ALLSEEN ALLIANCE

AllJoyn™ Notification Service Framework Interface Definition

Version 14.06 Update 1

September 29, 2014

Contents

Figures

1 Introduction

1.1 Purpose

This document provides the specification for the AllJoyn™ Notification interface. This interface is used by an AllJoyn application to send events or state update notifications to other devices connected to an end user's home network, such as a Wi-Fi network.

1.2 Scope

This document is targeted to the developers for AllJoyn applications.

1.3 Release history

Release version	Date	What changed
Pre-14.02	N/A	Notification interface version 1 was added.
14.02	2/28/2014	The following interfaces were added: ■ Dismisser interface version 1 ■ Producer interface version 1
14.06	6/30/2014	No updates
14.06 Update 1	9/29/2014	■ Updated the document title (changed Specification to Definition). ■ Added the release version number to the title for version tracking. ■ Added a note in the Definition Overview chapter to address the AllSeen Alliance Compliance and Certification program. ■ Added a Mandatory column for method and signal parameters to support the AllSeen Alliance Compliance and Certification program.

1.4 References

Except for supporting information, the following are reference documents found on the AllSeen Alliance web site's Docs/Downloads section.

- *AllJoyn™ Framework Tutorial*
- *Introduction to AllJoyn™ Thin Library*
- *AllJoyn™ Data Type Signature*

Documentation supporting the About feature are listed on the Service Framework tab for the supported platform such as Android, Linux, Thin Library, etc.

- *AllJoyn™ About Feature 1.0 Interface Specification*

1.5 Acronyms and terms

Term	Definition
AllJoyn device	An entity which has an AllJoyn application installed to send or receive notifications using the Notification service framework interface.
Consumer	Device that receives the notification and has a way to notify user such as a mobile phone or TV.
Notification message	A message sent by a producer specifying details of the notification including any notification text to be displayed to the user.
Notification service framework	Software layer that enables devices to send or receive human-consumable notifications.
Producer	Device that generates and sends the notification to a device such as a household appliance.
Sessionless signal	A broadcast AllJoyn signal which is received by all devices listening on the end user's home network (such as the Wi-Fi network). The sessionless signals are broadcast on the network until an associated time-to-live (TTL) value expires. The Notification service framework sends notification messages as sessionless signals over the Wi-Fi network.

2 Definition Overview

2.1 Introduction

This document captures the design for the Notification service framework, which is a software layer that enables Alljoyn devices to send notifications to other AllJoyn devices. These devices are categorized as producers and consumers. Producers produce and send notifications, while consumers consume and display these notifications. An end user's home (Wi-Fi) network can have multiple producers connected and generating notification messages, as well as multiple consumers connected and consuming these messages.

The Notification service framework supports text notification payload as well as rich notification media (icon and audio). For rich media, the notification message payload can include URL links or AllJoyn object path references to rich notification media. The consumer app receiving the notification message will fetch the rich notification media from the object path or the producer device.

The Notification service framework uses AllJoyn framework sessionless signal for delivering notification messages. The Notification service framework exposes the Notification Service API for application developers to deliver and receive notification messages. The device OEM uses the Notification service framework Producer API to send notification messages. The Notification service framework sends these notification messages over the AllJoyn sessionless signal transport mechanism and makes them available to consumer devices listening for sessionless signals. The consumer running the Notification service framework registers with the AllJoyn framework to receive notification messages. The application developer for the consumer device uses the Notification service framework Consumer API to register and receive notifications from any producer that is sending notification on the Wi-Fi network.

Note All methods and signals are considered mandatory to support the AllSeen Alliance Compliance and Certification program. Individual parameters for a given method or signal may be considered mandatory or optional, and are specified accordingly in this document.

2.2 Architecture

The Notification service framework implements the Notification interface which is the over-the-wire interface to deliver messages from producers to consumers. Application developers making use of the Notification service framework implement against the Notification service framework APIs (producer and consumer side). They do not implement the Notification interface.

Figure 1 illustrates the Notification service framework API and Notification interface on producers and consumers.

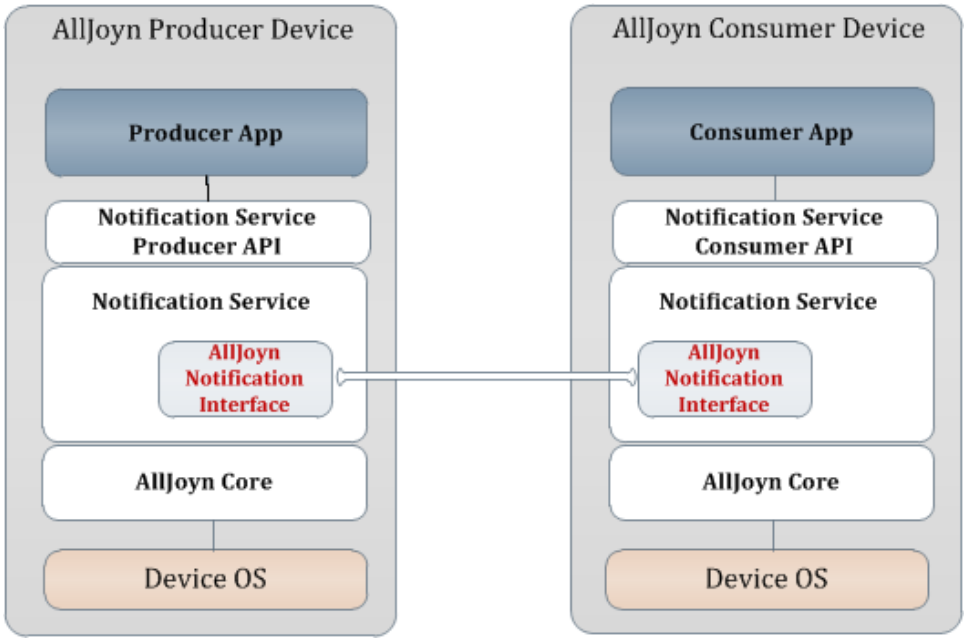

Figure 1: Notification service framework architecture within the AllJoyn framework

2.3 Typical call flow

Figure 2 illustrates a typical Notification service framework call flow with a single producer app generating a notification message. The message is then acquired by two consumer apps on the AllJoyn network.

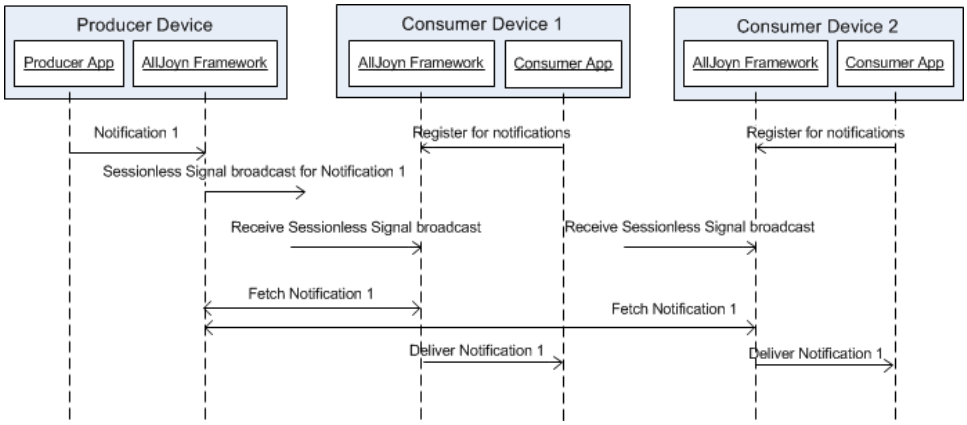

Figure 2: Typical Notification service framework call flow

The AllJoyn framework on the producer device does a sessionless signal broadcast for the notification message. This is received by the AllJoyn framework on the consumer

devices. The AllJoyn framework then fetches the notification message over unicast session from the producer AllJoyn core and delivers to the consumer application.

3 Specification

3.1 Notification messages

The notification message comprises a set of fields including message type and message TTL. These notification fields are specified by the producer app when sending notification message as part of Notification service framework Producer API.

3.1.1 Message type and TTL fields

The message type defines the type of notification messages (emergency, warning and information). Multiple types of notification messages can be sent at the same time by a producer. The message TTL defines the validity period of the notification message. Notification messages can be received by consumers that connect during the defined message TTL value.

Messages with the same message type will overwrite each other on the producer, so a consumer that connects to the network after the notification was sent will receive only the last of each message type.

3.1.2 Notification message behavior

The following behavior is supported using the Notification service framework.

- If another notification message of the same message type is sent by a producer app within the TTL period, the new message overwrites the existing message.
- If a consumer connects to the network after the TTL period expires, that consumer will not receive the message. For example, when a consumer such as a mobile phone is on the home network and the end user leaves the home; the consumer is no longer on the home network. The mobile phone will not receive notification messages when it reacquires the home network and the TTL of those notifications have expired.

Note The value is only used for message validity on the producer device. The TTL field is not sent as part of the notification message payload data over the end user's home network.

See *Notification Service Framework Use Cases* for use case scenarios related to notification message behavior.

3.1.3 Dismissing a notification

The dismiss notification is an option for consumers that have received the notification to let the producer know that this notification has been seen and there is no need to continue sending. It also lets other consumers know that the notification can be removed from the user display.

When a consumer attempts to dismiss a notification, the service framework creates a session with the producer using the original sender field sent in the notification.

Using the original sender field confirms that the notification is received by the actual producer and not the super agent in case the consumer received the notification from the super agent.

The producer will then send out a dismiss sessionless signal to notify the rest of the consumers in the network that this notification has been dismissed.

If the producer is not reachable, the consumer will send out the dismiss sessionless signal on its own.

3.2 Notification interface

The Notification interface is announced such that when a device scans the network, it can find all producer devices.

3.2.1 Interface name

Interface name	Version	Secured	Object path
org.alljoyn.Notification	1	no	■ /emergency ■ /warning ■ /info

3.2.2 Properties

Property name	Signature	List of values	Writable	Description
Version	q	Postive integers	no	Interface version number

3.2.3 Signals

Signal name	Parameters			Sessionless	Description
Notify	**Parameter name**	**Mandatory**	**Signature**	yes	AllJoyn signal carrying notification message.
	version	yes	q		Version of the Notification protocol.
	msgId	yes	i		Unique identification assigned to the notification message by the Notification service framework.
	msgType	yes	q		Type of notification message. ■ 0 – Emergency ■ 1 – Warning

Signal name	Parameters			Sessionless	Description
					■ 2 – Information
	deviceId	yes	s		Globally unique identifier for a given AllJoyn-enabled device.
	deviceName	yes	s		Name for a given AllJoyn-enabled device.
	appId	yes	ay		Globally unique identifier (GUID) for a given AllJoyn application.
	appName	yes	s		Name for a given AllJoyn-enabled device.
	langText	yes	a{ss}		Language-specific notification text.
	attributes	no	a{iv}		Set of attribute and value pair. This is used to hold optional fields in the notification message payload. See *Attributes*.
	customAttributes	no	a{ss}		Set of attribute and value pair. This can be used by the OEMs to add OEM-specific fields to the notification message.

3.2.4 Data types

Type	Definition	Signature	Description
notificationMsg	version	short	Version of the Notification protocol.
	msgId	integer	Unique identification assigned to the notification message by the Notification service framework.
	msgType	short	Type of notification message. ■ 0 – Emergency ■ 1 – Warning ■ 2 – Information
	deviceID	string	Globally unique identifier for a given AllJoyn-enabled device.
	deviceName	string	Name for a given AllJoyn-enabled device.
	appID	array of bytes	Globally unique identifier (GUID) for a given AllJoyn application.
	appName	string	Name for a given AllJoyn-enabled device.
	List<langText>	langText	Language-specific notification text.

Type	Definition	Signature	Description
	List<attributes>	attributes	Set of attribute and value pair. This is used to hold optional fields in the notification message payload. See *Attributes*.
	List<customAttributes>	customAttributes	Set of attribute and value pair. This can be used by the OEMs to add OEM-specific fields to the notification message.
langText	langTag	string	Language associated with the notification text. This is set as per RFC 5646.
	text	string	Notification message text in UTF-8 character encoding.
attributes	attrName	int	Name of the attribute.
	attrValue	variant	Value of the attribute.
customAttributes	attrName	string	Name of the attribute.
	attrValue	string	Value of the attribute.

Note If the richIconUrl, richAudioUrl, richIconObjectPath, richAudioObjectPath, or respObjectPath fields were specified by the producer app for a notification message, the Notification service framework sends this information as attributes in the attributes field, as per *Attributes*.

3.2.5 Attributes

Attribute	Values
Rich Notification Icon Url	■ attrName=0 ■ attrValue= ❒ variant signature=s ❒ value=<Icon URL>
Rich Notification Audio Url	■ attrName=1 ■ attrValue= ❒ variant signature=a{ss} ❒ value=List<langTag, Audio URL>
Rich Notification Icon Object Path	■ attrName=2 ■ attrValue= ❒ variant signature=o ❒ value= <Rich notification icon object path>
Rich Notification Audio Object Path	■ attrName=3

Attribute	Values
	■ attrValue= ❒ variant signature=o ❒ value= <Rich notification audio object path>
Response Object Path	■ attrName=4 ■ attrValue= ❒ variant signature=o ❒ value= <Response object path>
Original Sender	■ attrName=5 ■ attrValue= ❒ variant signature=s ❒ value= <Producer bus name>

3.2.6 Introspect XML

The following XML provides the Notification interface introspection XML.

```
<?xml version="1.0" encoding="UTF-8" ?>
<node
xsi:noNamespaceSchemaLocation="https://www.allseenalliance.org/schemas/introspect.xsd"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <interface name="org.alljoyn.Notification">
             <property name="Version" type="q" access="read"/>
            <signal name="Notify">
                <arg name="version" type="q"/>
                <arg name="msgId" type="i"/>
                <arg name="msgType" type="q"/>
                <arg name="deviceId" type="s"/>
                <arg name="deviceName" type="s"/>
                <arg name="appId" type="ay"/>
                <arg name="appName" type="s"/>
                <arg name="langText" type="a{ss}"/>
                <arg name="attributes" type="a{iv}"/>
                <arg name="customAttributes" type="a{ss}"/>
            </signal>
      </interface>
        </node>
```

3.3 Notification Producer interface

The Notification Producer interface is announced such that, when a device scans the network, it can find all producer devices.

3.3.1 Interface name

Interface name	Version	Secured	Object path
org.alljoyn.Notification.Producer	1	no	/notificationProducer

3.3.2 Properties

Property name	Signature	List of values	Writable	Description
Version	q	Positive integers	no	Interface version number

3.3.3 Methods

Method name	Parameters			Description
Dismiss	**Name**	**Mandatory**	**Data type**	A way to notify the producer that a notification was dismissed.
	msgId	yes	integer	

3.3.4 Introspect XML

The following XML provides the Notification Producer interface introspection XML.

```
<?xml version="1.0" encoding="UTF-8" ?>
<node
xsi:noNamespaceSchemaLocation="https://www.alljoyn.org/schemas/introspect.xsd"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <interface name="org.alljoyn.Notification.Producer">
        <method name="Dismiss">
            <arg name="msgId" type="i" direction="in"/>
        </method>
        <property name="Version" type="q" access="read"/>
    </interface>
</node>
```

3.4 Dismisser interface

The Dismiss sessionless signals are sent to notify other consumers on the proximal network that a notification has been dismissed.

3.4.1 Interface name

Interface name	Version	Secured	Object path
org.alljoyn.Notification.Dismisser	1	no	/notificationDismisser

3.4.2 Properties

Property name	Signature	List of values	Writable	Description
Version	q	Positive integers	no	Interface version number

3.4.3 Signals

Signal name	Parameters			Sessionless	Description
Dismiss	**Name**	**Mandatory**	**Signature**	yes	A way to notify consumers that the notification has been dismissed.
	msgId	yes	integer		
	appId	yes	array fo bytes		

3.4.4 Introspect XML

The following XML provides the Notification Dismisser interface introspection XML.

```
<?xml version="1.0" encoding="UTF-8" ?>
<node
xsi:noNamespaceSchemaLocation="https://www.alljoyn.org/schemas/introspect.xsd"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <interface name="org.alljoyn.Notification.Dismisser">
        <signal name="Dismiss">
            <arg name="msgId" type="i" direction="in"/>
            <arg name="appId" type="ay" direction="in"/>
        </signal>
        <property name="Version" type="q" access="read"/>
    </interface>
</node>
```

4 Notification Service Framework Use Cases

This chapter provides use case scenarios for how notification messages are handled.

4.1 Device connecting within and outside the TTL period

Figure 3 illustrates two consumers (television and tablet) connecting within the notification message TTL period and a third consumer (smartphone) connecting after the TTL period. The first two consumers receive the notification message, the third consumer does not.

Note The AllJoyn core block represents the collective AllJoyn framework functionality on various producers and consumers.

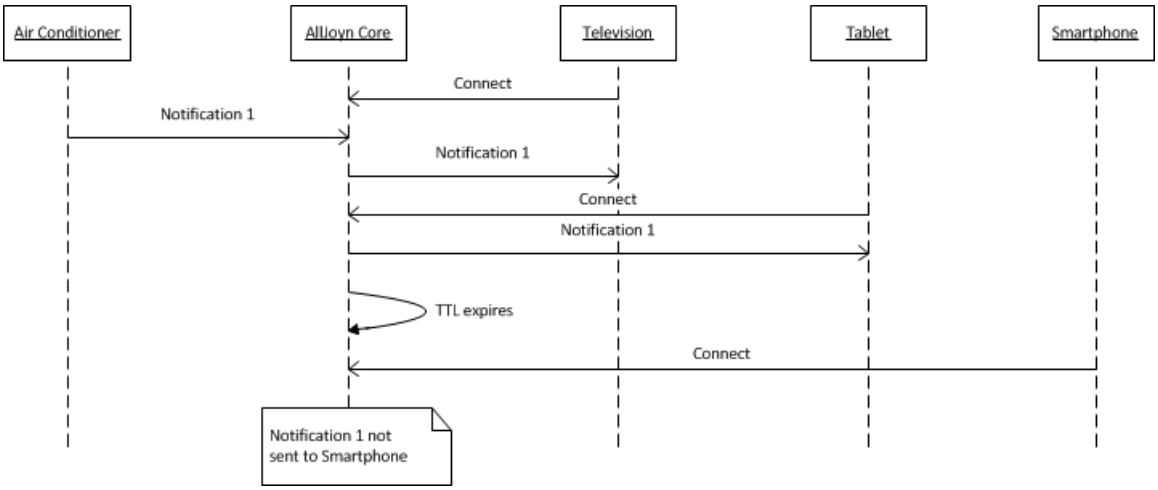

Figure 3: Notification message behavior within and outside the TTL period

4.2 Notification message handling based on message types

Figure 4 illustrates how a notification message overwrites a notification message of the same type, and how notification messages of different types can coexist using the AllJoyn framework.

Note The AllJoyn core block represents the collective AllJoyn framework functionality on various producers and consumers.

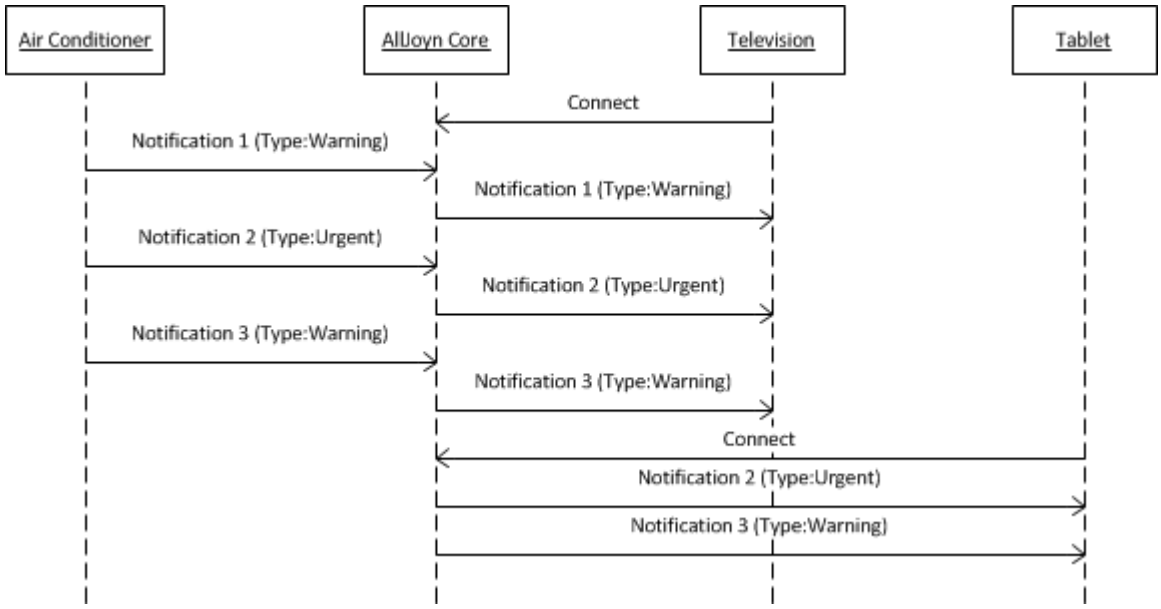

Figure 4: Notification message handling based on message type

4.3 Notifications dismissed when producer is on network

Figure 5 illustrates the flow of dismissing a notification from the consumer until it is received by other consumers on the network.

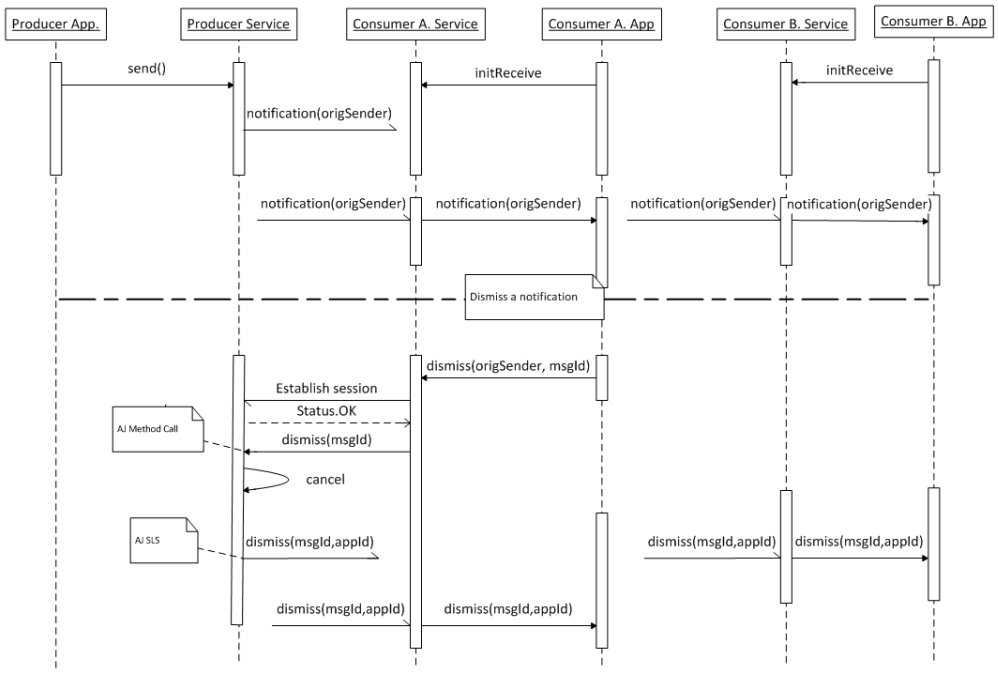

Figure 5: Notifications that are dismissed when the producer is on the network