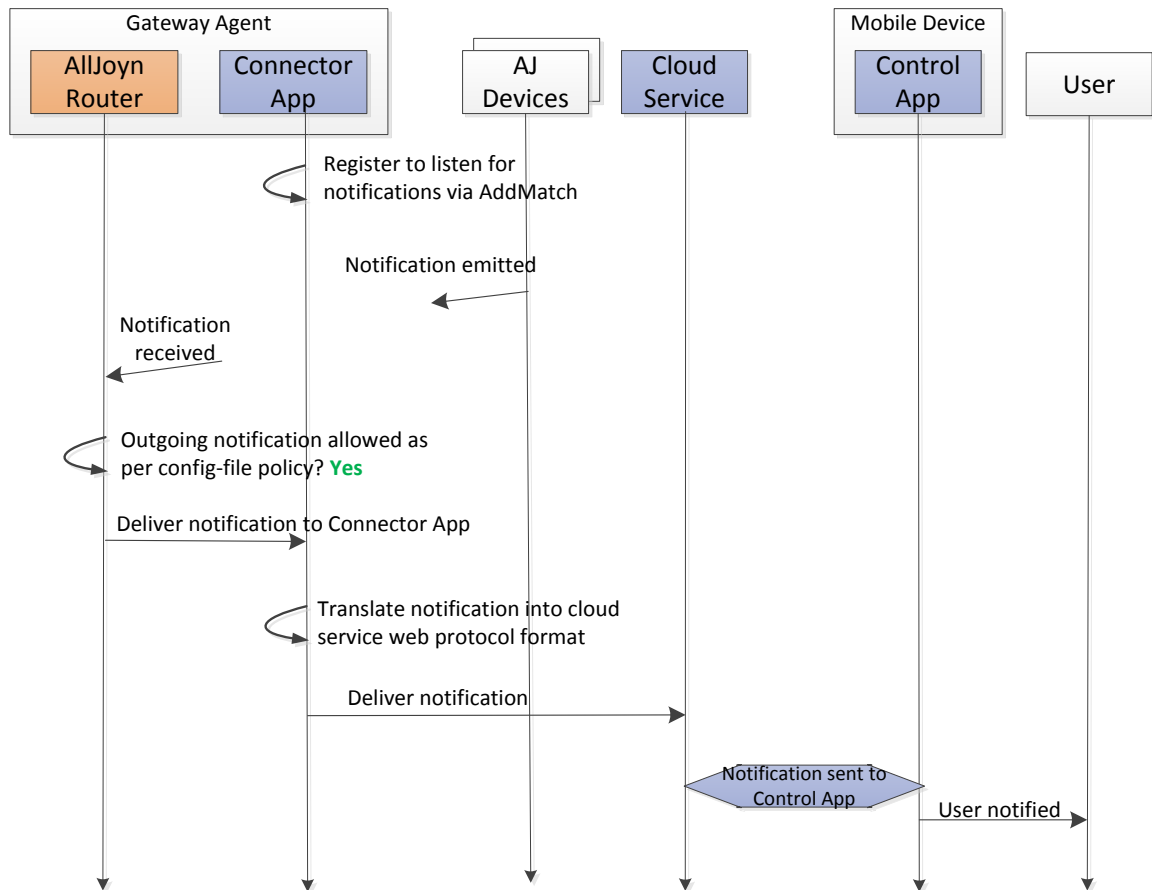


Figure 2-12 Outgoing notification

2.4.3.2 Incoming notification

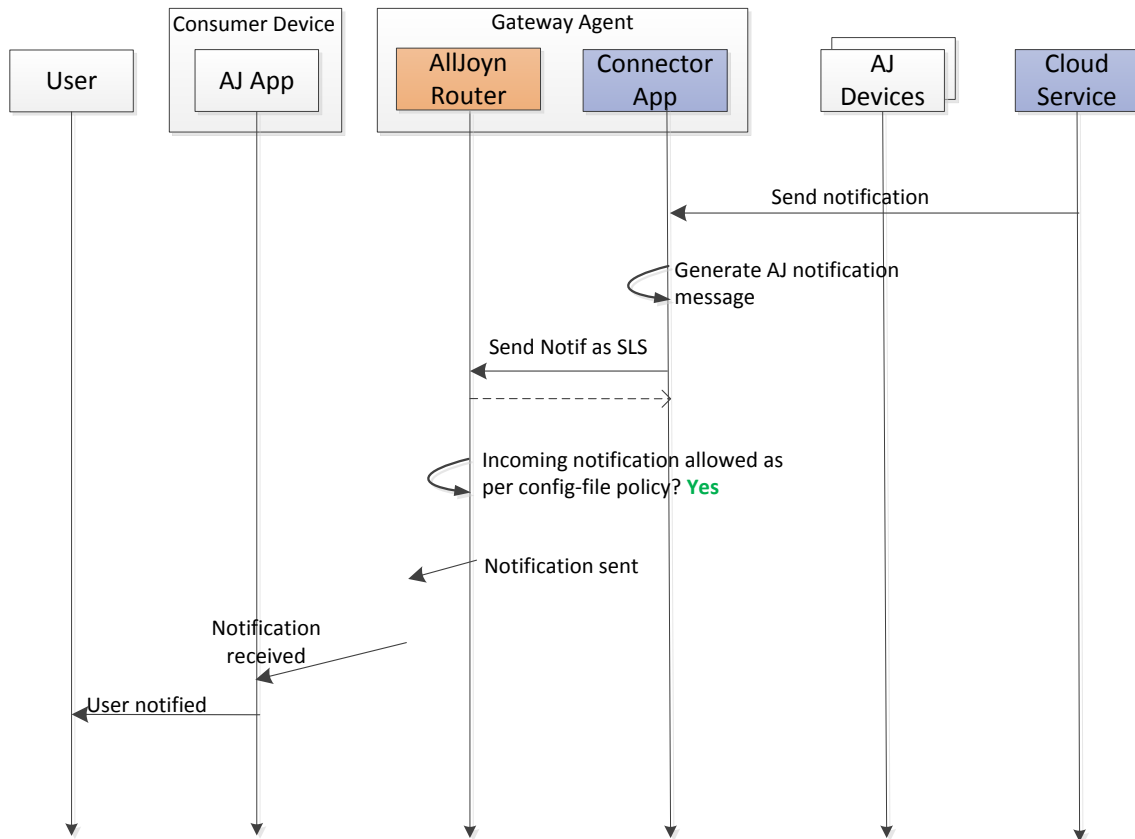


Figure 2-13 Incoming notification

2.4.4 AllJoyn device access

2.4.4.1 Method invocation

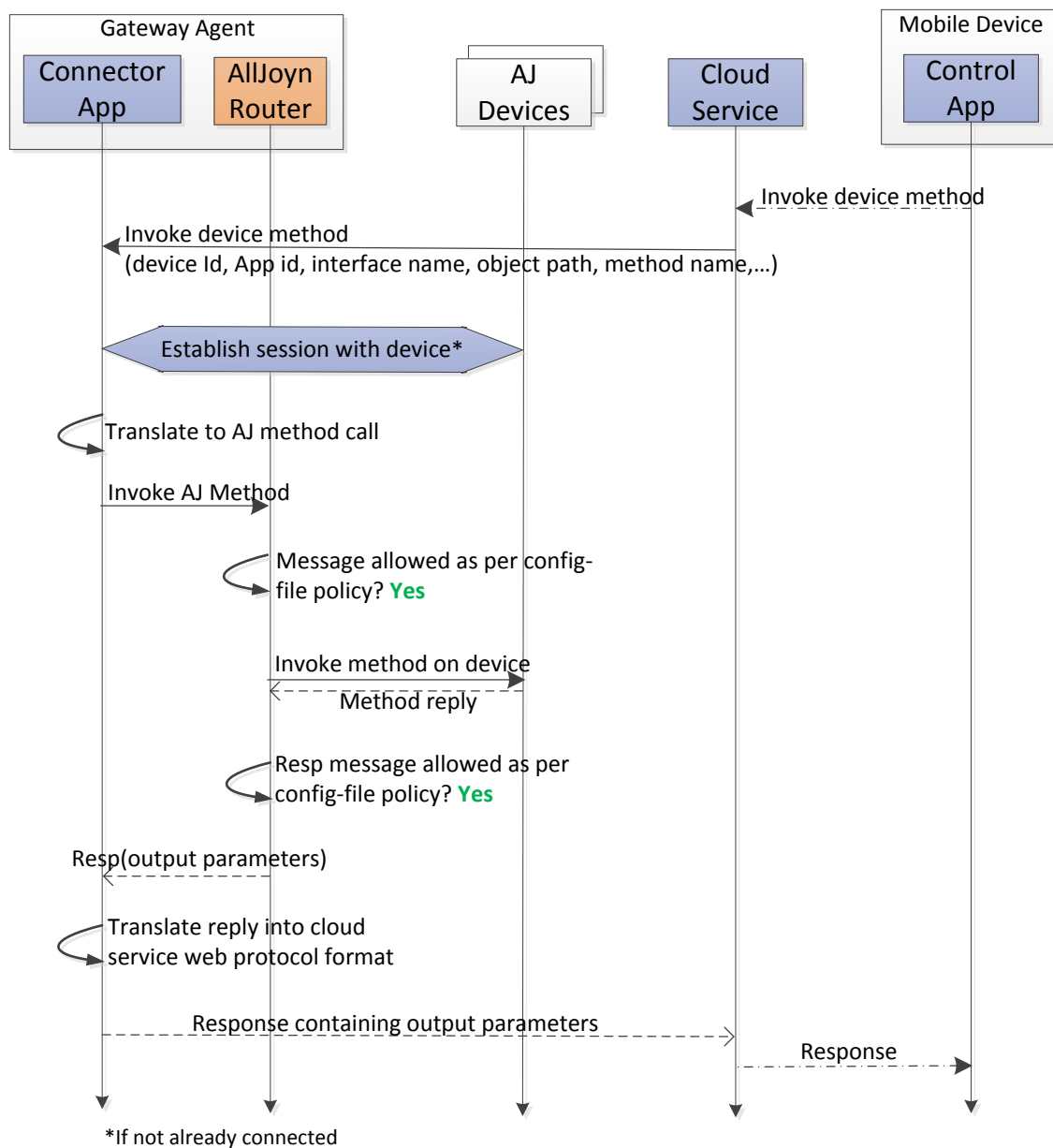


Figure 2-14 Method invocation

2.4.4.2 Get property

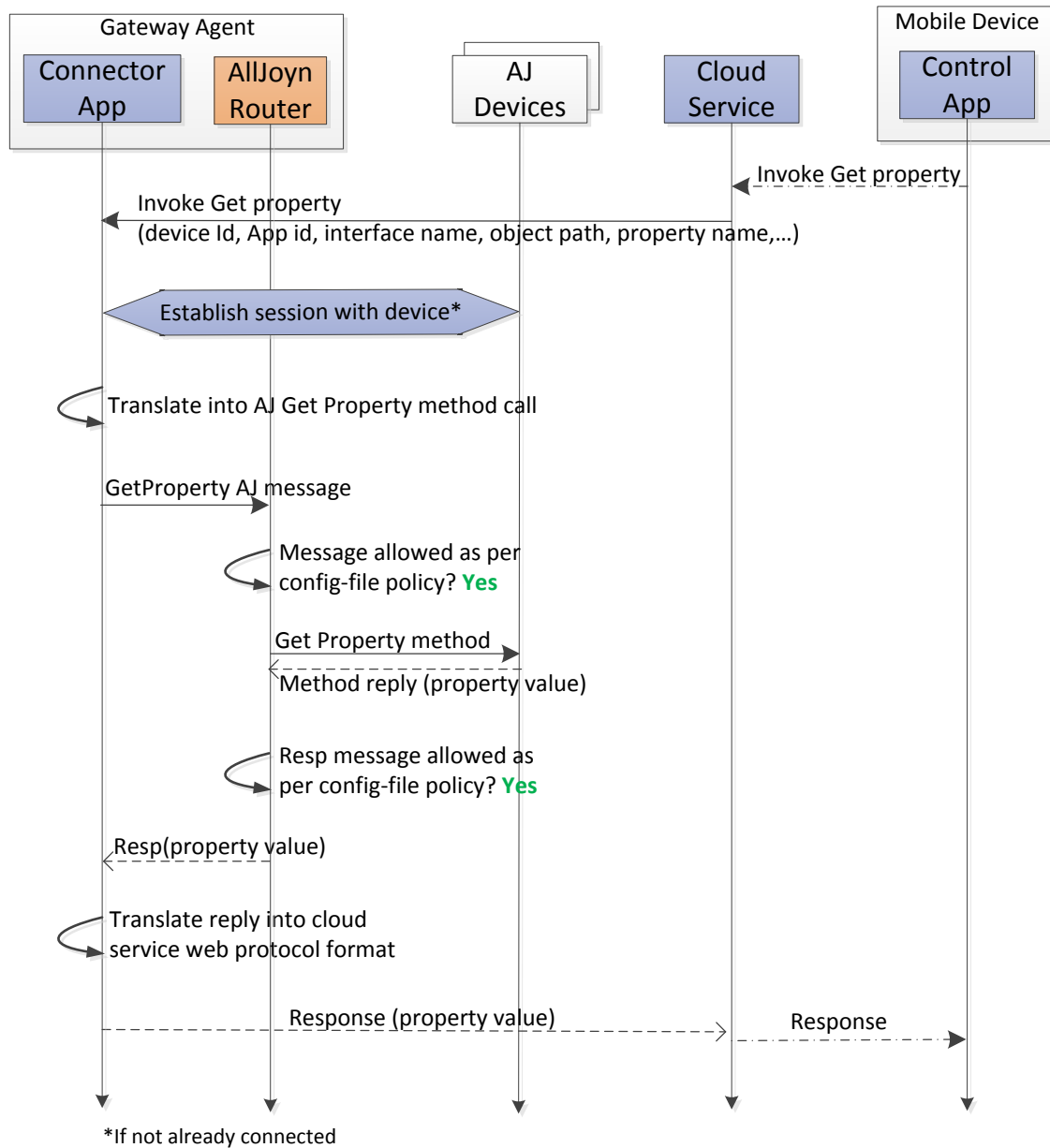


Figure 2-15 Get property

2.4.4.3 Set property

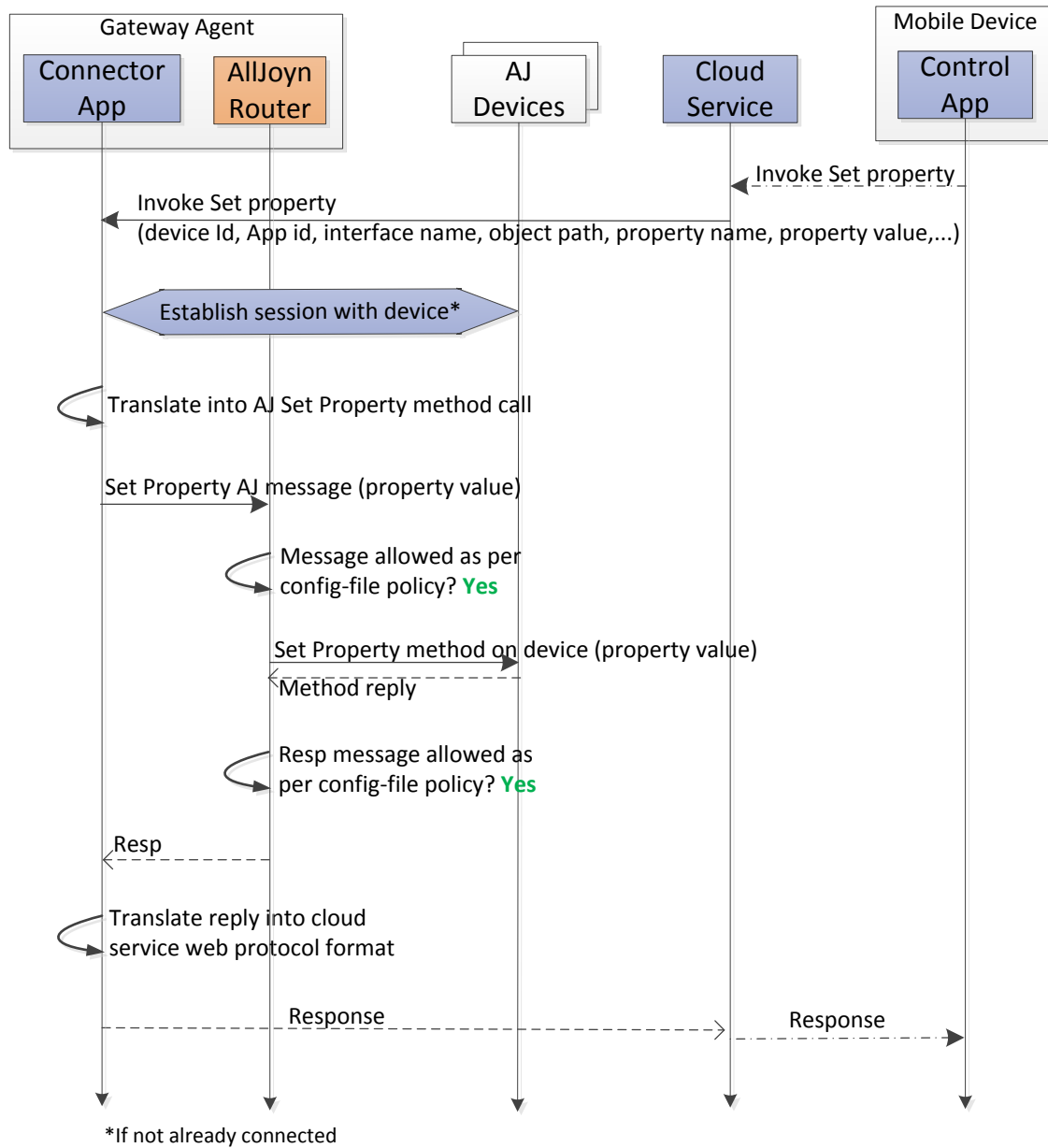


Figure 2-16 Set property

2.4.4.4 Remoting session-based signal

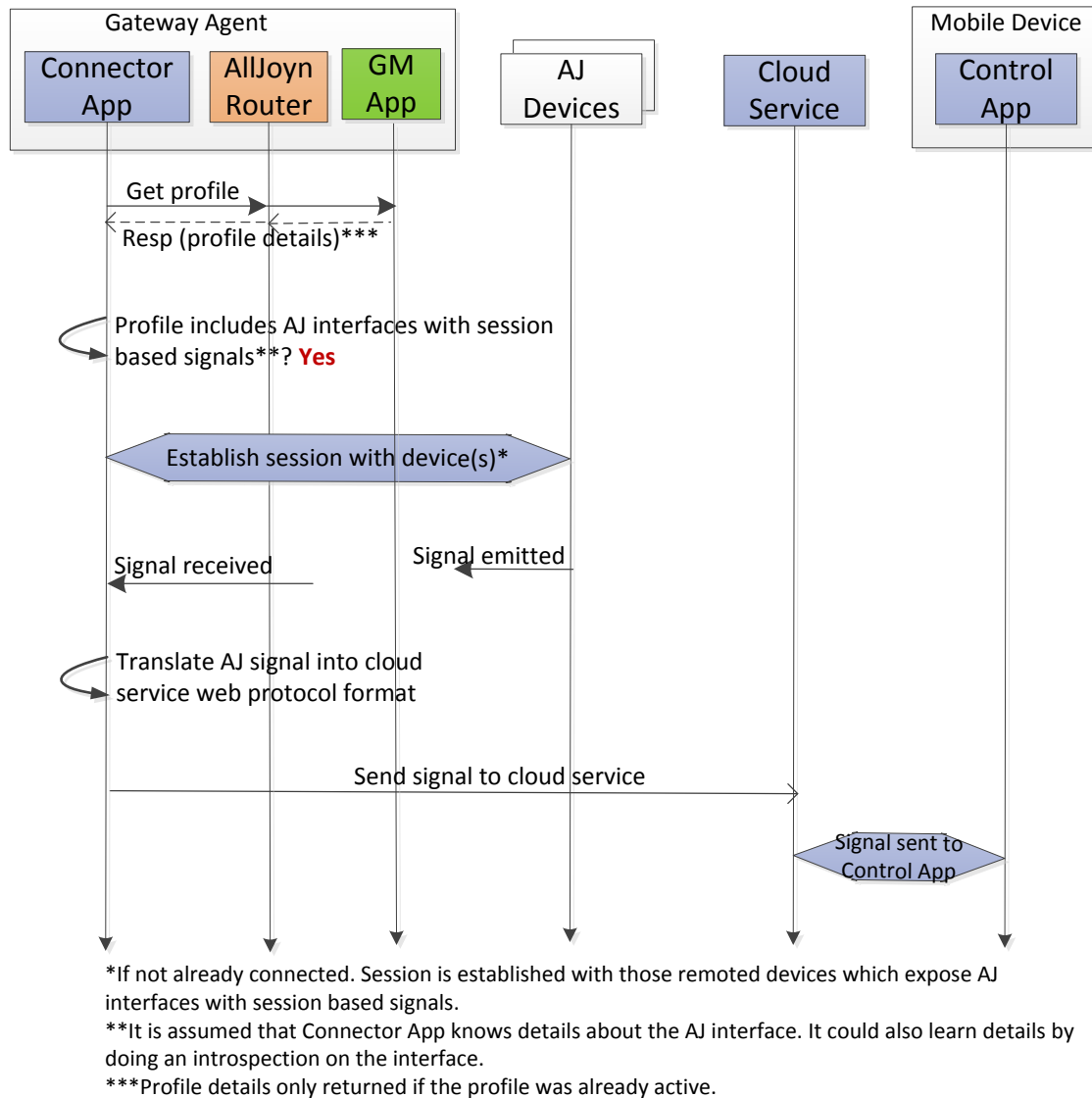


Figure 2-17 Remoting session-based signals

2.5 Functional design

2.5.1 Profile management interface

The Profile management interface is an AllJoyn interface that supports the profile management methods listed in Table 2-1.

Table 2-1 Profile management interface functions

Functional method	Description
Create profile	Creates a new service provider profile on the Gateway Management app. After creation, profile remains in an inactive state until explicitly activated.
Activate profile	Activates a currently inactive remote profile on the Gateway Management app. Activating a profile results in adding related access policy rules in the AllJoyn Router config file and starting the associated Connector app.
Get profile	Fetches information for an existing remote profile from the Gateway Management app.
GetProfileStatus	Returns status for a given profile.
GetProfileList	Fetches list of one or more remote profiles associated with a given Connector app.
Update profile	Updates an existing remote profile on the Gateway Management app. Update takes effect immediately for an already active profile. Updates to an already active profile results in appropriate config file policy changes as needed.
Delete profile	Deletes an existing remote profile on the Gateway Management app. Both active and inactive profiles can be deleted. Deleting a profile results in removing access policy rules for that profile from the config file and stopping of the associated Connector app. Deleted profile is removed from the persistent storage.
Deactivate profile	Deactivates a currently active remote profile on the Gateway Management app. Connector app Deactivating a profile results in removing access policy rules for that profile from the config file and stopping of the associated Connector app.

2.5.1.1 Create profile

Table 2-2 provides a functional definition for the Create profile method.

NOTE

The Gateway app will also store device apps unique name (learned based on receiving announcements from these apps) as part of the profile data. This is required to create access policy rules for device apps in the config file. For a Connector app, the associated user id is used when creating config file policy rules.

Table 2-2 Create profile method

Method	Parameters		
	Parameter Name	In/Out	Description
Create profile	Profile name	In	Name given to the service provider profile (by the user).
	Connector App Name	In	Application name for the Connector app associated with the profile.
	Connector App Id	In	Application ID for the Connector app associated with the profile.
	List <Exposed services>		List of AllJoyn services provided by the Connector app that are being exposed for access by devices on the proximal network.

Method	Parameters		
	Object path	In	Object path for the object implementing the listed interfaces. A wild card "*" means listed interfaces can be accessed at any object path.
	isPrefix	In	Indicates whether the Object path parameter is a prefix to an object path or a complete object path.
	List <Interface name>	In	List of exposed AllJoyn interfaces by the Connector app. A wild card "*" means all interfaces implemented by the object path are exposed.
	List <Remoted Apps>		Profile includes information on the list of AllJoyn apps that are being remoted (allowed for remote access) as part of this profile.
	Device Id	In	Unique device Id from the About feature.
	App Id	In	Unique app Id from the About feature.
	List <Interface info>		For each app, profile specifies list of AllJoyn interfaces that are being remoted.
	Object path	In	Object path for the object implementing the listed interfaces. A wild card "*" means listed interfaces can be accessed at any object path.
	isPrefix	In	Indicates whether the Object path parameter is a prefix to an object path or a complete object path.
	List <Interface name>	In	List of remoted AllJoyn interfaces. A wild card "*" means all interfaces implemented by the object path are remoted.
	List <Metadata>	In (optional)	Specifies a set of metadata for the remoted app.
	Key	In	Key describing the metadata.
	Value	In	Value of the metadata.
	Profile Id	Out	A unique identifier generated by the Gateway Management app for the remote profile.
	Profile Status	Out	Indicates current status for the profile. Set as per Table 2-16. This should be set to inactive in the response to the Create profile method.

2.5.1.2 Activate profile

Table 2-3 provides a functional definition for the Activate profile method. The Connector app associated with the profile is started when that profile gets activated. After startup, the Connector app attempts to establish a connection with the cloud service.

Table 2-3 Activate profile method

Method	Parameters		
	Parameter Name	In/Out	Description
Activate profile	Profile Id	In	Unique identifier for the remote profile to be activated.
	Profile Status	Out	Indicates current status for the profile. Set as per Table 2-16. Should be set to 'active' in the Activate profile method response.

2.5.1.3 Get profile

Table 2-4 provides a functional definition for the Get profile method.

Table 2-4 Get profile method

Method	Parameters		
	Parameter Name	In/Out	Description
Get profile	Profile Id	In	Unique identifier for the remote profile to be fetched.
	Profile name	Out	Name given to the service provider profile.
	Connector App name	Out	Connector app name associated with the profile.
	Connector App Id	Out	Application ID for the Connector app associated with the profile (retrieved from Announcement).
	List <Exposed services>		List of AllJoyn services provided by the Connector app that are being exposed for access on the proximal network.
	Object path	In	Object path for the object implementing the listed interfaces. A wild card "*" means listed interfaces can be accessed at any object path.
	isPrefix	In	Indicates whether the Object path parameter is a prefix to an object path or a complete object path.
	List <Interface name>	In	List of exposed AllJoyn interfaces by the Connector app. A wild card "*" means all interfaces implemented by object path are exposed.
	List <Remoted Apps>		Profile includes information on the list of AllJoyn apps that are being remoted as part of this profile.
	Device Id	Out	Device Id from the About feature.
	App Id	Out	App Id from the About feature.
	List <Interface info>		For each app, profile specifies list of AllJoyn interfaces that are being remoted.

Method	Parameters		
	Object path	Out	Object path for the object implementing the listed interfaces. A wild card "*" means listed interfaces can be accessed at any object path.
	isPrefix	In	Indicates whether the Object path parameter is a prefix to an object path or a complete object path.
	List <Interface name>	In	List of remotored AllJoyn interfaces. A wild card "*" means all interfaces implemented by the object path are remotored.
	List <Metadata>	In (optional)	Specifies a set of metadata for the remotored app.
	Key	In	Key describing the metadata.
	Value	In	Value of the metadata.
	Profile Status	Out	Indicates current status for the profile. Set as per Table 2-16.

2.5.1.4 Get profile status

Table 2-5 provides a functional definition for the Get profile status method.

Table 2-5 Get profile status method

Method	Parameters		
	Parameter Name	In/Out	Description
Get profile status	Profile Id	In	Unique identifier for the remote profile.
	Profile name	Out	Name assigned to the profile.
	Profile status	Out	Indicates current status for the profile. Set as per Table 2-16.

2.5.1.5 Get profile list

Table 2-6 provides a functional definition for the Get profile list method.

Table 2-6 Get profile list method

Method	Parameters		
	Parameter Name	In/Out	Description
Get profile list	Connector App Id	In	Identifier for the Connector app for which the profile list is being fetched.
	List <Profile Info>		Provides list of profiles associated with the specified Connector app.
	Profile Id	Out	Identifier for the profile
	Profile Name	Out	Name assigned to the profile.
	Profile status	Out	Indicates current status for the profile. Set as per Table 2-16.

2.5.1.6 Update profile

Table 2-7 provides a functional definition for the Update profile method. This call provides a complete set of updated profile information.

Table 2-7 Update profile method

Method	Parameters		
	Parameter Name	In/Out	Description
Update profile	Profile Id	In	Unique identifier for the service provider profile to be updated
	Profile name	In	Name given to the service provider profile
	Connector App name	In	Connector app name associated with the profile
	Connector App Id	In	Application ID for the Connector app associated with the profile
	List <Exposed services>		List of AllJoyn services provided by the Connector app that are being exposed for access on the proximal network.
	Object path	In	Object path for the object implementing the listed interfaces. A wild card "*" means listed interfaces can be accessed at any object path.
	isPrefix	In	Indicates whether the Object path parameter is a prefix to an object path or a complete object path.
	List <Interface name>	In	List of exposed AllJoyn interfaces by the Connector app. A wild card "*" means all interfaces implemented by object path are exposed.
	List <Remoted Apps>		Profile includes information on the list of AllJoyn apps that are being remoted as part of this profile.
	Device Id	In	Unique device Id from the About feature.
	App Id	In	Unique app Id from the About feature.
	List <Interface info>		For each app, profile specifies list of AllJoyn interfaces that are being remoted.
	Object path	In	Object path for the object implementing the listed interfaces. A wild card "*" means listed interfaces can be accessed at any object path.
	isPrefix	In	Indicates whether the Object path parameter is a prefix to an object path or a complete object path.
	List <Interface name>	In	List of remoted AllJoyn interfaces. A wild card "*" means all interfaces implemented by the object path are remoted.

Method	Parameters		
	List <Metadata>	In (optional)	Specifies a set of metadata for the remoted app.
	Key	In	Key describing the metadata.
	Value	In	Value of the metadata.
	Profile Status	Out	Indicates current status for the profile. Set as per Table 2-16.

2.5.1.7 Delete profile

Table 2-8 provides a functional definition for the Delete profile method.

Table 2-8 Delete profile method

Method	Parameters		
	Parameter Name	In/Out	Description
Delete profile	Profile Id	In	Unique identifier for the service provider profile to be deleted.
	Resp status	Out	Indicates success/failure status for the profile deletion.

2.5.1.8 Deactivate profile

Table 2-9 provides functional definition for the Deactivate profile method.

Table 2-9 Deactivate profile method

Method	Parameters		
	Parameter Name	In/Out	Description
Deactivate profile	Profile Id	In	Unique identifier for the service provider profile to be deleted.
	Profile Status	Out	Indicates current status for the profile. Should be set to 'inactive' in the Deactivate profile response.

2.5.2 App Access interface

The Connector app and Gateway Management app communicate with each other via the AllJoyn App Access interface. No AllJoyn session is needed between the two apps for communication because both apps are connected to the same AllJoyn Router. WKNs are used for communicating between two apps:

- Gateway Management app WKN: org.alljoyn.GWAgent.GMApp
- Connector app WKN: org.alljoyn.GWAgent.Connector.<Connector app id>

The App Access interface allows a Connector app to fetch profile data and provide connection status update to the Gateway Management app. The Gateway Management app uses this interface to notify the Connector app about profile updates and deletion.

The Gateway Management app also supports a shutdown signal as part of this interface to trigger a graceful shutdown of a Connector app.

The App Access interface supports functional methods/signals as summarized in Table 2-10.

Table 2-10 App Access interface functions

Functional method	Description
Get profile	Returns information for an existing active remote profile associated with the Connector app asking for the profile data. If there was no active profile, an error response should be returned.
Update connection status	Invoked by the Connector app to update its cloud service connection status at the Gateway Management app. This method requires no response to be sent.
Functional Signal	Description
Profile updated	Provides an indication that a specific profile associated with the Connector app has been updated. Sent by the Gateway Management app whenever an active profile is updated. This is a unicast signal delivered only to the associated Connector app.
Profile deleted	Provides an indication that a specific profile associated with the Connector app has been deleted. Sent by the Gateway Management app whenever an active profile is deleted. This is a unicast signal delivered only to the associated Connector app.
Shutdown App	Provides a trigger to the Connector app to perform a graceful shutdown.

2.5.2.1 Get profile

Table 2-11 provides a functional definition for the Get profile method. The Connector app can request a specific profile (by providing the profile Id), or it can request a set of all profiles associated with that Connector app. This method should return only active profiles associated with the Connector app. If the profile being requested is not active (in the case when a profile id is provided) or there is no active profile associated with the Connector app, an error response should be returned.

The Gateway Management app will maintain profiles for each Connector app as part of separate service objects. As explained earlier, this is for the security reasons so as to ensure that each Connector app can only access its own profile data. These service objects have a well-known object path of the format /Profiles/<Connector app id>. When sending the Get profile message to the Gateway app, the Connector app should use the well-known object path.

Table 2-11 Get profile method

Method	Parameters		
	Parameter Name	In/Out	Description
Get profile	Profile Id (optional)	In	Unique identifier for the service provider profile to fetched.
	List<Profile Info>		Response includes details on one or more active profiles associated with the Connector app.

Method	Parameters		
	Profile name	Out	Name given to the service provider profile.
	Connector App name	Out	Connector app name associated with the profile.
	Connector App Id	Out	Application ID for the Connector app associated with the profile (retrieved from Announcement).
	List <Exposed services>		List of AllJoyn services provided by the Connector app that are being exposed for access on the proximal network.
	Object path	Out	Object path for the object implementing the listed interfaces. A wild card "*" means listed interfaces can be accessed at any object path.
	isPrefix	In	Indicates whether the Object path parameter is a prefix to an object path or a complete object path.
	List <Interface name>	In	List of exposed AllJoyn interfaces by the Connector app. A wild card "*" means all interfaces implemented by object path are exposed.
	List <Remoted Apps>		Profile includes information on the list of AllJoyn apps that are being remoted as part of this profile.
	Device Id	Out	Device Id from the About feature.
	App Id	Out	App Id from the About feature.
	List <Interface info>		For each app, profile specifies list of AllJoyn interfaces that are being remoted.
	Object path	Out	Object path for the object implementing the listed interfaces. A wild card "*" means listed interfaces can be accessed at any object path.
	isPrefix	In	Indicates whether the Object path parameter is a prefix to an object path or a complete object path.
	List <Interface name>	In	List of remoted AllJoyn interfaces. A wild card "*" means all interfaces implemented by the object path are remoted.
	List <Metadata>	In (optional)	Specifies a set of metadata for the remoted app.
	Key	In	Key describing the metadata.
	Value	In	Value of the metadata.
	Profile Status	Out	Indicates current status for the profile. Set as per Table 2-16.

2.5.2.2 Update connection status

Table 2-12 provides a functional definition for the update connection status method.

Table 2-12 Update connection status method

Method	Parameters		
	Parameter Name	In/Out	Description
Update connection status	Connector App Id	In	Connector app Id for which the connection status with the cloud service is being updated.
	Connection Status	In	Indicates the connection status with the cloud service for the specified Connector app. Set as per Table 3-15.

2.5.2.3 Profile updated signal

Table 2-13 provides a functional definition for the profile updated signal.

Table 2-13 Profile updated signal

Signal	Parameters		
	Parameter Name	In/Out	Description
Profile Updated	Connector App Id	In	Connector app Id associated with the profile.
	Profile Id	In	Profile Id that has been updated.

2.5.2.4 Profile deleted signal

Table 2-14 provides a functional definition for the profile deleted signal.

Table 2-14 Profile deleted signal

Signal	Parameters		
	Parameter Name	In/Out	Description
Profile Deleted	Connector App Id	In	Connector app Id associated with the profile.
	Profile Id	In	Profile Id that has been deleted.

2.5.2.5 Shutdown app signal

Table 2-15 provides a functional definition for the shutdown signal sent to the Connector app for a graceful shutdown.

Table 2-15 Shutdown app signal

Signal	Parameters		
	Parameter Name	In/Out	Description
Shutdown App	Connector App Id	In	App Id for the Connector app that is being shutdown.

2.5.3 Fields

2.5.3.1 Profile status

Table 2-16 Profile status

Value	Description
Inactive	Profile is not active.
Active	Profile is active.
To be deleted	Profile is marked to be deleted soon.

2.5.4 Config file policy enforcement

Access to remoted device interfaces is controlled by the AllJoyn Router enforcing policy rules written in the config file by the Gateway Management app. After a profile is activated, the Gateway Management app updates the config file to allow communication between the Connector app and remoted devices. The Gateway Management app is also responsible for removing allow policy rules from the config file when a profile is deactivated to stop remote access to device interfaces.

For creating allows rules for a profile, the Gateway Management app makes use of user IDs associated with Connector apps, and unique names for AllJoyn apps that are being remoted. The Gateway Management app should update the config file policy to accomplish the following:

1. The policy should allow the Gateway Management app to receive announcements from all device apps in the proximal network. This will be achieved by default.
2. At startup, the Gateway Management app should add a deny-all rule for all the preinstalled Connector apps based on their user id.
3. After a new Connector app is installed, the Gateway Management app should add a deny-all rule for that Connector app based on its user id.
4. After a profile is activated, the Gateway Management app should update the config file to add allow policy rules as captured in Table 2-17.
5. After an active profile is deleted or deactivated, the Gateway Management app should update the config file to remove all the allow policy rules related to that profile.

Table 2-17 Config file allow rules

Config file rules added after profile activation	
1.	Allow Connector app to access Profile Retrieval Interface from the Gateway Management app only at the service object that is maintaining profile for that Connector app.
2.	Allow Connector app to send out its announcement (if any) over the proximal network.
3.	Allow Connector app to send out its announcement (if any) over the proximal network.
4.	Allow Connector app to invoke method calls and receive method replies from the devices/apps/interfaces/object paths that have been remototed in the profile.
5.	Allow Connector app to receive signals from the devices/apps/interfaces/object paths that have been remototed in the profile.
6.	Allow Connector app to receive notifications from the remototed device apps for which the outgoing notifications are enabled in the profile. Note that there is a separate notification object path for each notification priority level.
7.	Allow Connector app to send notifications in the proximal network for those incoming notification types that are configured as part of the profile.
8.	Allow Connector app to receive method calls and send method replies for Connector app interfaces exposed in the profile.
9.	Allow Connector app to send signals for Connector app interfaces exposed in the profile.
10.	Allow Connector app to introspect any of the remototed interfaces.

Exact syntax for these config file policy rules will be defined during the implementation phase.

2.5.4.1 Unique name retrieval

The Gateway Management app needs to know the unique name for device apps to create allow policy rules in the config file for devices/apps in active profiles. The Gateway Management app should start listening for announcements after the very first profile is activated, and continue to receive announcements after that. Since there is no way to re-fetch announcements, the Gateway Management app must cache all the announcements it receives and use the associated unique names when creating an allow policy for AllJoyn apps.

Device apps can also get a new unique name assigned in various scenarios such as when the app is restarted. In this case, a new announcement will be received by the Gateway Management app with the new unique name. The Gateway Management app will use (device Id, App Id) pair to determine an updated announcement from a given app and figure out if the unique name has changed. If yes, the Gateway Management app updates the allow policy rules in the config file to reflect the new unique name. The Gateway Management app will then send a trigger to the AllJoyn Router to load the updated config file.

2.5.4.2 Config file reload

After updating the config file, the Gateway Management app notifies the AllJoyn Router so that it can reload the updated config file. The following two methods will be supported by the AllJoyn Router to receive trigger for config file updates:

- **Platform-based signal:** This method uses device platform signaling (e.g., OpenWRT kill signaling) to send a signal to the AllJoyn Router that results in the reloading of the config file.
- **AllJoyn Method trigger:** An AllJoyn method call can be invoked at the AllJoyn Router to trigger reloading of config file. Details on which interface supports this method call will be defined during the implementation phase.

The Gateway Management app should invoke the AllJoyn method to trigger reloading of config file at the AllJoyn Router.

2.6 Versioning

Versioning for AllJoyn interfaces should be done using a version property. Each of the following AllJoyn interfaces supported by the Gateway Management app should be versioned independently:

- Profile Management interface
- App Access interface
- App Management interface

The current version for AllJoyn gateway related interfaces is '1'.

3 App Management

3.1 Architecture

Figure 3-1 shows the Gateway Agent architecture for Connector app management functionality.

- An App Download server (likely from a service provider) hosts Connector app packages.
- The Control app provides a UI for initiating the app install/upgrade and provides download URL to the Gateway Management app for downloading app packages from the App Download server.
- The Gateway Management app exposes an App Management interface to enable the Control app to manage Connector apps on the Gateway Agent including install, upgrade, uninstall and restart of the apps.

Due to security concerns, it has been decided to not use OpenWRT opkg package manager for installing Connector apps. A separate Package Manager (PM) application is supported on the Gateway Agent for installing/upgrading/uninstalling Connector apps. The Packet Manager app performs the download of the app package and verifies the signature before installing the package. The exact nature of the Package Manager app (AllJoyn vs. nonAllJoyn app) is considered implementation detail and is outside of this HLD scope.

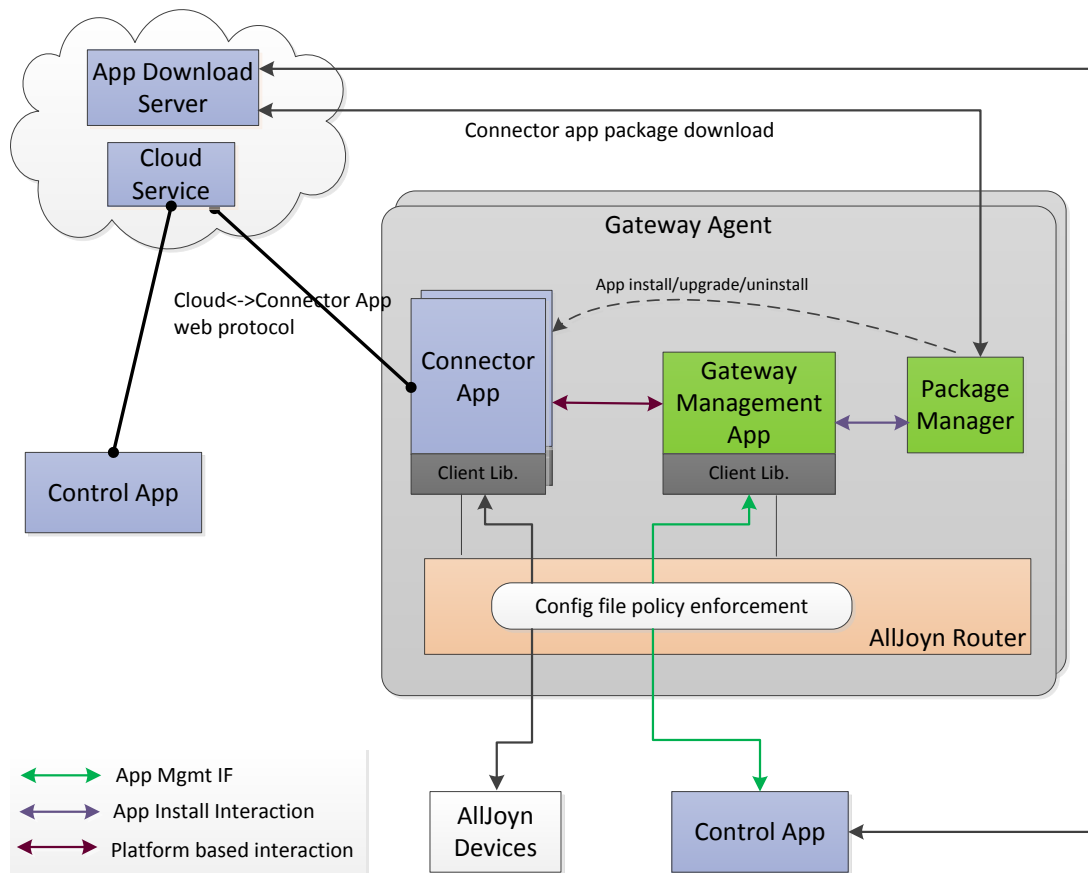


Figure 3-1 Gateway Agent app management architecture

The Gateway Management app and the Package Manager app will be packaged as OpenWRT packages and will be installable using the OpenWRT package manager (opkg). A Gateway Agent device may be packaged only with Gateway Management app and Package Manager app and may not have any Connector apps installed initially. In this case, all Connector apps will get installed via the App Management interface.

3.1.1 App management at the Gateway Management app

The Gateway Management app exposes a secure App Management interface to enable Control app to manage Connector apps on the Gateway Agent. This interface provides install, upgrade and uninstall functionality for Connector apps. It can be used to dynamically install new Connector apps and upgrade already installed Connector apps on the Gateway Agent. The Control app provides the download URL for the app package to the Gateway Management app.

The Gateway Management app interacts with the Package Manager app to perform app package download and installation. Each Connector app gets installed in a separate UNIX user id. The Gateway Management app assigns unique user Id to Connector apps. All Connector apps get assigned the same UNIX group id.

The App Management interface also provides restart functionality for a Connector app. The Control app can retrieve status information for a given Connector app (including install status, operational status, and cloud service connection status) via this interface as well. The App Management interface is advertised in the About announcement signal sent out by the Gateway Management app and is used by the Control app to discover the Gateway Management app.

The Gateway Management app should have a logical app table and maintain an app entry for each Connector app containing the details listed in Table 3-1.

Table 3-1 App details maintained at the Gateway Management app

App Detail	Description
App Id	AllJoyn identifier for the Connector app.
App Name	App Name for the Connector app.
App Package Name	Package name for the Connector app package.
App version	Latest installed version for the Connector app version.
App package File URL	File location for the Connector app package.
App User Id	UNIX user Id associated with the Connector app.
App Group Id	UNIX group Id associated with the Connector app.
Install Status	Provides installation status for the Connector app as per Table 3-16.
Install Description	Provides a description about the results of the last app install/upgrade/uninstall operation done for that Connector app.
Connection Status	Provides cloud service connection status for the Connector as per Table 3-15.
Operational Status	Provides operational status for the Connector app as per Table 3-17.

For a Connector app for which the installation failed, there will be an entry maintained indicating 'install failed' status. Such entries can be purged after some time period. For preinstalled Connector apps, these app details should get populated as part of the offline install procedure.

Every Connector app will have a manifest file proving metadata about the app. The app manifest file should reside in a predefined location inside the app package directory (e.g. root directory of the app package).

3.1.2 Package Manager app

The Package Manager app provides functionality to enable Gateway Management app to install/upgrade/uninstall Connector apps. The Gateway Management app passes the download URL to the Package Manager app that downloads the app package from the App Download server, performs signature verification, and installs the app package in a new user account. For an app upgrade, the upgraded package gets installed with the same user account as associated with the old package. The Package Manager App should support a CLI (command-line-interface) for app installation functionality it provides.

As mentioned above, the exact nature of the Package Manager app (AllJoyn vs. nonAllJoyn app) and communication channel details between the Gateway Management

app and Package Manager App is considered implementation detail and is outside of this HLD scope.

3.1.3 App management at the Control app

The Control app enables user to install, uninstall and upgrade Connector apps on the Gateway Agent. The Control app determines whether new Connector apps or upgrades to already installed Connector apps are available for the Gateway Agent. For this, the Control app can use About data exposed by the Gateway Management app including (Manufacturer, ModelNumber, Platform, PlatformVersion, SoftwareVersion, AllJoynSoftwareVersion and HardwareVersion) to filter apps available for the Gateway Agent device.

NOTE

Only some of these fields are received in the About announcement signal. Others must be retrieved by connecting with the About object exposed by the Gateway Management app.

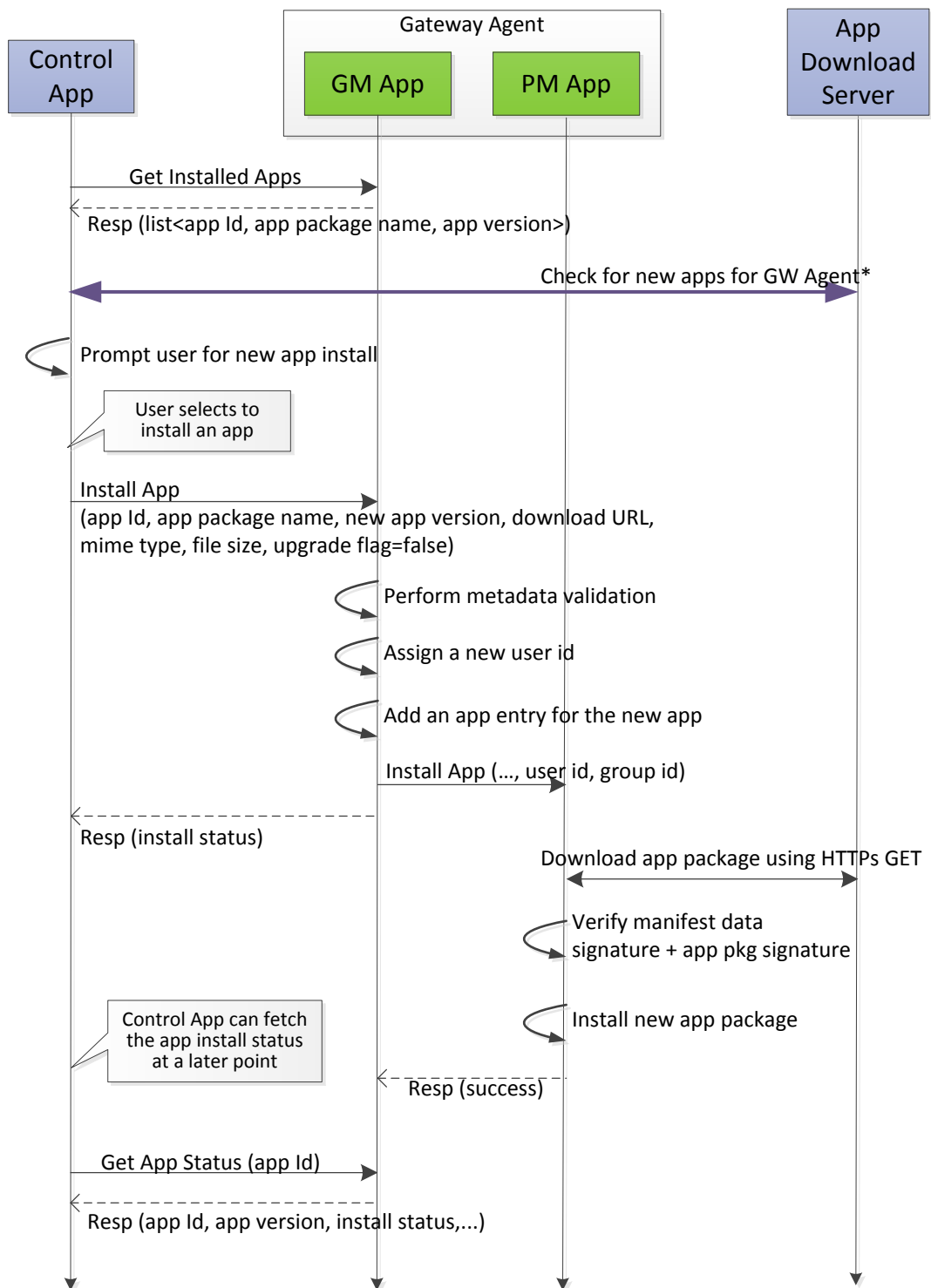
The Control app can also get manifest files for already installed apps from the Gateway Management app and for new apps from the App Download server and use this information to determine apps availability for upgrades. The Control app provides install/upgrade options to the user and invokes App Management interface based on user action.

The Control app also enables a user to restart an app and check latest status for the app including install status, operational status and cloud service connection status.

3.2 Call flows

This section captures likely scenario call flows for various Connector app management use cases.

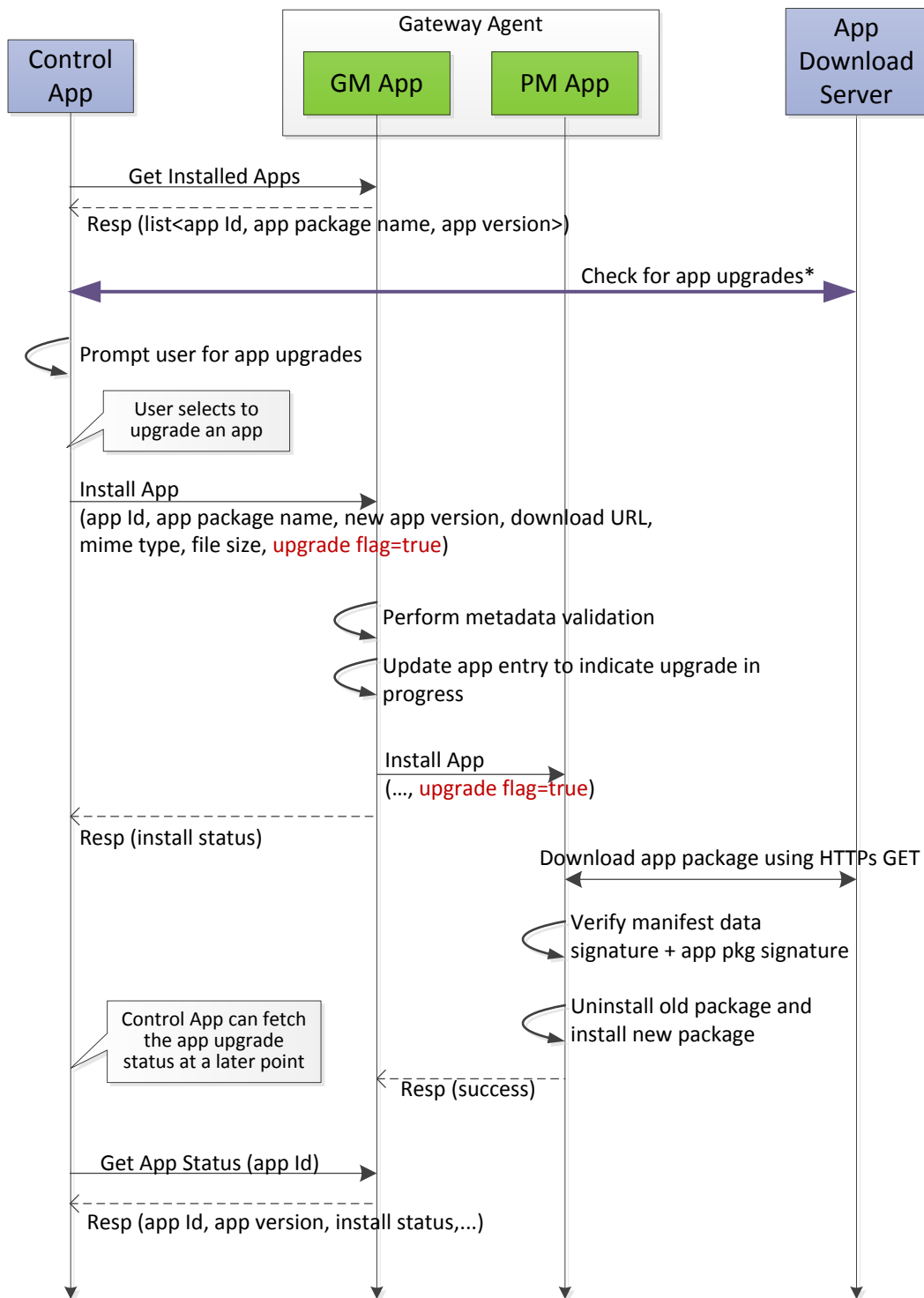
3.2.1 Connector app install



* Control App can use About data fields from GM App to filter available apps for the Gateway Agent device.

Figure 3-2 Connector app install

3.2.2 Connector app upgrade



* Control App can use About data fields from GM App to filter available apps for the Gateway Agent device.

Figure 3-3 Connector app upgrade

3.2.3 Connector app uninstall

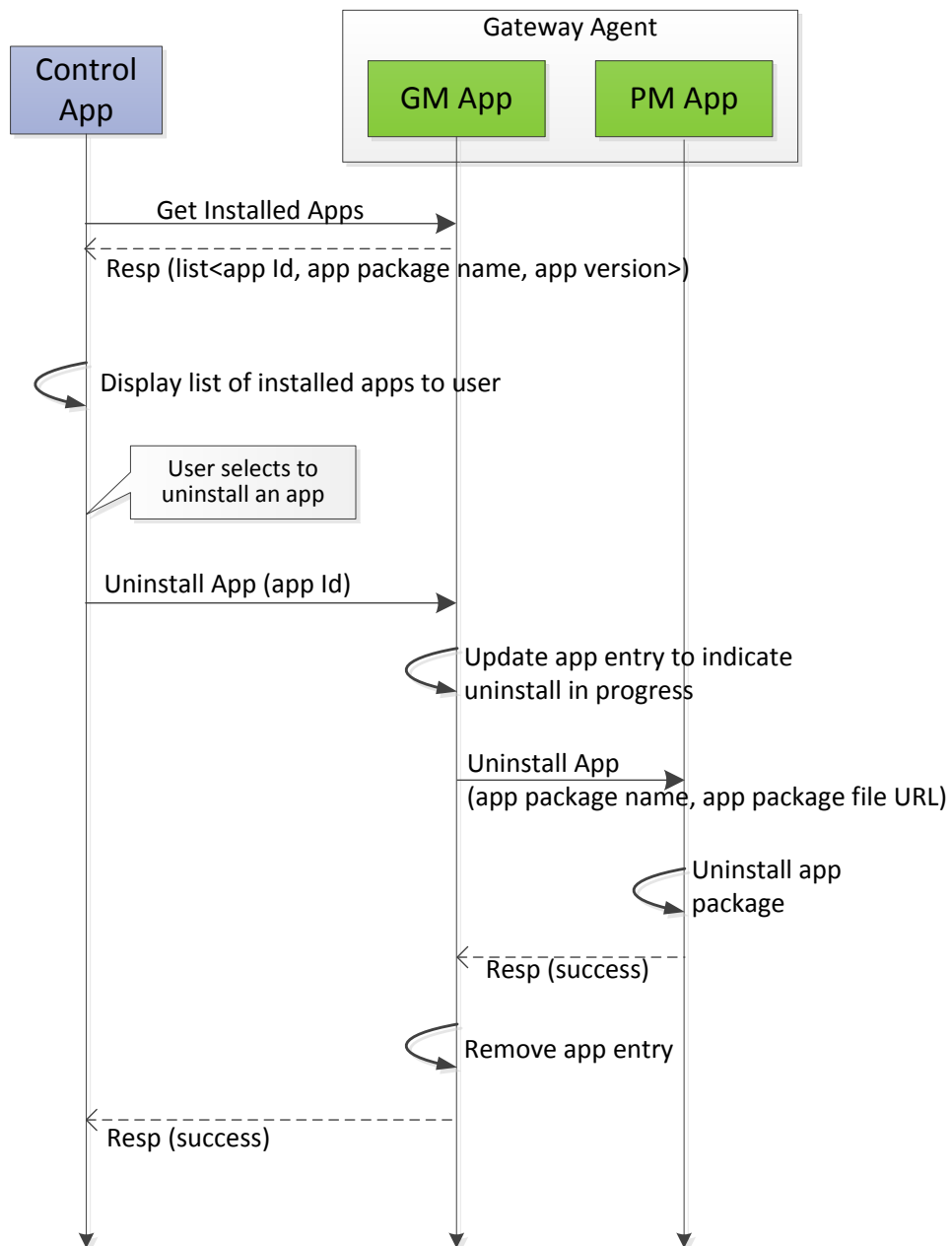


Figure 3-4 Connector app uninstall

3.2.4 Connector app restart

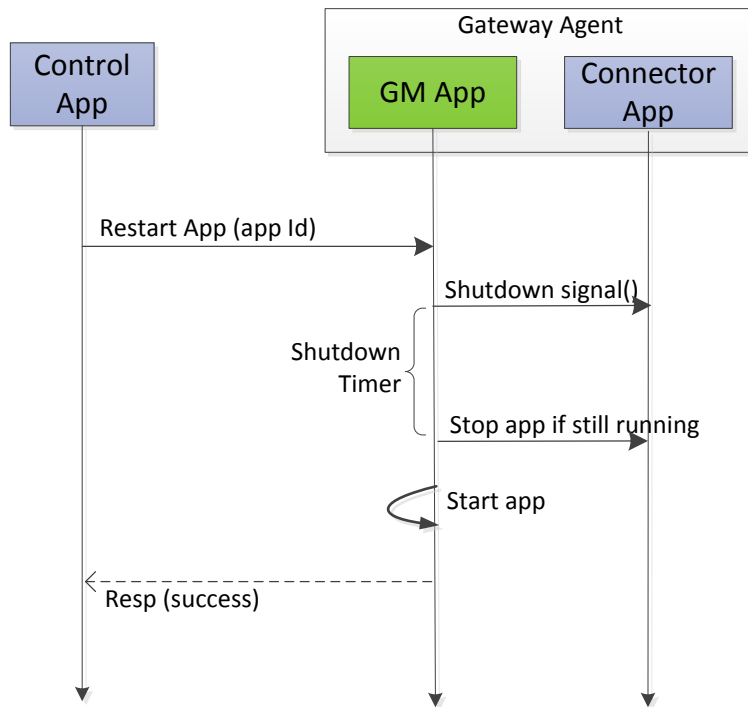


Figure 3-5 Connector app restart

3.2.5 Add a new device/app

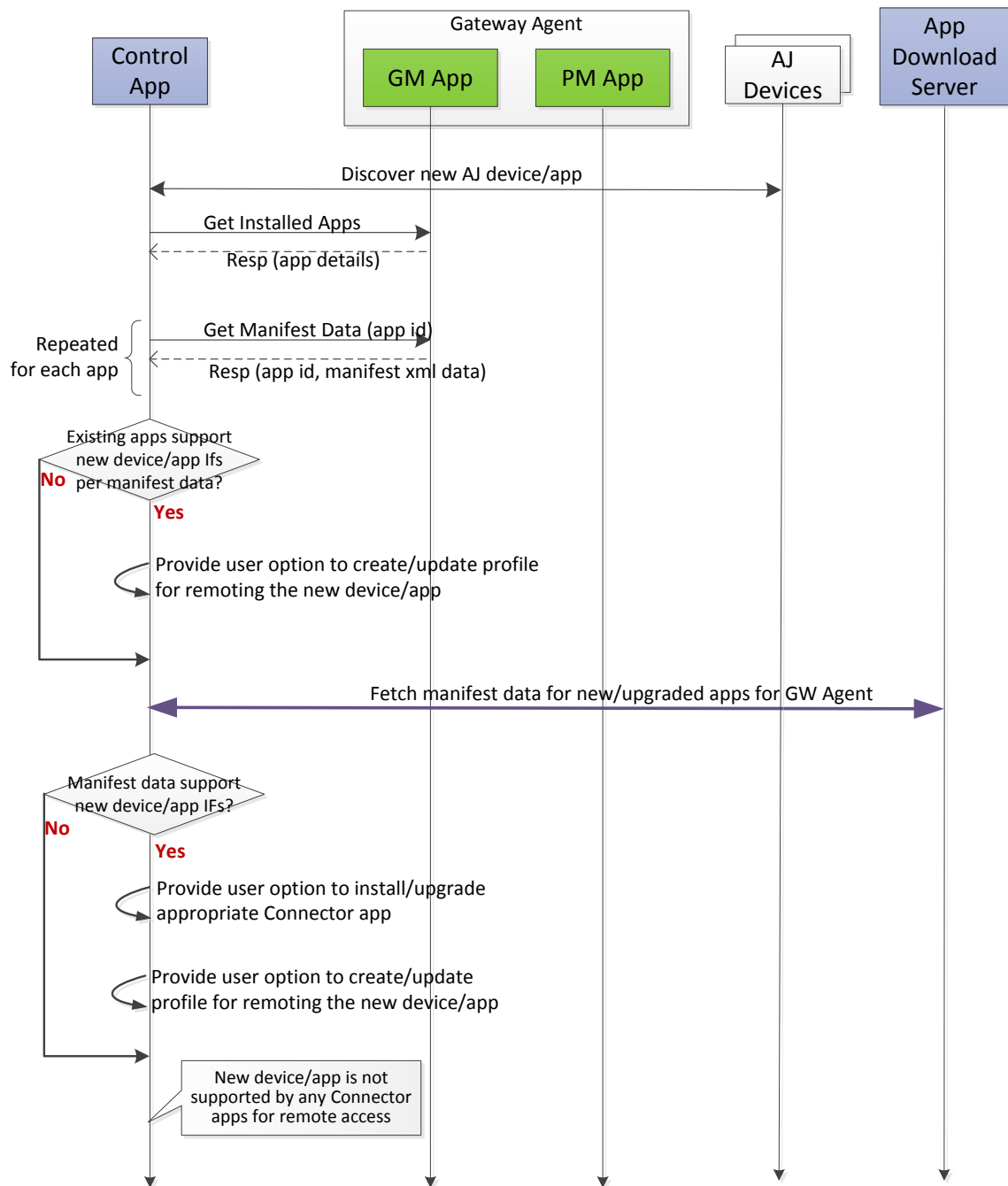


Figure 3-6 Add a new device/app

3.2.6 Remove a device

If a remote accessible IoE device is permanently removed from the proximal network, the Connector app and in turn the cloud service will get failure when trying to access that device. The user may or may not remove that device from the remote profile. When the user removes that device from the profile, the device will no longer be visible for remote access.

A remote accessible IoE device could be temporarily removed from the proximal network and come back in network after some time. The Connector app and in turn the cloud service will get failure when trying to access the device during the period it was removed. The remote access to device will be automatically restored when the device comes back in the proximal network.

3.3 Functional design

3.3.1 App Management interface

The App Management interface is an AllJoyn interface provided by the Gateway Management app and supports functional methods as summarized in Table 3-2.

Table 3-2 App Management interface functions

Functional method	Description
Install App	Results in download of the Connector app package from the URL provided and installs the package on the Gateway Agent in a new user account. The package signature is verified before installing the app. Also used for upgrading an existing Connector app to a newer version.
Uninstall App	Results in uninstalling the specified Connector app from the Gateway Agent.
Restart App	Results in restarting the specified Connector app. This call will fail if the profile associated with the Connector app is not active.
Get App Status	Returns status information for the specified Connector app including install status, operational status and cloud service connection status.
Get Installed Apps	Returns list of already installed Connector app on the Gateway Agent.
Get Manifest Data	Returns manifest file data for the specified Connector app.
Get Manifest Interfaces	Returns details about AllJoyn interfaces from the manifest file for a given Connector app.
Functional Signal	Description
App Status Changed	A signal sent out to Control app when status information changes for a Connector app. This is a session-based signal.

3.3.1.1 Install App

Table 3-3 provides a functional definition for the Install App method.

Table 3-3 Install App method

Method	Parameters		
	Parameter Name	In/Out	Description
Install App	App Id	In	App Id for the Connector app that is being installed.
	App Name	In	App name for the Connector app being installed.
	App package name	In	Package name for the Connector app that is being installed.
	App version	In	Version for the Connector app that is being installed.
	Download URL	In	Web URL to download the app package.
	App package file size	In	File size for the entire app package including manifest data.
	Upgrade Flag	In	Flag set to false for new app install and set to true for app upgrades.
	Install Status	Out	Indicates installation status for the Connector app. This will be set to 'Install in progress' or 'upgrade in progress'.

3.3.1.2 Uninstall App

Table 3-4 provides a functional definition for the Uninstall App method.

Table 3-4 Uninstall App method

Method	Parameters		
	Parameter Name	In/Out	Description
Uninstall App	App Id	In	App Id for the Connector app that needs to be uninstalled.
	Resp Status	Out	Indicates success/failure status for the uninstall of the Connector app.
	Install Status	Out (conditional)	Indicates uninstallation status for the Connector app. Only included if the Resp Status was a failure. This will be set to 'Installed'.
	Install Description	Out (conditional)	Provides description about the uninstall failure. Only included if the Resp Status was a failure.

3.3.1.3 Restart App

Table 3-5 provides a functional definition for the Restart App method.

Table 3-5 Restart App method

Method	Parameters		
	Parameter Name	In/Out	Description
Restart App	App Id	In	App Id for the Connector app that needs to be uninstalled.
	Resp Status	Out	Indicates success/failure status for the restart of the Connector app.

3.3.1.4 Get App Status

Table 3-6 provides a functional definition for the Get App Status method.

Table 3-6 Get App Status method

Method	Parameters		
	Parameter Name	In/Out	Description
Get App Status	App Id	In	App Id for the Connector app that needs to be uninstalled.
	Install Status	Out	Indicates installation status for the Connector app.
	Install Description	Out (optional)	Provides description indicating the results of the last app install/upgrade/uninstall operation done for the Connector app.
	Connection Status	Out	Indicates connection status of the Connector app with the cloud service.
	Operational Status	Out	Indicates the operational status for the Connector app.

3.3.1.5 Get Installed Apps

Table 3-7 provides a functional definition for the Get Installed Apps method.

Table 3-7 Get Installed App method

Method	Parameters		
	Parameter Name	In/Out	Description
Get Installed App	List <Installed App Info>		Response provides information on the list of Connector apps that are already installed on the Gateway Agent.
	App Id	Out	App Id for the installed Connector app.
	App name	Out	App name for the installed Connector app.
	App version	Out	Version for the installed Connector app.
	App object path	Out	Object path for accessing app-specific information.

3.3.1.6 Get Manifest Data

Table 3-9 provides a functional definition for the Get Manifest Data method.

Table 3-8 Get Manifest Data Method

Method	Parameters		
	Parameter Name	In/Out	Description
Get Manifest Data	App Id	In	App Id for the installed Connector app.
	Manifest Data	Out	Provides manifest data blob.

3.3.1.7 Get Manifest Interfaces

Table 3-9 provides a functional definition for the Get Manifest Interfaces method.

Table 3-9 Get Manifest Interfaces method

Method	Parameters		
	Parameter Name	In/Out	Description
Get Manifest Interfaces	App Id	In	App Id for the installed Connector app.
	List <Supported interfaces>		Provides manifest data blob.
	Interface name	Out	Interface name for AllJoyn interface supported by the Connector app.
	Interface friendly name	Out	User friendly name for the AllJoyn interface.
	List <Desired Interfaces>		
	Interface name	Out	Interface name for AllJoyn interface the Connector app would like to access.
	Interface friendly name	Out	User friendly name for the AllJoyn interface.

3.3.1.8 App Status Changed signal

Table 3-9 provides a functional definition for the Get Manifest Interfaces method.

Table 3-10 Get Manifest Interfaces method

Signal	Parameters		
	Parameter Name	In/Out	Description
App Status Changed signal	Connector App Id	In	Connector app Id associated with the profile.
	Install Status	In	Indicates installation status for the Connector app.

Signal	Parameters		
	Install Description	In (optional)	Provides description indicating the results of the last app install/upgrade/uninstall operation done for the Connector app.
	Connection Status	In	Indicates connection status of the Connector app with the cloud service.
	Operational Status	In	Indicates the operational status for the Connector app.

3.3.2 Package Manager app functions

The Package Manager app provides install and uninstall functionality as captures in

Table 3-11 Package Manager app functions

Logical function	Description
InstallApp	Results in the download of the Connector app package from the URL provided and installs the package on the Gateway Agent in a new user account. The package signature is verified before installing the app. Also used for upgrading an existing app to a newer version.
UninstallApp	Results in uninstalling the specified Connector app from the Gateway Agent.

3.3.2.1 InstallApp

Table 3-12 provides a list of input and output parameters for the InstallApp logical function provided by the Package Manager app.

Table 3-12 InstallApp function

Logical function	Parameters		
	Parameter Name	In/Out	Description
InstallApp	App Id	In	App Id for the Connector app that is being installed.
	App package name	In	Package name for the Connector app that is being installed.
	App version	In	Version for the Connector app that is being installed.
	Download URL	In	Web URL to download the app package.
	App package file size	In	File size for the entire app package including manifest data.
	Upgrade Flag	In	Flag set to false for new app install and set to true for app upgrades.
	App user Id	In	UNIX user Id to be used when installing/upgrading the app. Set to a new unique user Id for new app installs. Set to the current app user id for app upgrade.

Logical function	Parameters		
	Resp Status	Out	Indicates success/failure status for the install/upgrade of the Connector app.

3.3.2.2 UninstallApp

Table 3-13 provides a list of input and output parameters for the UninstallApp logical function provided by the Package Manager app.

Table 3-13 UninstallApp function

Logical function	Parameters		
	Parameter Name	In/Out	Description
UninstallApp	App package name	In	Package name for the Connector app that is being uninstalled.
	App package file URL	In	File location for the Connector app package that is being uninstalled.
	Resp Status	Out	Indicates success/failure status for the install/upgrade of the Connector app.

3.3.3 AllJoyn package format

Connector app packages should be made available in an AllJoyn package format. The app package file should include following in a single binary:

- Manifest file providing metadata for the app
- App package files

Each Connector app package will have access to a predefined set of standard OpenWRT libraries. If an app needs access to more libraries, those additional libraries need to be bundled along with the app package.

3.3.3.1 Manifest file

Each Connector app will have an associated manifest file that will provide metadata related to the app. The manifest file will reside in a predefined location in the app package directory (e.g., root directory of the app package). The manifest file should include information captured in Table 3-14.

Table 3-14 Manifest file content

Fields	Description
App Id	AllJoyn identifier for the Connector app.
App Package Name	Package name for the Connector app package.
App version	Connector app version.
Min AJ SDK version	Minimum version of AllJoyn SDK version required by the Connector app.
List <Desired AllJoyn Interfaces for access>	List of AllJoyn interfaces the Connector app would like to access.

List <AllJoyn interfaces app supports>	List of AllJoyn interfaces the Connector app supports that can be made available for access by AllJoyn devices in the proximal network.
--	---

3.3.3.2 Signature verification

The app package file will get signed with an app certificate issued to the Connector app developer. The app signing certificate should chain up to a root certificate maintained at the Package Manager app.

NOTE

Self-signed certificates are not used.

As part of app package signing, both the manifest file content and app files get signed. The signature and the app certificate (X.509 format) get attached to the app package binary.

The Package Manager app will verify that the received app certificate chains up to the same root certificate that it maintains. The Package Manager app then verifies the received signature using the public key received in the app certificate.

3.3.4 Fields

This section provides definition for some of fields related to Connector app management functionality.

3.3.4.1 Connection status

Table 3-15 Connection status

Value	Description
Not initialized	Connection with the cloud service is not initialized as the profile is not active yet.
In-progress	Connection is in the process of being established.
Connected	Connection with the cloud service is established and operational.
Not connected	Connection with the cloud service is down.
Error	There is an error condition in the Connector app.

3.3.4.2 Install status

Table 3-16 Install status

Value	Description
Installed	Connector app is installed on the Gateway Agent.
Install in progress	Connector app is currently being installed on the Gateway Agent.
Upgrade in progress	Connector app is currently being upgraded on the Gateway Agent.
Uninstall in progress	Connector app is currently being uninstalled on the Gateway Agent.

Install failed	App installation failed for the Connector app. An app entry with this status is maintained for a certain time period so that the Control app can fetch this status. After that time period, the app entry is removed.
----------------	---

3.3.4.3 Install description

This field provides a textual description about the results of the last app install/upgrade/uninstall operation done for that app. This could indicate 'app upgrade failed' or 'app uninstall failed'. In both these cases, the Install Status field will be set to 'Installed'.

3.3.4.4 Operational status

Table 3-17 Operational status

Value	Description
Running	Connector App is running
Stopped	Connector App has stopped

4 Performance

[GWA-HLD-0001] The Gateway Management application shall support performance and scalability requirements as captured by exit criteria defined in the *Gateway Agent 1.0 PRD*.

5 Requirements

This section captures system requirements for the Gateway Agent.

5.1 Overall requirements

Table 5-1 Gateway Management application requirements

Requirement	Description
GWA-HLD-0002	The Gateway Management Application shall support the org.alljoyn.GWAgent.ProfileMgmt interface.
GWA-HLD-0003	The Gateway Management Application shall support the org.alljoyn.GWAgent.AppAccess interface.
GWA-HLD-0004	The Gateway Management Application shall support the org.alljoyn.GWAgent.AppMgmt interface.
GWA-HLD-0005	The org.alljoyn.GWAgent.ProfileMgmt interface shall be a secure interface.
GWA-HLD-0006	The org.alljoyn.GWAgent.AppMgmt interface shall be a secure interface.
GWA-HLD-0007	The Gateway Management Application shall store the profile data persistently.
GWA-HLD-0008	The Gateway Management Application shall support only one profile for a given Connector application.
GWA-HLD-0009	The Gateway Management Application shall support profiles for multiple Connector applications.
GWA-HLD-0010	The Gateway Management Application shall support updating the config file policy rules to control communication between Connector applications and AllJoyn devices.
GWA-HLD-0011	The AllJoyn Router shall enforce config file policy rules when passing messages to/from Connector applications.
GWA-HLD-0012	The Gateway Management Application shall create firewall rules to block Connector applications from opening IP connections to other devices (AllJoyn or non-AllJoyn) on the proximal Wi-Fi network.
GWA-HLD-0081	The Gateway Management Application shall support ALLJOYN_SRP_KEYX auth mechanism for secure AllJoyn interfaces.
GWA-HLD-0082	The Gateway Management Application shall be configured with a default password to be used for secure AllJoyn interfaces.
GWA-HLD-0083	The Gateway Management Application shall support Config service to update the password for secure AllJoyn interfaces.

5.2 Gateway discovery

Table 5-2 Gateway discovery requirements

Requirement	Description
GWA-HLD-0014	The Gateway Management Application shall announce the Profile Management interface(s) in the About announcement signal.
GWA-HLD-0015	The Gateway Management Application shall announce the Application Management interface(s) in the About announcement signal.
GWA-HLD-0016	The Gateway Management Application shall populate the AppName field in the announcement signal with the configured value for the 'App Name' parameter.

5.3 Profile management

Table 5-3 Profile management requirements

Requirement	Description
GWA-HLD-0017	The Gateway Management Application shall support functional methods for Profile Management interface as per Table 2-1.
GWA-HLD-0018	The Gateway Management Application shall support behavior as per Figure 2-6 for creating a profile.
GWA-HLD-0019	The Gateway Management Application shall support behavior as per Figure 2-7 for activating a profile.
GWA-HLD-0020	The Gateway Management Application shall support behavior as per Figure 2-9 for updating a profile.
GWA-HLD-0021	The Gateway Management Application shall support behavior as per Figure 2-10 for deleting a profile.
GWA-HLD-0023	The Gateway Management Application shall support behavior as per Figure 2-11 for deactivating a profile.
GWA-HLD-0025	The Gateway Management Application shall support functional definition for Create Profile method as per Table 2-2.
GWA-HLD-0026	The Gateway Management Application shall support functional definition for Activate Profile method as per Table 2-3.
GWA-HLD-0027	The Gateway Management Application shall support functional definition for Get Profile method as per Table 2-4.
GWA-HLD-0084	The Gateway Management Application shall support functional definition for Get Profile Status method as per Table 2-5.
GWA-HLD-0085	The Gateway Management Application shall support functional definition for Get Profile Status method as per Table 2-6.
GWA-HLD-0028	The Gateway Management Application shall support functional definition for Update Profile method as per Table 2-7.
GWA-HLD-0029	The Gateway Management Application shall support functional definition for Delete Profile method as per Table 2-8.
GWA-HLD-0031	The Gateway Management Application shall support functional definition for Deactivate Profile method as per Table 2-9.

5.4 App access for Connector apps

Table 5-4 App access for Connector apps requirements

Requirement	Description
GWA-HLD-0033	The Gateway Management Application shall register the “org.alljoyn.GWAgent.GMApp” well-known name with the AllJoyn Router.
GWA-HLD-0034	Each Connector application shall register the “org.alljoyn.GWAgent.Connector.<Connector app id>” well-known name with the AllJoyn Router.
GWA-HLD-0035	The Gateway Management Application shall maintain profile data for each third-party app as part of a separate service object with a well-known object path of the format “/Profiles/<Connector app id>”.
GWA-HLD-0036	The Gateway Management Application shall support functional methods for the App Access interface as per Table 2-10.
GWA-HLD-0037	The Gateway Management Application shall support functional definition for the Get Profile method in the App Access interface as per Table 2-11.
GWA-HLD-0038	The Gateway Management Application shall provide profile information to the Connector app in the Get Profile response only if the associated profile is active.
GWA-HLD-0039	The Gateway Management Application shall support a functional definition for Update Connection Status method as per Table 2-12.
GWA-HLD-0040	The Gateway Management Application shall support a functional definition for Profile Updated signal as per Table 2-13.
GWA-HLD-0041	The Gateway Management Application shall send out the Profile Updated signal to a Connector app whenever the active profile associated with that app is updated.
GWA-HLD-0042	The Gateway Management Application shall support a functional definition for the Profile Deleted signal as per Table 2-14.
GWA-HLD-0043	The Gateway Management Application shall send out the Profile Deleted signal to a Connector app whenever the active profile associated with that app is deleted.
GWA-HLD-0044	The Gateway Management Application shall support a functional definition for Shutdown App signal as per Table 2-15.
GWA-HLD-0045	The Gateway Management Application shall send out the Shutdown App signal to a Connector app whenever it needs to stop that application.

5.5 Config file policy

Table 5-5 Config file policy requirements

Requirement	Description
GWA-HLD-0046	The Gateway Management Application shall use the UNIX user Id associated with a given Connector app when creating allow policy rules for that app in the config file.
GWA-HLD-0047	The Gateway Management Application shall use the latest unique name for removed AllJoyn apps when creating allow policy rules in the config file.

Requirement	Description
GWA-HLD-0048	At startup, the Gateway Management Application shall update the config file policy rules to allow announcements from all device apps in the proximal network for itself.
GWA-HLD-0049	At startup, the Gateway Management Application shall update the config file to add a deny-all policy rule for all the preinstalled Connector apps.
GWA-HLD-0050	After a new Connector app is installed, the Gateway Management Application shall update the config file to add a deny-all policy rule for that Connector app.
GWA-HLD-0051	After a profile is activated, the Gateway Management Application shall update the config file to add allow policy rules as per Table 2-17.
GWA-HLD-0052	After an active profile is deleted or deactivated, the Gateway Management Application shall update the config file to remove all the allow policy rules related to that profile.
GWA-HLD-0053	The Gateway Management Application shall start listening for announcements from AllJoyn apps after the very first profile is activated and should continue to receive announcements after that.
GWA-HLD-0054	The Gateway Management Application shall cache all the received announcements from AllJoyn apps.
GWA-HLD-0055	When the Gateway Management Application determines that the unique name associated with a remoted AllJoyn app has changed, it shall update the allow policy rules in the config file to reflect the new unique name.
GWA-HLD-0056	After updating the config file, the Gateway Management Application shall notify the AllJoyn Router via an AllJoyn method to trigger reload of the config file.

5.6 App management

Table 5-6 App management requirements

Requirement	Description
GWA-HLD-0057	The Gateway Management Application shall maintain details as per Table 3-1 for each Connector app.
GWA-HLD-0058	The Gateway Management Application shall maintain application details for Connector apps for which the install failed at least for a purge timer period.
GWA-HLD-0059	The Gateway Management Application shall get configured with the application details for preinstalled Connector apps as part of offline install procedure for those apps.
GWA-HLD-0060	The Package Manager Application shall provide app install, upgrade and uninstall functionality for Connector apps.
GWA-HLD-0061	The Package Manager Application shall support a command line interface for app install, upgrade and uninstall.
GWA-HLD-0062	The Gateway Management Application shall support functional methods for the App Management interface as per Table 3-2.
GWA-HLD-0063	The Gateway Management Application shall support behavior as per Figure 3-2 for Connector app installation.
GWA-HLD-0064	The Gateway Management Application shall support behavior as per Figure 3-3 for Connector app upgrade.

Requirement	Description
GWA-HLD-0065	The Gateway Management Application shall support behavior as per Figure 3-4 for Connector app uninstallation.
GWA-HLD-0066	The Gateway Management Application shall support behavior as per Figure 3-4 for Connector app restart.
GWA-HLD-0067	The Gateway Management Application shall support functional definition for the Install App method as per Table 3-3.
GWA-HLD-0068	The Gateway Management Application shall support functional definition for the Uninstall App method as per Table 3-4.
GWA-HLD-0069	The Gateway Management Application shall support functional definition for the Restart App method as per Table 3-5.
GWA-HLD-0070	The Gateway Management Application shall support functional definition for the Get App Status method as per Table 3-6.
GWA-HLD-0071	The Gateway Management Application shall support functional definition for the Get Installed Apps method as per Table 3-7.
GWA-HLD-0072	The Gateway Management Application shall support functional definition for the Get Manifest Data method as per Table 3-9.
GWA-HLD-0073	The Package Manager Application shall provide functions as per Error! Reference source not found..
GWA-HLD-0074	The Package Manager Application shall support parameters for the InstallApp function as per Table 3-12.
GWA-HLD-0075	The Package Manager Application shall support parameters for the UninstallApp function as per Table 3-13.
GWA-HLD-0076	The Package Manager Application shall support downloading Connector app packages over HTTPs from the download URL provided.
GWA-HLD-0077	The Package Manager Application shall support AllJoyn package format including manifest file and app package files.
GWA-HLD-0078	The manifest file for a Connector app shall include metadata as per Table 3-14.
GWA-HLD-0079	The Package Manager Application shall maintain a root certificate for Connector app package signature verification and shall verify that the received app certificate chains up to the same root certificate.
GWA-HLD-0080	The Package Manager Application shall verify the app package signature received as part of package binary.

6 Configuration Parameters

The Gateway Management app will support the following configuration parameters as per Table 6-1.

Table 6-1 Gateway Management app config parameters

Parameter	Default Value	Description
App name	“GM App”	Can be set to indicate a service provided-specific name for the Gateway Management app that gets advertised in the announcement signal.
Response timer	TBD	Defines timer value to wait for a response from the Connector app. A single configurable timer is used across all use cases where the Gateway Management app needs a response from a Connector app.
Shutdown timer	200ms	Defines timer value for Gateway Management app to wait after sending a shutdown signal to the Connector app. If the Connector app is still running when timer expires, it gets force killed.
Purge timer	TBD	Defines timer value to purge those app entries on the Gateway Management app for which the installation failed.

Appendix A Gateway Agent Architecture with Security 2.0

This section captures possible Gateway Agent architecture with Security 2.0.

A.1 Accessing secure interfaces with Security 2.0

Security 2.0 requires permissions to be set for accessing devices and apps on the AllJoyn network. Figure 6-1 shows the flow for setting these permissions and accessing devices/apps using these set permissions as it relates to the Gateway Agent. For details, refer to the *AllJoyn™ Security 2.0 HLD*.

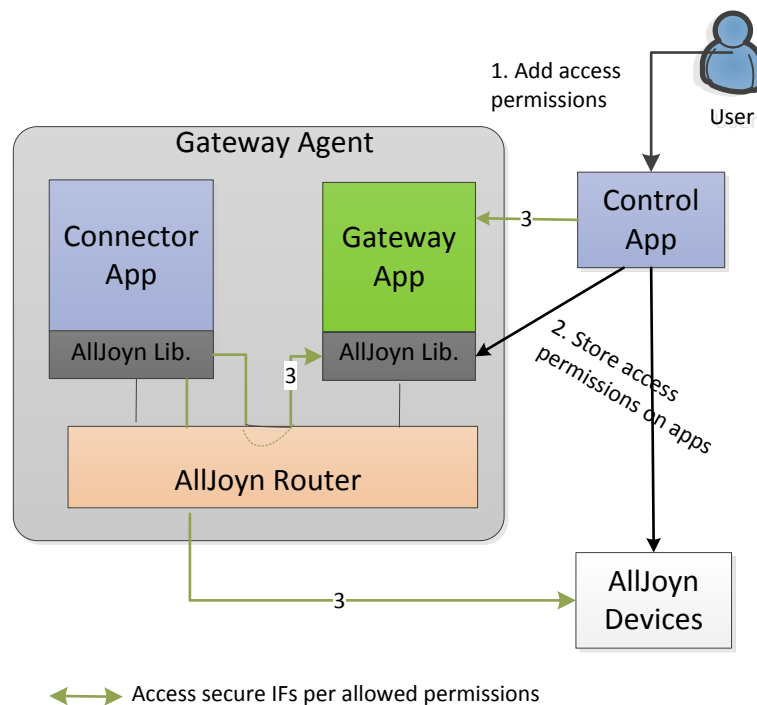


Figure 6-1 Gateway Agent Security 2.0